Constrained Non-Monotone Submodular Maximization: Offline and Secretary Algorithms

Anupam Gupta * Aaron Roth † Grant Schoenebeck ‡ Kunal Talwar §

Abstract

Constrained submodular maximization problems have long been studied, with near-optimal results known under a variety of constraints when the submodular function is *monotone*. The case of non-monotone submodular maximization is less understood: the first approximation algorithms even for the unconstrainted setting were given by Feige et al. *(FOCS '07)*. More recently, Lee et al. *(STOC '09, APPROX '09)* show how to approximately maximize non-monotone submodular functions when the constraints are given by the intersection of p matroid constraints; their algorithm is based on local-search procedures that consider p-swaps, and hence the running time may be $n^{\Omega(p)}$, implying their algorithm is polynomial-time only for constantly many matroids.

In this paper, we give algorithms that work for p-*independence systems* (which generalize constraints given by the intersection of p matroids), where the running time is $\text{poly}(n, p)$. Our algorithm essentially reduces the non-monotone maximization problem to multiple runs of the greedy algorithm previously used in the monotone case. Our idea of using existing algorithms for monotone functions to solve the non-monotone case also works for maximizing a submodular function with respect to *a knapsack constraint*: we get a simple greedy-based constant-factor approximation for this problem.

With these simpler algorithms, we are able to adapt our approach to constrained non-monotone submodular maximization to the *(online) secretary setting*, where elements arrive one at a time in random order, and the algorithm must make irrevocable decisions about whether or not to select each element as it arrives. We give constant approximations in this secretary setting when the algorithm is constrained subject to a uniform matroid or a partition matroid, and give an $O(\log k)$ approximation when it is constrained by a general matroid of rank k.

*Carnegie Mellon University, Pittsburgh, PA. Email: anupamg@cs.cmu.edu
†Carnegie Mellon University, Pittsburgh, PA. Email: alroth@cs.cmu.edu
‡UC Berkeley, Berkeley, CA. Email: grant@cs.berkeley.edu
§Microsoft Research, Mountain View, CA. Email: kunal@microsoft.com

1 Introduction

We present algorithms for maximizing (not necessarily monotone) non-negative submodular functions satisfying $f(\emptyset) = 0$ under a variety of constraints considered earlier in the literature. Lee et al. [LMNS09, LSV09] gave the first algorithms for these problems via local-search algorithms: in this paper, we consider greedy approaches that have been successful for *monotone* submodular maximization, and show how these algorithms can be adapted very simply to non-monotone maximization as well. Using this idea, we show the following results:

- We give an $O(p)$-approximation for maximizing submodular functions subject to a p-independence system. This extends the result of Lee et al. [LMNS09, LSV09] which applied to constraints given by the intersection of p matroids, where p was a constant. (Intersections of p matroids give p-indep. systems, but the converse is not true.) Our greedy-based algorithm has a run-time polynomial in p, and hence give polynomial-time algorithms for even non-constant values of p.
- We give a constant-factor approximation for maximizing submodular functions subject to a knapsack constraint. This greedy-based algorithm gives an alternate approach to solve this problem; Lee et al. [LMNS09] gave LP-rounding-based algorithms that achieved a $(5 + \epsilon)$-approximation algorithm for constraints given by the intersection of p knapsack constraints, where p is a constant.

Armed with simpler greedy algorithms for nonmonotone submodular maximization, we are able to perform constrained nonmonotone submodular maximization in several special cases in the secretary setting as well: when items arrive online in random order, and the algorithm must make irrevocable decisions as they arrive.

- We give an $O(1)$-approximation for maximizing submodular functions subject to a cardinality constraint and subject to a partition matroid. (Using a reduction of [BDG+09], the latter implies $O(1)$-approximations to e.g., graphical matroids.) Our secretary algorithms are simple and efficient.
- We give an $O(\log k)$-approximation for maximizing submodular functions subject to an arbitrary rank k matroid constraint. This matches the known bound for the *matroid secretary problem*, in which the function to be maximized is simply linear.

No prior results were known for submodular maximization in the secretary setting, even for *monotone* submodular maximization. (Concurrently and independently of our work, Bobby Kleinberg has given an algorithm similar to ours for monotone secretary submodular maximization under a cardinality constraint [Kle09].)

Where previous results exist, our approximation factors are worse by constant factors from the best known: maximizing nonmonotone submodular functions subject to (constant) $p \geq 2$ matroid constraints currently has a $(\frac{p^2}{p-1} + \epsilon)$ approximation due to a paper of Lee, Sviridenko and Vondrák [LSV09], and for $p = 1$ the best result is a 3.23-approximation by Vondrák [Von09]. While we have not tried hard to get the best possible constants, it seems likely that matching, or improving on, those results will need more than just choosing the parameters carefully. We leave such improvements as an open problem.

1.1 Our Main Ideas

At a high level, the simple yet crucial observation for the offline results is this: many of the previous algorithms and proofs for constrained monotone submodular maximization can be adapted to show that the set S produced by them satisfies $f(S) \geq \beta f(S \cup C^*)$, for some $0 < \beta \leq 1$, and C^* being an optimal solution. In the monotone case, the right hand side is at least $f(C^*) = \mathbf{OPT}$ and we are done. In the non-monotone case, we cannot do this. However, we observe that if $f(S \cap C^*)$ is a reasonable fraction of $\mathbf{OPT}$, then (approximately) finding the most valuable set within S would give us a large value—and since we work with constraints that are downwards closed, finding such a set is just *unconstrained* maximization on $f(\cdot)$ restricted to S, for which Feige et al. [FMV07] give good algorithms! On the other hand, if $f(S \cap C^*) \leq \epsilon\mathbf{OPT}$ and $f(S)$ is also too small, then one can show that deleting the elements in S and running the procedure again to find another set $S' \subseteq X \setminus S$ with $f(S') \geq \beta f(S' \cap (C^* \setminus S))$ would guarantee

a good solution! Details for the specific problems appear in the following sections; we first consider the simplest cardinality constraint case in Section 2 to illustrate the general idea, and then give more general results in Sections 3.1 and 3.2.

For the secretary case where the elements arrive in random order, algorithms were not known for the monotone case either—the main complication being that we cannot run a greedy algorithm (since the elements are arriving randomly), and moreover the value of an incoming element depends on the previously chosen set of elements. Moreover, to extend the results to the non-monotone case, one needs to avoid the local-search algorithms (which, in fact, motivated the above results), since these algorithms necessarily implement multiple passes over the input, while the secretary model only allows a single pass over it. The details on all these are given in Section 4.

1.2 Related Work

Monotone Submodular Maximization. The (offline) monotone submodular optimization problem has been long studied: Fisher, Nemhauser, and Wolsey [NWF78, FNW78] showed that the greedy and local-search algorithms give a $(1-1/\mathrm{e})$-approximation with cardinality constraints, and a $p+1$-approximation under p matroid constraints. In another line of work, [Jen76, KH78, HKJ80] showed that the greedy algorithm is a p-approximation for maximizing a *modular* (i.e., additive) function subject to a p-independence system. This proof extends to show a $(p+1)$-approximation for monotone submodular functions under the same constraints (see, e.g., [CCPV09]). Vondrák showed how to maximize monotone submodular functions subject to a single knapsack to within $(1-1/\mathrm{e})$ [Von08], beating the long-standing factor of $1/2$. Lee et al. [LSV09] give algorithms that beat the $1/(p+1)$-bound for p matroid constraints.

Knapsack constraints. Sviridenko [Svi04] extended results of Wolsey [Wol82] and Khuller et al. [KMN99] to show that a greedy-like algorithm with partial enumeration gives an $(1-1/\mathrm{e})$-approximation to monotone submodular maximization subject to a knapsack constraint. Kulik et al. [KST09] recently showed that one could get essentially the same approximation subject to a constant number of knapsack constraints.

Non-Monotone Submodular Maximization. In the non-monotone case, even the unconstrained problem is NP-hard (it captures max-cut). Feige, Mirrokni and Vondrák [FMV07] first gave constant-factor approximations for this problem. Lee et al. [LMNS09] gave the first approximation algorithms for constrained non-monotone maximization (subject to p matroid constraints, or p knapsack constraints); the approximation factors were improved by Lee et al. [LSV09]. The algorithms in the previous two papers are based on local-search with p-swaps and would take $n^{\Theta(p)}$ time. Recent work by Vondrák [Von09] gives much further insight into the approximability of submodular maximization problems.

Secretary Problems. The original secretary problem seeks to maximize the probability of picking the element in a collection having the highest value, given that the elements are examined in random order [Dyn63, Fre83, Fer89]. The problem was used to model item-pricing problems by Hajiaghayi et al. [HKP04]. Kleinberg [Kle05] showed that the problem of maximizing a *modular* function subject to a cardinality constraint in the secretary setting admits a $(1-\frac{\Theta(1)}{\sqrt{k}})$-approximation, where k is the cardinality. (We show that maximizing a *submodular* function subject to a cardinality constraint cannot be approximated to better than some universal constant, independent of the value of k.) Babaioff et al. [BIK07] wanted to maximize modular functions subject to matroid constraints, again in a secretary-setting, and gave constant-factor approximations for some special matroids, and an $O(\log k)$ approximation for general matroids having rank k. This line of research has seen several developments recently [BIKK07, DP08, KP09, BDG+09].

1.3 Preliminaries

Given a set S and an element e, we use $S+e$ to denote $S \cup \{e\}$. A function $f : 2^\Omega \to \mathbb{R}_+$ is *submodular* if for all $S, T \subseteq \Omega$, $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$. An alternative (but equivalent) characterization of submodular functions states that f is submodular if it represents a set function with *decreasing marginal utility*. That is, for all $S \subseteq T \subseteq \Omega$, and for all $e \in \Omega$, $f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T)$. Given f and $S \subseteq \Omega$, define $f_S : 2^\Omega \to \mathbb{R}$ as $f_S(A) := f(S \cup A) - f(S)$. The following facts are standard.

Proposition 1.1. *If f is submodular with $f(\emptyset) = 0$, then*

- *for any S, f_S is submodular with $f_S(\emptyset) = 0$, and*
- *f is also* subadditive*; i.e., for disjoint sets A, B, we have $f(A) + f(B) \geq f(A \cup B)$.*

Matroids. A *matroid* is a pair $\mathcal{M} = (\Omega, \mathcal{I} \subseteq 2^\Omega)$, where $\mathcal{I}$ contains $\emptyset$, if $A \in \mathcal{I}$ and $B \subseteq A$ then $B \in \mathcal{I}$, and for every $A, B \in \mathcal{I}$ with $|A| < |B|$, there exists $e \in B \setminus A$ such that $A + e \in \mathcal{I}$. The sets in $\mathcal{I}$ are called *independent*, and the *rank* of a matroid is the size of any maximal independent set (base) in $\mathcal{M}$. In a *uniform* matroid, $\mathcal{I}$ contains all subsets of size at most k. A *partition* matroid, we have groups $g_1, g_2, \ldots, g_k \subseteq \Omega$ with $g_i \cap g_j = \emptyset$ and $\cup_j g_j = \Omega$; the independent sets are $S \subseteq \Omega$ such that $|S \cap g_i| \leq 1$.

Unconstrained (Non-Monotone) Submodular Maximization. We use $\mathsf{FMV}_\alpha(S)$ to denote an approximation algorithm given by Feige, Mirrokni, and Vondrák [FMV07] for unconstrained submodular maximization in the non-monotone setting: it returns a set $T \subseteq S$ such that $f(T) \geq \frac{1}{\alpha} \max_{T' \subseteq S} f(T')$. In fact, Feige et al. present many such algorithms, the best approximation ratio among these is $\alpha = 2.5$ via a local-search algorithm, the easiest is a 4-approximation that just returns a uniformly random subset of S.

2 Submodular Maximixation subject to a Cardinality Constraint

We first give an offline algorithm for submodular maximization subject to a cardinality constraint. This is inspired by the local-search algorithm of Lee et al. [LMNS09], but uses greedy ideas and illustrates our simple approach; we will then build on this approach in the following sections. The proof of the following lemma is a slight adaptation of the one used for monotone functions subject to a cardinality constraint.

Lemma 2.1. *For any set $|C| \leq k$, the greedy algorithm returns a set S that satisfies $f(S) \geq \frac{1}{2} f(S \cup C)$.*

Proof. Suppose not. Then $f_S(C) = f(S \cup C) - f(S) > f(S)$, and hence there is at least one element $e \in C \setminus S$ that has $f_S(\{e\}) > \frac{f(S)}{|C \setminus S|} > \frac{f(S)}{k}$. Since we ran the greedy algorithm, at each step this element e would have been a contender to be added, and by submodularity, e's marginal value would have been only higher then. Hence the elements actually added in each of the k steps would have had marginal value more than e's marginal value at that time, which is more than $f(S)/k$. This implies that $f(S) > k \cdot f(S)/k$, a contradiction. □

Lemma 2.2 (Special Case of Claim 5 in [LMNS09])**.** *Given sets $C, S_1 \subseteq U$, let $C' = C \setminus S_1$, and $S_2 \subseteq U \setminus S_1$. Then $f(S_1 \cup C) + f(S_1 \cap C) + f(S_2 \cup C') \geq f(C)$.*

Proof. By submodularity, it follows that $f(S_1 \cup C) + f(S_2 \cup C') \geq f(S_1 \cup S_2 \cup C) + f(C')$. Again using submodularity, we get $f(C') + f(S_1 \cap C) \geq f(C) + f(\emptyset)$. Putting these together and using non-negativity of $f(\cdot)$, the lemma follows. □

We now give our algorithm Submod-Max-Cardinality (Figure 1) for submodular maximization: it has the same multi-pass structure as that of Lee et al., but uses the greedy analysis above instead of a local-search algorithm.

Theorem 2.3. *The algorithm Submod-Max-Cardinality is a $(4 + \alpha)$-approximation.*

Proof. Let C^* be the optimal solution with $f(C^*) = \mathbf{OPT}$. We know that $f(S_1) \geq \frac{1}{2} f(S_1 \cup C^*)$. Also, if $f(S_1 \cap C^*)$ is at least $\epsilon\,\mathbf{OPT}$, then we know that the α-approximate algorithm FMV_α gives us a value of at $(\epsilon/\alpha)\mathbf{OPT}$. Else,

$$f(S_1) \geq \tfrac{1}{2} f(S_1 \cup C^*) \geq \tfrac{1}{2} f(S_1 \cup C^*) + \tfrac{1}{2} f(S_1 \cap C^*) - \epsilon\,\mathbf{OPT}/2 \tag{1}$$

1: **let** $X_1 \leftarrow X$
2: **for** $i = 1$ to 2 **do**
3: **let** $S_i \leftarrow \text{Greedy}(X_1)$
4: **let** $S_i' \leftarrow \mathsf{FMV}_\alpha(S_i)$
5: **let** $X_{i+1} \leftarrow X_i \setminus S_i$.
6: **end for**
7: **return** best of S_1, S_1', S_2.

Figure 1: Algorithm Submod-Max-Cardinality

Similarly, we get that $f(S_2) \geq \frac{1}{2} f(S_2 \cup (C^* \setminus S_1))$. Adding this to (1), we get

$$\begin{aligned} 2\max(f(S_1), f(S_2)) &\geq f(S_1) + f(S_2) \\ &\geq \tfrac{1}{2}\big(f(S_1 \cup C^*) + f(S_1 \cap C^*) + f(S_2 \cup (C^* \setminus S_1))\big) - \epsilon \mathbf{OPT}/2 \quad (2) \\ &\geq \tfrac{1}{2} f(C^*) - \epsilon \mathbf{OPT}/2 \quad (3) \\ &\geq \tfrac{1}{2}(1-\epsilon)\,\mathbf{OPT}. \end{aligned}$$

where we used Lemma 2.2 to get from (2) to (3). Hence $\max\{f(S_1), f(S_2)\} \geq \frac{1-\epsilon}{4}\,\mathbf{OPT}$. The approximation factor now is $\max\{\alpha/\epsilon, 4/(1-\epsilon)\}$. Setting $\epsilon = \frac{\alpha}{\alpha+4}$, we get a $(4+\alpha)$-approximation, as claimed. □

Using the known value of $\alpha = 2.5$ from Feige et al. [FMV07], we get a 6.5-approximation for submodular maximization under cardinality constraints. While this is weaker than the 3.23-approximation of Vondrák [Von09], or even the 4-approximation we could get from Lee et al. [LMNS09] for this special case, the algorithm is faster, and the idea behind the improvement works in several other contexts, as we show in the following sections.

3 Fast Algorithms for p-Systems and Knapsacks

In this section, we show our greedy-style algorithms which achieve an $O(p)$-approximation for submodular maximization over p-systems, and a constant-factor approximation for submodular maximization over a knapsack. Due to space constraints, many proofs are deferred to the appendices.

3.1 Submodular Maximization for Independence Systems

Let Ω be a universe of elements and consider a collection $\mathcal{I} \subseteq 2^\Omega$ of subsets of Ω. $(\Omega, \mathcal{I})$ is called an *independence system* if (a) $\emptyset \in \mathcal{I}$, and (b) if $X \in \mathcal{I}$ and $Y \subseteq X$, then $Y \in \mathcal{I}$ as well. The subsets in $\mathcal{I}$ are called *independent*; for any set S of elements, an inclusion-wise maximal subset T of S is called a *basis* of S. For brevity, we say that T is a basis, if it is a basis of Ω.

Definition 3.1. Given an independence system $(\Omega, \mathcal{I})$ and a subset $S \subseteq \Omega$. The *rank* $r(S)$ is defined as the cardinality of the *largest* basis of S, and the *lower rank* $\rho(S)$ is the cardinality of the *smallest* basis of S. The independence system is called a p-independence system (or a p-system) if $\max_{S \subseteq \Omega} \frac{r(S)}{\rho(S)} \leq p$.

See, e.g., [CCPV09] for a discussion of independence systems and their relationship to other families of constraints; it is useful to recall that intersections of p matroids forms a p-independent system.

3.1.1 The Algorithm for p-Independence Systems

Suppose we are given an independence system $(\Omega, \mathcal{I})$, a subset $X \subseteq \Omega$ and a non-negative submodular function f that is potentially non-monotone, but has $f(\emptyset) = 0$. We want to find (or at least approximate) $\max_{S \subseteq X : S \in \mathcal{I}} f(S)$. The greedy algorithm for this problem is what you would expect: start with the set $S = \emptyset$, and at each step pick an element $e \in X \setminus S$ that maximizes $f_S(e)$ and ensures that $S \cup \{e\}$ is also independent. If no such element exists, the algorithm terminates, else we set $S \leftarrow S \cup \{e\}$, and repeat. (Ideally, we would also check to see if $f_S(e) \leq 0$, and terminate at the first time this happens; we don't do that, and instead we add elements even when the marginal gain is negative until we cannot add any more elements without violating independence.) The proof of the following lemma appears in Section A, and closely follows that for the monotone case from [CCPV09].

Lemma 3.2. *For a p-independence system, if S is the independent set returned by the greedy algorithm, then $f(S) \geq \frac{1}{p+1} f(O \cup S)$.*

The algorithm Submod-Max-p-Systems (Figure 2) for maximizing a non-monotone submodular function f with $f(\emptyset) = 0$ over a p-independence system now immediately suggests itself.

Theorem 3.3. *The algorithm Submod-Max-p-System is a $(1 + \alpha)(p + 2 + 1/p)$-approximation for maximizing a non-monotone submodular function over a p-independence system, where α is the approximation guarantee for unconstrained (non-monotone) submodular maximization.*

1: $X_1 \leftarrow X$
2: **for** $i = 1$ to $p + 1$ **do**
3: $\quad S_i \leftarrow \text{Greedy}(X_i, \mathcal{I}, f)$
4: $\quad S'_i \leftarrow \mathsf{FMV}_\alpha(S_i)$
5: $\quad X_{i+1} \leftarrow X_i \setminus S_i$
6: **end for**
7: **return** $S \leftarrow$ best among $\{S_i\}_{i=1}^{p+1} \cup \{S'_i\}_{i=1}^{p+1}$.

Figure 2: Submod-Max-p-System$(X, \mathcal{I}, f)$

Proof. Let C be an optimal solution with $\mathbf{OPT} = f(C)$, and let $C_i = C \cap X_i$ for all $i \in [p + 1]$—hence $C_1 = C$. Note that C_i is a feasible solution to the greedy optimization in Step 3. Hence, by Lemma 3.2, we know that $f(S_i) \geq \frac{1}{p+1} f(C_i \cup S_i)$. Now, if for some i, it holds that $f(S_i \cap C_i) \geq \epsilon\mathbf{OPT}$ (for $\epsilon > 0$ to be chosen later), then the guarantees of FMV_α ensure that $f(S'_i) \geq (\epsilon\mathbf{OPT})/\alpha$, and we will get a α/ϵ-approximation. Else, it holds for all $i \in [p + 1]$ that

$$f(S_i) \geq \tfrac{1}{p+1} f(C_i \cup S_i) + f(C_i \cap S_i) - \epsilon\,\mathbf{OPT}. \tag{4}$$

Now we can add all these inequalities and use [LMNS09, Claim 5] to infer that

$$f(S) \geq \tfrac{p}{(p+1)^2} f(C) - \epsilon\,\mathbf{OPT} = \mathbf{OPT}\left(\tfrac{p}{(p+1)^2} - \epsilon\right). \tag{5}$$

Thus the approximation factor is $\max\{\alpha/\epsilon, (\frac{p}{(p+1)^2} - \epsilon)^{-1}\}$. Setting $\epsilon = \frac{\alpha}{1+\alpha}\frac{p}{(p+1)^2}$, we get the claimed approximation ratio. □

Note that even using $\alpha = 1$, we would not be able to match the ratios in Lee et al. [LMNS09, LSV09]. However, the proof here is somewhat simpler and also works seamlessly for all p-independence systems instead of just intersections of matroids. Moreover the running time is only linear in the number of matroids, instead of being exponential as in the local-search. Note that running the algorithm just twice instead of $p + 1$ times reduces the run-time further; we can then use Lemma 2.2 instead of the full power of [LMNS09, Claim 5], and hence the constants are slightly worse.

3.2 Submodular Maximization over Knapsacks

The paper of Sviridenko [Svi04] gives a greedy algorithm with partial enumeration that achieves a $\frac{e}{e-1}$ approximation for *monotone* submodular maximization with respect to a knapsack constraint. In particular, each element $e \in X$ has a size c_e, and we are given a bound B: the goal is to maximize $f(S)$ over subsets $S \subseteq X$ such that $\sum_{e \in S} c_e \leq B$. His algorithm is the following—for each possible choice of at most three elements from X, start with those three elements and iteratively include the element which maximizes the gain in the function value per unit size, and the resulting set still fits in the knapsack. (If none of the remaining elements gives a positive gain, or fit in the knapsack, stop.) Finally, from among these $O(|X|^3)$ solutions, choose the best one—Sviridenko shows that in the monotone submodular case, this is an $(1 - 1/\mathrm{e})$-approximation algorithm.

One can modify Sviridenko's algorithm and proof to show the following result for non-monotone submodular functions. (The details are in Appendix B).

Theorem 3.4. *There is a polynomial-time algorithm that given the above input, outputs a (polynomial sized) collection of sets such that for any valid solution C, the collection contains a set S satisfying $f(S) \geq (1 - 1/\mathrm{e}) f(S \cup C)$.*

Now using an argument very similar to that in Theorem 2.3 gives us the following result for non-montone submodular maximization with respect to a knapsack constraint.

Theorem 3.5. *There is an $(\alpha + \frac{2\mathrm{e}}{\mathrm{e}-1})$-approximation for the problem of maximizing a submodular function with respect a knapsack constraint, where α is the approximation guarantee for unconstrained (non-monotone) submodular maximization.*

4 Constrained Submodular Maximization in the Secretary Setting

In this section, we will give algorithms for submodular maximization in the secretary setting: first subject to a cardinality constraint, then with respect to a partition matroid, and finally an algorithm for general matroids. The main algorithmic concerns tackled in this section when developing secretary algorithms are: (a) previous algorithms for non-monotone maximization required local-search, which seems difficult in an online secretary setting, so we developed greedy-style algorithms; (b) we need multiple passes for non-monotone optimization, and while that can be achieved using randomization and running algorithms in parallel, these parallel runs of the algorithms may have correlations that we need to control (or better still, avoid); and of course (c) the marginal value function changes over the course of the algorithm's execution as we pick more elements—in the case of partition matroids, e.g., this ever-changing function creates several complications.

We also show an information theoretic lower bound: no secretary algorithm can approximately maximize a submodular function subject to a cardinality constraint k to a factor better than some universal constant greater than 1, independent of k (This is ignoring computational constraints, and so the computational inapproximability of offline submodular maximization does not apply). This is in contrast to the additive secretary problem, for which Kleinberg gives a secretary algorithm achieving a $\frac{1}{1-5/\sqrt{k}}$ approximation [Kle05]. This lower bound is found in appendix D.

4.1 Subject to a Cardinality Constraint

The offline algorithm presented in Section 2 builds three potential solutions and chooses the best amongst them. We now want to build just one solution in an *online* fashion, so that elements arrive in random order, and when an element is added to the solution, it is never discarded subsequently. We first give an online algorithm that is given the optimal value **OPT** as input but where the elements can come in *worst-case* order (we call this an "online algorithm with advice"). Using sampling ideas we can estimate **OPT**, and hence use this advice-taking online algorithm in the secretary model where elements arrive in random order. (This algorithm was independently discovered by R.D. Kleinberg [Kle09].)

To get the advice-taking online algorithm, we make two changes. First, we do not use the greedy algorithm which selects elements of highest marginal utility, but instead use a *threshold algorithm*, which selects any element that has marginal utility above a certain threshold. Second, we will change Step 4 of Algorithm Submod-Max-Cardinality to use FMV_4, which simply selects a random subset of the elements to get a 4-approximation to the unconstrained submodular maximization problem [FMV07]. The *Threshold Algorithm* with inputs (τ, k) simply selects each element as it appears if it has marginal utility at least τ, up to a maximum of k elements.

Lemma 4.1 (Threshold Algorithm)**.** *Let C^* satisfy $f(C^*) = \mathbf{OPT}$. The threshold algorithm on inputs (τ, k) returns a set S that either has k elements and hence a value of τk, or a set S with value $f(S) \geq f(S \cup C^*) - |C^*|\tau$.*

Proof. The claim is immediate if the algorithm picks k elements, so suppose it does not pick k elements, and also $f(S) < f(S \cup C^*) - |C^*|\tau$. Then $f_S(C^*) > |C^*|\tau$, or $\tau < \frac{f_S(C^*)}{|C^*|} \leq \frac{\sum_{e \in C^*} f_S(e)}{|C^*|}$. By averaging, this implies there exists an element $e \in C^*$ such that $f_S(e) > \tau$; this element cannot have been chosen into S (otherwise the marginal value would be 0), but it would have been chosen into S when it was considered by the algorithm (since at that time its marginal value would only have been higher). This gives the desired contradiction. □

Theorem 4.2. *If we change Algorithm Submod-Max-Cardinality above to use the threshold algorithm with threshold $\tau = \frac{\mathbf{OPT}}{7k}$, and to use the random sampling algorithm FMV_4 in Step 4, and return a (uniformly) random one of S_1, S_1', S_2 in Step 7, the expected value of the returned set is at least $\mathbf{OPT}/21$.*

Proof. If S_1 or S_2 has k elements, then $f(S_1) + f(S_2) \geq \tau k = \mathbf{OPT}/7$; if $f(S_1 \cap C^*) \geq 4\tau k$, then FMV_4 guarantees that $f(S_1') \geq \tau k$. Else Lemma 2.2 implies $f(S_1) + f(S_2) \geq \mathbf{OPT} - 6\tau k = \tau k$. In all these

cases, $f(S_1) + f(S_1') + f(S_2) \geq \tau k = \frac{\mathbf{OPT}}{7}$, and picking a random one of these gets a third of that in expectation. □

Observation 4.3. *Given the value of* **OPT**, *the algorithm of Theorem 4.2 can be implemented in an online fashion where we (irrevocably) pick at most k elements.*

Proof. We can randomly choose which one of S_1, S_1', S_2 we want to output before observing any elements. Clearly S_1 can be determined online, as can S_2 by choosing any element that has high marginal value and is not chosen in S_1. Moreover, S_1' just selects elements from S_1 independently with probability $1/2$. □

Observation 4.4. *In both the algorithms of Theorems 2.3 and 4.2, if we use some value $Z \leq$* **OPT** *instead of* **OPT**, *the returned set has value at least $(4+\alpha)Z$, and expected value at least $21Z$, respectively.*

Finally, it will be convenient to recall Dynkin's algorithm: given a stream of n numbers randomly ordered, it samples the first $1/e$ fraction of the numbers and picks the next element that is larger than all elements in the sample.

4.1.1 The Secretary Algorithm for the Cardinality Case

For a constrained submodular optimization, if we are given *(a)* a ρ_{off}-approximate offline algorithm, and also *(b)* a ρ_{on}-approximate online advice-taking algorithm that works given an estimate of **OPT**, we can now get an algorithm in the secretary model thus: we use the offline algorithm to estimate **OPT** on the first half of the elements, and then run the advice-taking online algorithm with that estimate. The formal algorithm appears in Figure 3. Because of space constraints, we have deferred the proof of the following theorem to Appendix C

Let Solution $\leftarrow \emptyset$.
Flip a fair coin
if heads **then**
  Solution $\leftarrow$ most valuable item using Dynkin's-Algo
else
  Let $m \in B(n, 1/2)$ be a draw from the binomial distribution
  $A_1 \leftarrow \rho_{\text{off}}$-approximate offline algorithm on the first m elements.
  $A_2 \leftarrow \rho_{\text{on}}$-approximate advice-taking online algorithm with $f(A_1)$ as the guess for **OPT**.
  Return A_2
end if

Figure 3: **Algorithm** SubmodularSecretaries

Theorem 4.5. *The above algorithm is an $O(1)$-approximation algorithm for the cardinality-constrained submodular maximization problem in the secretary setting.*

4.2 Subject to a Partition Matroid Constraint

In this section, we give a constant-factor approximation for maximizing submodular functions subject to a partition matroid. Recall that in such a matroid, the universe is partitioned into k "groups", and the independent sets are those which contain at most one element from each group. To get a secretary-style algorithm for *modular (additive)* function maximization subject to a partition matroid, we can run Dynkin's algorithm on each group independently. However, if we have a submodular function, the marginal value of an element depends on the elements previously picked—and hence the marginal value of an element as seen by the online algorithm and the adversary become very different.

We first build some intuition by considering a simpler "contiguous partitions" model where all the elements of each group arrive together (in random order), but the groups of the partition are presented in some *arbitrary* order $g_1, g_2, \ldots, g_r$. We then go on to handle the case when all the elements indeed come in completely random order, using what is morally a reduction to the contiguous partitions case.

4.2.1 A Special Case: Contiguous Partitions

For the contiguous case, one can show that executing Dynkin's algorithm with the obvious marginal valuation function is a good algorithm: this is not immediate, since the valuation function changes as we pick some elements—but it works out, since the groups come contiguously. Now, as in the previous section, one wants to run two parallel copies of this algorithm (with the second one picking elements from among those not picked by the first)—but the correlation causes the second algorithm to not see a random permutation any more! We get around this by coupling the two together as follows:

> Initially, the algorithm determines whether it is one of 3 different modes (A, B, or C) uniformly at random. The algorithm maintains a set of selected elements, initially S_0. When group g_i of the partition arrives, it runs Dynkin's secretary algorithm on the elements from this group using valuation function $f_{S_{i-1}}$. If Dynkin's algorithm selects an element x, our algorithm flips a coin. If we are in modes A or B, we let $S_i \leftarrow S_{i-1} \cup \{x\}$ if the coin is heads, and let $S_i \leftarrow S_{i-1}$ otherwise. If we are in mode C, we do the reverse, and let $S_i \leftarrow S_{i-1} \cup \{x\}$ if the coin is tails, and let $S_i \leftarrow S_{i-1}$ otherwise. Finally, after the algorithm has completed, if we are in mode B, we discard each element of S_r with probability $1/2$. (Note that we can actually implement this step online, by 'marking' but not selecting elements with probability $1/2$ when they arrive).

Lemma 4.6. *The above algorithm is a $(3 + 6\mathrm{e})$-approximation for the submodular maximization problem under partition matroids, when each group of the partition comes as a contiguous segment.*

Proof. We first analyze the case in which the algorithm is in mode A or C. Consider a hypothetical run of *two* versions of our algorithm simultaneously, one in mode A and one in mode C which share coins and produce sets S_r^A and S_r^C. The two algorithms run with identical marginal distributions, but are coupled such that whenever both algorithms attempt to select the same element (each with probability $1/2$), we flip only one coin, so one succeeds while the other fails. Note that $S_r^C \subseteq U \setminus S_r^A$, and so we will be able to apply Lemma 2.2. For a fixed permutation π, let $S_r^A(\pi)$ be the set chosen by the mode A algorithm for that particular permutation. As usual, we define $f_A(B) = f(A \cup B) - f(A)$. Hence, $f(S_r^A(\pi)) = f(S_r^A(\pi) \cup C^*) - f_{S_r^A(\pi)}(C^*)$, and taking expectations, we get

$$\mathrm{E}[f(S_r^A)] = \mathrm{E}[f(S_r^A \cup C^*)] - \mathrm{E}[f_{S_r^A}(C^*)] \tag{6}$$

Now, for any $e \in X$, let $j(e)$ be the index of the group containing e; hence we have

$$\begin{aligned}\mathrm{E}[f_{S_r^A}(C^*)] &\leq \sum_{e \in C^*} \mathrm{E}[f_{S_r^A}(\{e\})] \leq \sum_{e \in C^*} \mathrm{E}[f_{S_{j(e)-1}^A}(\{e\})] \quad \leq \sum_{e \in C^*} 2\mathrm{e} \cdot \mathrm{E}[f_{S_{j(e)-1}^A}(\{Y_{j(e)}\})] \\ &= 2\mathrm{e} \cdot \mathrm{E}[f(S_r^A)], \end{aligned} \tag{7}$$

where the first inequality is just subadditivity, the second submodularity, the third follows from the fact that Dynkin's algorithm is an e-approximation for the secretary problem and selecting the element that Dynkin's selects with probability $1/2$ gives a $2e$ approximation, and the resulting telescoping sum gives the fourth equality. Now substituting (7) into (6) and rearranging, we get $\mathrm{E}[f(S_r^A)] \geq \frac{1}{1+2\mathrm{e}}\, f(S_r^A \cup C^*)$. An identical analysis of the second hypothetical algorithm gives: $\mathrm{E}[f(S_r^C)] \geq \frac{1}{1+2\mathrm{e}}\, f(S_r^C \cup C^* \setminus S_r^A)$.

It remains to analyze the case in which the algorithm runs in mode B. In this case, the algorithm generates a set S_r^B by selecting each element in S_r^A uniformly at random. By the theorem of [FMV07], uniform random sampling achieves a 4-approximation to the problem of *unconstrained* submodular maximization. Therefore, we have in this case: $\mathrm{E}[f(S_r^B)] \geq \frac{1}{4} f(S_r^A \cap C^*)$. By Lemma 2.2, we therefore have: $\mathrm{E}[f(S_r^A)] + \mathrm{E}[f(S_r^B)] + \mathrm{E}[f(S_r^C)] \geq \frac{1}{1+2e} f(C^*)$. Since our algorithm outputs one of these three sets uniformly at random, it gets a $(3 + 6e)$ approximation to $f(C^*)$. □

4.2.2 General Case

We now consider the general secretary setting, in which the elements come in random order, not necessarily grouped by partition. Our previous approach will not work: we cannot simply run Dynkin's secretary

algorithm on contiguous chunks of elements, because some elements may be blocked by our previous choices. We instead do something similar in spirit: we divide the elements up into k 'epochs', and attempt to select a single element from each. We treat every element that arrives before the current epoch as part of a sample, and according to the current valuation function at the beginning of an epoch, we select the first element that we encounter that has higher value than any element from its own partition group in the sample, so long as we have not already selected something from the same partition group. Our algorithm is as follows:

> Initially, the algorithm determines whether it is one of 3 different modes (A, B, or C) uniformly at random. The algorithm maintains a set of selected elements, initially S_0, and observes the first half of the elements without selecting anything. The algorithm then divides the next 1/20th of the elements into k *epochs*, contiguous blocks each containing a $1/20k$ fraction of the elements. At epoch i, we use valuation function $f_{S_{i-1}}$. When an element arrives, we ignore it with probability 99/100. Otherwise, if it has higher value than any element from its own partition group that arrived earlier than epoch i, we flip a coin. If we are in modes A or B, we let $S_i \leftarrow S_{i-1} \cup \{x\}$ if the coin is heads, and let $S_i \leftarrow S_{i-1}$ otherwise. If we are in mode C, we do the reverse, and let $S_i \leftarrow S_{i-1} \cup \{x\}$ if the coin is tails, and let $S_i \leftarrow S_{i-1}$ otherwise. After all k epochs have passed, we ignore the remaining 9/20ths of the elements. Finally, after the algorithm has completed, if we are in mode B, we discard each element of S_r with probability 1/2. (Note that we can actually implement this step online, by 'marking' but not selecting elements with probability 1/2 when they arrive).

If we were guaranteed to select an element in every epoch i that was the highest valued element according to $f_{S_{i-1}}$, then the analysis of this algorithm would be identical to the analysis in the contiguous case. This is of course not the case. However, we prove a technical lemma that says that we are "close enough" to this case.

Lemma 4.7. *For all partition groups i and epochs j, the algorithm selects the highest element from group i (according to the valuation function $f_{S_{j-1}}$ used during epoch j) during epoch j with probability at least $\Omega(\frac{1}{k})$.*

Because of space constraints, we defer the proof of this technical lemma to Appendix C.

Note an immediate consequence of the above lemma: if e is the element selected from epoch j, by summing over the elements in the optimal set C^* (1 from each of the k partition groups), we get:

$$\mathrm{E}[f_{S_{j-1}}(e)] \geq \Omega(\frac{1}{k}) \sum_{e' \in C^*} f_{S_{j-1}}(e') \geq \Omega(\frac{f_{S_{j-1}}(C^*)}{k})$$

Summing over the expected contribution to S_r from each of the k epochs and applying submodularity, we get $\mathrm{E}[f_{S_r^A}(C^*)] \leq O(\mathrm{E}[f(S_r^A)])$. Using this derivation in place of inequality 7 in the proof of lemma 4.6 proves that our algorithm gives an $O(1)$ approximation to the non-monotone submodular maximization problem subject to a partition matroid constraint.

4.3 Subject to a General Matroid Constraint

We consider matroid constraints where the matroid is $\mathcal{M} = (\Omega, \mathcal{I})$ with rank k. Let $w_1 = \max_{e \in \Omega} f(\{e\})$ the maximum value obtained by any single element, and let e_1 be the element that achieves this maximum value. (Note that we do not know these values up-front in the secretary setting.) In this section, we first give an algorithm that gets a set of fairly high value given a threshold τ. We then show how to choose this threshold, assuming we know the value w_1 of the most valuable element, and why this implies an advice-taking online algorithm having a logarithmic approximation. Finally, we show how to implement this in a secretary framework.

A Threshold Algorithm. Given a value τ, run the following algorithm. Initialize $S_1, S_2 \leftarrow \emptyset$. Go over the elements of the universe Ω in *arbitrary* order: when considering element e, add it to S_1 if $f_{S_1}(e) \geq \epsilon\tau$,

else add it to S_2 if $f_{S_2}(e) \geq \epsilon\tau$, else discard it. (We will choose the value of ϵ later.) Finally, output a uniformly random one of S_1 or S_2.

To analyze this algorithm, let C^* be the optimal set with $f(C^*) = \mathbf{OPT}$. Order the elements of C^* by picking its elements greedily based on marginal values. Given $\tau > 0$, let $C^*_\tau \subseteq C^*$ be the elements whose marginal benefit was at least τ when added in this greedy order: note that $f(C^*_\tau) \geq |C^*_\tau|\tau$.

Lemma 4.8. *For $\epsilon = 1/4$, the set produced has expected value is at least $|C^*_\tau| \cdot \tau/16$.*

Proof. If either $|S_1|$ or $|S_2|$ is at least $|C^*_\tau|/4$, we get value at least $|C^*_\tau|/4 \cdot \epsilon\tau$. Else both these sets have small cardinality. Since we are in a matroid, there must be a set $A \subseteq C^*_\tau$ of cardinality $|A| \geq |C^*_\tau| - |S_1| - |S_2| \geq |C^*_\tau|/2$, such that A is disjoint from both S_1 and S_2, and both $S_1 \cup A$ and $S_2 \cup A$ lie in $\mathcal{I}$ (i.e., they are independent).

We claim that $f(S_1) \geq f(S_1 \cup A) - |A| \cdot \epsilon\tau$. Indeed, an element in $e \in A$ was not added by the threshold algorithm; since it could be added while maintaining independence, it must have been discarded because the marginal value was less than $\epsilon\tau$. Hence $f_{S_1}(\{e\}) < \epsilon\tau$, and hence $f(S_1 \cup A) - f(S_1) = f_{S_1}(A) \leq \sum_{e \in A} f_{S_1}(\{e\}) < |A| \cdot \epsilon\tau$. Similarly, $f(S_2) \geq f(S_2 \cup A) - |A| \cdot \epsilon\tau$. And by disjointness, $f(S_1 \cap A) = f(\emptyset) = 0$. Hence, summing these and applying Lemma 2.2, we get that $f(S_1) + f(S_2) \geq f(S_1 \cup A) + f(S_2 \cup A) + f(S_1 \cap A) - 2\epsilon\tau|A| \geq f(A) - 2\epsilon\tau|A|$.

Since the marginal values of all the elements in C^*_τ were at least τ when they were added by the greedy ordering, and $A \subseteq C^*_\tau$, submodularity implies that $f(A) \geq |A|\tau$, which in turn implies $f(S_1) + f(S_2) \geq (1 - 2\epsilon)\tau|A| \geq (1 - 2\epsilon)\tau|C^*_\tau|/2$. A random one of S_1, S_2 gets half of that in expectation. Taking the minimum of $|C^*_\tau|/4 \cdot \epsilon\tau$ and $(1 - 2\epsilon)\tau|C^*_\tau|/2$ and setting $\epsilon = 2/5$, we get the claim. □

Lemma 4.9. $\sum_{i=0}^{\log 2k} |C^*_{w_1/2^i}| \cdot \frac{w_1}{2^i} \geq f(C^*)/4 = \mathbf{OPT}/4$.

Proof. Consider the greedy enumeration $\{e_1, e_2, \ldots, e_t\}$ of C, and let $w_j = f_{\{e_1, e_2, \ldots, e_{i-1}\}}(\{e_j\})$. First consider an infinte summation $\sum_{i=0}^{\infty} |C^*_{w_1/2^i}| \cdot \frac{w_1}{2^i}$—each element e_j contributes at least $w_j/2$ to it, and hence the summation is at least $\frac{1}{2}\sum_j w_j$. But $f(C^*) = \sum_{j=1}^{t} w_j$, which says the infinite sum is at least $f(C^*)/2 = \mathbf{OPT}/2$. But the finite sum merely drops a contribution of $w_1/4k$ from at most $|C^*| \leq k$ elements, and clearly $\mathbf{OPT}$ is at least w_1, so removing this contribution means the finite sum is at least $\mathbf{OPT}/4$. □

Hence, if we choose a value τ uniformly from $w_1, w_1/2, w_1/4, \ldots, w_1/2k$ and run the above threshold algorithm with that setting of τ, we get that the expected value of the set output by the algorithm is:

$$\frac{1}{1+\log 2k} \sum_{i=0}^{\log 2k} |C^*_{w_1/2^i}| \cdot \frac{w_1}{16 \cdot 2^i} \geq \frac{1}{1+\log 2k} \frac{\mathbf{OPT}}{64}. \tag{8}$$

The Secretary Algorithm. The secretary algorithm for general matroids is the following:

> Sample half the elements, let W be the weight of the highest weight element in the first half. Choose a value $i \in \{0, 1, \ldots, 2 + \log 2k\}$ uniformly at random. Let $S \leftarrow \emptyset$. On seeing an element e, pick it if $S + e$ is independent and $f_S(\{e\}) \geq W/2^i$. If picked, update $S \leftarrow S + e$.

Lemma 4.10. *The algorithm is an $O(\log k)$-approximation in the secretary setting for rank k matroids.*

Proof. With probability $\Theta(1/\log k)$, we choose the value $i = 0$. In this case, with constant probability the element with second-highest value comes in the first half, and the highest-value element e_1 comes in the second half; hence our (conditional) expected value in this case is at least w_1. In case this single element accounts for more than half of the optimal value, we get $\Omega(\mathbf{OPT}/\log k)$. We ignore the case $i = 1$. If we choose $i \geq 2$, now with constant probability e_1 comes in the first half, implying that $W = w_1$. Moreover, each element in $C - e_1$ appears in the second half with probability slightly higher than $1/2$. Since e_1 accounts for at most half the optimal value, the expected optimal value in the second half is at least $\mathbf{OPT}/4$. Finally, we pick an element if it has value at least $W/2^i = (1/4)w_1/2^J$, where $J := i - 2 \in_R \{0, 1, \ldots, \log_2 2k\}$, which the analysis following Lemma 4.9 assures us gives $\Omega(\mathbf{OPT}/\log k)$ in expectation. □

Acknowledgments. We thank V. Nagarajan, M.I. Sviridenko, J. Vondrák, and especially R.D. Kleinberg for valuable comments, suggestions, and conversations.

References

[BDG+09] Moshe Babaioff, Michael Dinitz, Anupam Gupta, Nicole Immorlica, and Kunal Talwar. Secretary problems: weights and discounts. In *19thProceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 1245–1254, 2009.

[BIK07] Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *SODA '07*, pages 434–443, 2007.

[BIKK07] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. A knapsack secretary problem with applications. In *APPROX '07*, 2007.

[CCPV09] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *to appear*, 2009.

[DP08] Nedialko B. Dimitrov and C. Greg Plaxton. Competitive weighted matching in transversal matroids. In *Automata, languages and programming. Part I*, volume 5125 of *Lecture Notes in Comput. Sci.*, pages 397–408, Berlin, 2008. Springer.

[Dyn63] E. B. Dynkin. Optimal choice of the stopping moment of a Markov process. *Dokl. Akad. Nauk SSSR*, 150:238–240, 1963.

[Fer89] T.S. Ferguson. Who solved the secretary problem? *Statistical Science*, 4:282–296, 1989.

[FMV07] U. Feige, V. Mirrokni, and J. Vondrak. Maximizing non-monotone submodular functions. In *Proceedings of 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2007.

[FNW78] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions. II. *Math. Programming Stud.*, (8):73–87, 1978. Polyhedral combinatorics.

[Fre83] P. R. Freeman. The secretary problem and its extensions: a review. *Internat. Statist. Rev.*, 51(2):189–206, 1983.

[HKJ80] D. Hausmann, B. Korte, and T. A. Jenkyns. Worst case analysis of greedy type algorithms for independence systems. *Math. Programming Stud.*, (12):120–131, 1980. Combinatorial optimization.

[HKP04] Mohammad Taghi Hajiaghayi, Robert Kleinberg, and David C. Parkes. Adaptive limited-supply online auctions. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 71–80, New York, NY, USA, 2004. ACM.

[Jen76] Thomas A. Jenkyns. The efficacy of the "greedy" algorithm. In *Proceedings of the Seventh Southeastern Conference on Combinatorics, Graph Theory, and Computing (Louisiana State Univ., Baton Rouge, La., 1976)*, pages 341–350. Congressus Numerantium, No. XVII, Winnipeg, Man., 1976. Utilitas Math.

[KH78] Bernhard Korte and Dirk Hausmann. An analysis of the greedy heuristic for independence systems. *Ann. Discrete Math.*, 2:65–74, 1978. Algorithmic aspects of combinatorics (Conf., Vancouver Island, B.C., 1976).

[Kle05] Robert Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *16th SODA*, pages 630–631. ACM, 2005.

[Kle09] Robert D. Kleinberg. A secretary problem with submodular payoff function. manuscript, 2009.

[KMN99] Samir Khuller, Anna Moss, and Joseph Naor. The budgeted maximum coverage problem. *Inform. Process. Lett.*, 70(1):39–45, 1999.

[KP09] Nitish Korula and Martin Pál. Algorithms for secretary problems on graphs and hypergraphs. pages 508–520, 2009.

[KST09] Ariel Kulik, Hadas Shachnai, and Tami Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *SODA '09: Proceedings of the Nineteenth Annual ACM -SIAM Symposium on Discrete Algorithms*, pages 545–554, 2009.

[LMNS09] J. Lee, V. Mirrokni, V. Nagarajan, and M. Sviridenko. Maximizing non-monotone submodular functions under matroid and knapsack constraints. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, 2009.

[LSV09] Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. In *Proceedings, International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 244–257, 2009.

[NWF78] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. I. *Math. Programming*, 14(3):265–294, 1978.

[Svi04] Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.*, 32(1):41–43, 2004.

[Von08] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings, ACM Symposium on Theory of Computing*, pages 67–74, 2008.

[Von09] Jan Vondrák. Symmetry and approximability of submodular maximization problems. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, page to appear, 2009.

[Wol82] Laurence A. Wolsey. Maximising real-valued submodular functions: primal and dual heuristics for location problems. *Math. Oper. Res.*, 7(3):410–425, 1982.

A Proof of Main Lemma for p-Systems

Let $e_1, e_2, \ldots, e_k$ be the elements added to S by greedy, and let S_i be the first i elements in this order, with $\delta_i = f_{S_{i-1}}(\{e_i\}) = f(S_i) - f(S_{i-1})$, which may be positive or negative. Since $f(\emptyset) = 0$, we have $f(S = S_k) = \sum_i \delta_i$. And since f is submodular, $\delta_i \geq \delta_{i+1}$ for all i. Let O be an optimal solution; i.e., $\mathbf{OPT} = f(O)$.

Lemma A.1 (following [CCPV09])**.** $f(S_k) \geq \frac{1}{p+1} f(O \cup S_k)$.

Proof. We show the existence of a partition of O into $O_1, O_2, \ldots, O_k$ with the following two properties:

- for all $i \in [k]$, $p_1 + p_2 + \ldots + p_i \leq i \cdot p$ where $p_i := |O_i|$, and
- for all $i \in [k]$, $p_i \delta_i \geq f_{S_k}(O_i)$.

Assuming such a partition, we can complete the proof thus:

$$p \sum_i \delta_i \geq \sum_i p_i \delta_i \geq \sum_i f_{S_k}(O_i) \geq f_{S_k}(O) = f(S_k \cup O) - f(S_k), \tag{9}$$

where the first inequality follows from [CCPV09, Claim A.1] (using the first property above, and that the δ's are non-increasing), the second from the second property of the partition of O, the third from subadditivity of $f_{S_k}(\cdot)$ (which is implied by the submodularity of f and applications of both facts in Proposition 1.1), and the fourth from the definition of $f_{S_k}(\cdot)$. Using the fact that $\sum_i \delta_i = f(S_k)$, and rearranging, we get the lemma.

Now to prove the existence of such a partition of O. Define $A_0, A_1, \ldots, A_k$ as follows: $A_i = \{e \in O \setminus S_i \mid S_i + e \in \mathcal{I}\}$. Note that since $O \in \mathcal{I}$, it follows that $A_0 = O$; since the independence system is closed under subsets, we have $A_i \subseteq A_{i-1}$; and since the greedy algorithm stops only when there are no more elements to add, we get $A_k = \emptyset$. Defining $O_i = A_{i-1} \setminus A_i$ ensures we have a partition $O_1, O_2, \ldots, O_k$ of O.

Fix a value i. We claim that S_i is a basis (a maximal independent set) for $S_i \cup (O_1 \cup O_2 \cup \ldots \cup O_i) = S_i \cup (O \setminus A_i)$. Clearly $S_i \in \mathcal{I}$ by construction; moreover, any $e \in (O \setminus A_i) \setminus S_i$ was not considered but not added to A_i because $S_i + e \notin \mathcal{I}$. Moreover, $(O_1 \cup O_2 \cup \ldots \cup O_i) \subseteq O$ is clearly independent by subset-closure. Since $\mathcal{I}$ is a p-independence system, $|O_1 \cup O_2 \cup \ldots \cup O_i| \leq p \cdot |S_i|$, and thus $\sum_i |O_i| = \sum_i p_i \leq i \cdot p$, proving the first property.

For the second property, note that $O_i = A_{i-1} \setminus A_i \subseteq A_{i-1}$; hence each $e \in O_i$ does not belong to S_{i-1} but could have been added to S_{i-1} whilst maintaining independence, and was considered by the greedy algorithm. Since greedy chose the e_i maximizing the "gain", $\delta_i \geq f_{S_{i-1}}(\{e\})$ for each $e \in O_i$. Summing over all $e \in O_i$, we get $p_i \delta_i \leq \sum_{e \in O_i} f_{S_{i-1}}(\{e\}) \leq f_{S_{i-1}}(O_i)$, where the last inequality is by the subadditivity of $f_{S_{i-1}}$. Again, by submodularity, $f_{S_{i-1}}(O_i) \leq f_{S_k}(O_i)$, which proves the second fact about the partition $\{O_j\}_{j=1}^k$ of O. □

Clearly, the greedy algorithm works no worse if we stop it when the best "gain" is negative, but the above proof does not use that fact.

B Proofs for Knapsack Constraints

The proof is virtually identical to that in [Svi04] except we avoid any uses of monotonicity; we give the complete proof here only for completeness. We use the notation similar to his paper for consistency. Let f be a non-negative submodular function with $f(\emptyset) = 0$. Let $I = [n]$, and we are given n items with weights $c_i \in \mathbb{Z}_+$, and $B \geq 0$; let $\mathcal{F} = \{S \subseteq I \mid c(S) \leq B$, where $c(S) = \sum_{i \in S} c_i$. Our goal to solve $\max_{S \subseteq \mathcal{F}} f(S)$. To that end, we want to prove the following result:

Theorem B.1. *There is a polynomial-time algorithm that outputs a collection of sets such that for any $C \in \mathcal{F}$, the collection contains a set S satisfying $f(S) \geq (1 - 1/\mathrm{e}) f(S \cup C)$.*

B.1 The Algorithm

The algorithm is the following: it constructs a polynomial number of solutions and chooses the best among them (and in case of ties, outputs the lexicographically smallest one of them).

- First, the family contains all solutions with cardinality $1, 2, 3$: clearly, if $|C| \leq 3$ then we will output C itself, which will satisfy the condition of the theorem.
- Now for each solution $U \subseteq I$ of cardinality 3, we greedily extend it as follows: Set $S_0 = U$, $I_0 = I$. At step t, we have a partial solution S_{t-1}. Now compute

$$\theta_t = \max_{i \in I_{t-1} \setminus S_{t-1}} \frac{f(S_{t-1} + i) - f(S_{t-1})}{c_i}. \tag{10}$$

  Let the maximum be achieved on index i_t. If $\theta_t \leq 0$, terminate the algorithm. Else check if $c(S_{t-1} + i_t) \leq B$: if so, set $S_t = S_{t-1} + i_t$ and $I_t = I_{t-1}$, else set $S_t = S_{t-1}$ and $I_t = I_{t-1} - i_t$. Stop if $I_t \setminus S_t = \emptyset$.

The family of sets we output is all sets of cardinality at most three, as well as for each greedy extension of a set of cardinality three, we output all the sets S_t created during the run of the algorithm. Since each set can have at most n elements, we get $O(n^4)$ sets output by the algorithm.

B.2 The Analysis

Let us assume that $|C| = t > 3$, and order C as $j_1, j_2, \ldots, j_t$ such that

$$j_k = \max_{j \in C \setminus \{j_1, \ldots, j_{k-1}\}} f_{\{j_1, \ldots, j_{k-1}\}}(\{j\}), \tag{11}$$

i.e., index the elements in the order they would be considered by the greedy algorithm that picks items of maximum marginal value (and does not consider their weights c_i). Let $Y = \{j_1, j_2, j_3\}$. Submodularity and the ordering of C gives us the following:

Lemma B.2. *For any $j_k \in C$ with $k \geq 4$ and any $Z \subseteq I \setminus \{j_1, j_2, j_3, j_k\}$, it holds that:*

$$\begin{aligned}
&f_{Y \cup Z}(\{j_k\}) \leq f(\{j_k\}) \leq f(\{j_1\}) \\
&f_{Y \cup Z}(\{j_k\}) \leq f(\{j_1, j_k\}) - f(\{j_1\}) \leq f(\{j_1, j_2\}) - f(\{j_1\}) \\
&f_{Y \cup Z}(\{j_k\}) \leq f(\{j_1, j_2, j_k\}) - f(\{j_1, j_2\}) \leq f(\{j_1, j_2, j_3\}) - f(\{j_1, j_2\})
\end{aligned}$$

Summing the above three inequalities we get that for $j_k \notin Y \cup Z$,

$$3\, f_{Y \cup Z}(\{j_k\}) \leq f(Y). \tag{12}$$

For the rest of the discussion, consider the iteration of the algorithm which starts with $S_0 = Y$. For S such that $S_0 = Y \subseteq S \subseteq I$, recall that $f_Y(S) = f(Y \cup S) - f(Y) = f(S) - f(Y)$. Proposition 1.1 shows that $f_Y(\cdot)$ is a submodular function with $f_Y(\emptyset) = 0$. The following lemma is the analog of [Svi04, eq. 2]:

Lemma B.3. *For any submodular function g and all $S, T \subseteq I$ it holds that*

$$g(T \cup S) \leq g(S) + \sum_{i \in T \setminus S} (g(S + i) - g(S)) \tag{13}$$

Proof. $g(T \cup S) = g(S) + (g(T \cup S) - g(S)) = g(S) + g_S(T \setminus S) \leq g(S) + \sum_{i \in T \setminus S} g_S(\{i\}) = g(S) + \sum_{i \in T \setminus S} (g(S + i) - g(S))$, where we used subadditivity of the submodular function g_S. □

Let $\tau + 1$ be the first step in the greedy algorithm at which either (a) the algorithm stops because $\theta_{\tau+1} \leq 0$, or (b) we consider some element $i_{\tau+1} \in C$ and it is dropped by the greedy algorithm—i.e., we set $S_{\tau+1} = S_\tau$ and $I_{\tau+1} = I_\tau - i_{\tau+1}$. Note that before this point either we considered elements from C and picked them, or the element considered was not in C. In fact, let us assume that there are no elements that are neither in C nor are picked by our algorithm, since we can drop them and perform the same algorithm and analysis again, it will not change anything—hence we can assume we have not dropped any elements before this, and $S_t = \{i_1, i_2, \ldots, i_t\}$ for all $t \in \{0, 1, \ldots, \tau\}$.

Now we apply Lemma B.3 to the submodular function $f_Y(\cdot)$ with sets $S = S_t$ and $T = C$ to get

$$f_Y(C \cup S_t) \leq f_Y(S_t) + \sum_{i \in C \setminus S_t} f_Y(S_t + i) - f_Y(S_t) = f_Y(S_t) + \sum_{i \in C \setminus S_t} f(S_t + i) - f(S_t) \tag{14}$$

Suppose case (a) happened and we stopped because $\theta_{\tau+1} \leq 0$. This means that every term in the summation in (14) must be negative, and hence $f_Y(C \cup S_\tau) \leq f_Y(S_\tau)$, or equivalently, $f(C \cup S_\tau) \leq f(S_\tau)$. In this case, we are not even losing the $(1 - 1/\mathrm{e})$ factor.

Case (b) is if the greedy algorithm drops the element $i_{\tau+1} \in C$. Since $i_{\tau+1}$ was dropped, it must be the case that $c(S_\tau) \leq B$ but $c(S_\tau + i_{\tau+1}) = B' > B$. In this case the right-hand expression in (14) has some positive terms for each of the values of $t \leq \tau$, and hence for each t, we get

$$f_Y(C \cup S_t) \leq f_Y(S_t) + B \cdot \theta_{t+1}. \tag{15}$$

We now make a technical lemma; the proof consists of some algebraic violence, and the casual reader can skip to the remainder of the proof of the main theorem.

Lemma B.4. $\frac{f_Y(S_\tau + i_{\tau+1})}{f_Y(S_\tau \cup C)} \geq (1 - 1/\mathrm{e})$.

Proof. For $t \leq \tau$, define $B_t = \sum_{t' \leq t} c_{i_{t'}} = c(S_t \setminus S_0)$, and let $B_{\tau+1} = B' = B_\tau + c_{\tau+1}$. Note that $B_\tau \leq B < B'$.

For all integers $t \leq \tau + 1$ and $j \in \{B_{t-1} + 1, \ldots, B_t\}$, define $\rho_j = \theta_t$. Also,

$$f_Y(S_\tau + i_{\tau+1}) = \textstyle\sum_{t' \leq \tau+1} c_{i_{t'}} \theta_{t'} = \sum_{j=1}^{B'} \rho_j. \tag{16}$$

In fact, for all $t \leq \tau$, it holds that

$$f_Y(S_t) = \textstyle\sum_{t' \leq \tau+1} c_{i_{t'}} \theta_{t'} = \sum_{j=1}^{B_t} \rho_j. \tag{17}$$

By the greedy algorithm, the ρ's are non-increasing. Hence

$$\begin{aligned} \min_{s \in 1 \ldots B'} \{\textstyle\sum_{i=1}^{s-1} \rho_j + B \cdot \rho_s\} &= \min_{t \in 1 \ldots \tau} \{\textstyle\sum_{i=1}^{B_t} \rho_j + B \cdot \rho_{B_t+1}\} \\ &= \min_{t \in 1 \ldots \tau} \{f_Y(S_t) + B \cdot \theta_{t+1}\} \\ &\geq f_Y(C \cup S_t) \qquad \text{(by (15))} \end{aligned}$$

Hence, from the above calculations and from (16)

$$\begin{aligned} \frac{f_Y(S_\tau + i_{\tau+1})}{f_Y(S_\tau \cup C)} &\geq \frac{\sum_{j=1}^{B'} \rho_j}{\min_{s \in 1 \ldots B'} \{\sum_{i=1}^{s-1} \rho_j + B \cdot \rho_s\}} \\ &\geq (1 - \mathrm{e}^{-B'/B}) \geq (1 - 1/\mathrm{e}). \end{aligned}$$

where the inequalities on the second line follow from [Svi04, eq. 2]. □

Now for the final calculations:

$$\begin{aligned}
f(S_\tau) &\geq f(Y) + f_Y(S_\tau) \\
&\geq f(Y) + f_Y(S_\tau + i_{\tau+1}) - \big(f_Y(S_\tau + i_{\tau+1}) - f_Y(S_\tau)\big) \\
&\geq f(Y) + f_Y(S_\tau + i_{\tau+1}) - \big(f(S_\tau + i_{\tau+1}) - f(S_\tau)\big) \\
&\geq f(Y) + (1 - 1/\mathrm{e}) f_Y(S_\tau \cup C) - f(Y)/3 && \text{(using Lemma B.4 and (12))} \\
&\geq (1 - 1/\mathrm{e}) f(S_\tau \cup C). && \text{(by the definition of } f_Y() \text{ and } \mathrm{e} \leq 3)
\end{aligned}$$

Hence this set S_τ will be in the family of sets output, and will satisfy the claim of the theorem.

C Proofs from the Submodular Secretaries Section

In this section, we give the missing proofs from Section 4.

C.1 Proof for Cardinality Constrained Submodular Secretaries

Theorem 4.5 *The algorithm for the cardinality-constrained submodular maximization problem in the secretary setting gives an $O(1)$ approximation to* **OPT**.

The proof basically shows that with reasonable probability, both the first and the second half of the stream have a reasonable fraction of **OPT**, so when we run the offline algorithm on the first half, using its output to extract value from the second half gives us a constant fraction of **OPT**.

Proof. Let $C^* = \{e_1, \ldots, e_{k'}\}$ denote some set with $k' \leq k$ elements such that $f(C^*) = \mathbf{OPT}$. Without loss of generality, we normalize so that $\mathbf{OPT} = 1$. Suppose the elements of C^* have been listed in the "greedy order" (i.e., in order of decreasing marginal utility), and let a_i denote the marginal utility of e_i when it is added to $\{e_1, e_2, \ldots, e_{i-1}\}$. We consider two cases: in the first case, $a_1 \geq 1/c$, where $c \geq 1$ is some constant to be determined. In this case, with probability $1/2e$, the algorithm runs Dynkin's secretary algorithm and selects a_1, achieving an $1/(2ce)$ approximation.

In the other case, $a_i < 1/c$ for all i. We imagine randomly partitioning the elements of the input set X into two sets, X_1 and X_2, with each element belonging to X_1 independently with probability $1/2$. This corresponds to the algorithm's division of $\boldsymbol{\sigma}$ into the first (random) m elements $\boldsymbol{\sigma}_m$ and the remaining elements $\boldsymbol{\sigma} - \boldsymbol{\sigma}_m$. Let C_1^* and C_2^* denote the optimal solutions restricted to sets X_1 and X_2 respectively. Define the random variable $A = \sum_{i=1}^{k'} Y_i a_i$ where each $Y_i \in_r \{-1, 1\}$ is selected uniformly at random. Note that by submodularity, $f(C_1^*) + f(C_2^*) \geq f(C^*) = 1$. We wish to lower bound $\min(f(C_1^*), f(C_2^*))$, and to do this it is sufficient to upper bound the absolute value $|A|$. To see this, suppose that, for some setting of the Y_i's it holds that $\sum_{i:Y_i=1} a_i \geq \sum_{i:Y_i=-1} a_i$ (the other case is identical). Now if $|A| = \sum_{i:Y_i=1} a_i - \sum_{i:Y_i=-1} a_i \leq x$, we have:

$$\sum_{i:Y_i=-1} a_i \geq (\sum_{i:Y_i=1} a_i) - x = 1 - (\sum_{i:Y_i=-1} a_i) - x$$

and hence

$$\min(f(C_1^*), f(C_2^*)) \geq \sum_{i:Y_i=-1} a_i \geq \frac{1-x}{2}.$$

Hence, we would like to upper bound $|A|$ with high probability. Since each Y_i is independent with expectation 0, we have $\mathrm{E}[A] = 0$ and $\mathrm{E}[A^2] = \sum_{i=1}^{k'} a_i^2$. The standard deviation of A is:

$$\sigma = \sqrt{\sum_{i=1}^{k'} a_i^2} \leq \sqrt{c \cdot \frac{1}{c}^2} = \frac{1}{\sqrt{c}}.$$

By Chebyshev's inequality, for any $d \geq 0$, we have

$$\Pr[|A| \geq \frac{d}{\sqrt{c}}] \leq \frac{1}{d^2}.$$

That is, except with probability $1/d^2$, $\min(f(C_1^*), f(C_2^*)) \geq (1 - \frac{d}{\sqrt{c}})/2$. Now for some calculations. With probability $1/2$, we do not run Dynkin's algorithm. Independently of this, with probability $1/2$, $f(C_1^*) \leq f(C_2^*)$—i.e., the value $\min(f(C_1^*), f(C_2^*))$ is achieved on $\boldsymbol{\sigma}_m$. With probability $(1 - 1/d^2)$, this value is at least $(1 - \frac{d}{\sqrt{c}})/2$. Now we run a ρ_{off}-approximation on $\boldsymbol{\sigma}_m$, and thus with probability $\frac{1}{4}(1 - 1/d^2)$,

$$f(A_1) \geq \frac{1}{2}(1 - \frac{d}{\sqrt{c}}) \cdot \frac{1}{\rho_{\text{off}}}.$$

If we use this as a lower bound for $f(C_2^*)$ (which is fine since we are in the case where $f(A_1) \leq f(C_1^*) \leq f(C_2^*)$), the semi-online algorithm gives us a value of at least $\frac{f(A_1)}{\rho_{on}}$. Hence we have

$$\mathrm{E}[f(A_2)] \geq \frac{1}{2}(1 - \frac{d}{\sqrt{c}}) \cdot \frac{1}{\rho_{\text{off}}} \cdot \frac{1}{4}(1 - 1/d^2) \cdot \frac{1}{\rho_{\text{on}}}. \tag{18}$$

Combining both cases and optimizing over parameters d and c ($d \leftarrow 3.08,\ c \leftarrow 260.24$) we have:

$$\mathrm{E}[f(A_2)] \geq \min\left(\frac{1}{8\,\rho_{\text{off}}\,\rho_{\text{on}}}(1 - 1/d^2)(1 - \frac{d}{\sqrt{c}}), \frac{1}{2ce}\right) \cdot \mathbf{OPT} \geq \frac{\mathbf{OPT}}{1417} \tag{19}$$

□

C.2 Proof for Partition Matroid Submodular Secretaries

For the following lemma, we need to keep track of several events:

1. $H_{i,j}$: The highest element from partition group i defined under the valuation function used during epoch j falls into epoch j.
2. $F_{i,j}$: The highest element from partition group i among those seen until the end of epoch j (defined under the valuation function used during epoch j) falls into epoch j.
3. $L_{i,j}$: The highest element from partition group i defined under the valuation function used during epoch j does not fall before epoch j.
4. $S_{i,j}$: The second highest element from partition group i defined under the valuation function used during epoch j falls before epoch j.
5. $P_{i,j}$: Some element from partition group i has already been selected before epoch j.

We write $q = \frac{1}{100} \cdot \frac{1}{2} = \frac{1}{200}$ to be the probability that the algorithm selects a candidate element.

Lemma 4.7 *For all partition groups i and epochs j, the algorithm selects the highest element from group i (according to the valuation function during epoch j) during epoch j with probability at least $\Omega(\frac{1}{k})$. Specifically:*

$$\Pr[(H_{i,j} \wedge S_{i,j} \wedge \neg P_{i,j}) \wedge (\bigwedge_{i' \neq i} \neg F_{i',j})] = \Omega(\frac{1}{k})$$

where the probability is over the random permutation of the elements.

Proof. Under any (arbitrary) valuation function, $\Pr[L_{i,j} \wedge S_{i,j}] \geq \frac{1}{5}$, since at least 1/2 of the elements fall before any epoch, and 9/20ths fall after any epoch. Similarly, $\Pr[F_{i,j}] \leq \frac{1/20k}{1/2+1/20k} \leq \frac{1}{10k}$, since at least 1/2 of the elements fall before any epoch. Note that $\Pr[P_{i,j}] \leq \sum_{j' \leq j} \Pr[F_{i,j}] \leq \frac{j-1}{10k} < \frac{1}{10}$. Combining these expressions, we get: $\Pr[L_{i,j} \wedge S_{i,j} | \neg P_{i,j}] \geq \Pr[L_{i,j} \wedge S_{i,j}] - \Pr[P_{i,j}] \geq \frac{1}{10}$.

For convenience, let us define event $\mathcal{E}_{i,j} = (L_{i,j} \wedge S_{i,j} \wedge \neg P_{i,j})$. We have:

$$\Pr[\mathcal{E}_{i,j}] = \Pr[L_{i,j} \wedge S_{i,j} | \neg P_{i,j}] \Pr[\neg P_{i,j}] \geq \frac{1}{10}(1 - \frac{1}{10}) = \frac{9}{100}$$

We now wish to consider the probability that groups $i' \neq i$ have elements in epoch j that the algorithm might select, conditioned on $\mathcal{E}_{i,j}$. Note that the algorithm selects a proposed element with probability q: we may however equivalently imagine that each proposed element is simply removed from the permutation with probability $(1-q)$ before the algorithm encounters it. So:

$$\Pr[F_{i',j} | \mathcal{E}_{i,j}] \leq qE[Q_{i',j} | \mathcal{E}_{i,j}] \cdot \frac{1/20k}{2/5} = qE[Q_{i',j} | \mathcal{E}_{i,j}] \frac{1}{8k}$$

where $Q_{i',j}$ represents the number of elements from group i' that appear in epoch j and are higher (according to the valuation function at epoch j) than any element seen before epoch j. But

$$E[Q_{i',j} | \mathcal{E}_{i,j}] \leq \frac{E[Q_{i',j}]}{\Pr[\mathcal{E}_{i,j}]} \leq \frac{E[\text{Geom}(1/2)]}{\Pr[\mathcal{E}_{i,j}]} \leq 2 \cdot \frac{100}{9} = \frac{200}{9}$$

where Geom(1/2) represents the geometric distribution with success probability 1/2. Plugging this in above, we find $\Pr[F_{i',j} | \mathcal{E}_{i,j}] \leq \frac{200q}{72k} = \frac{1}{72k}$, taking $q = \frac{1}{200}$. Since there are k groups i', by a union bound: $\Pr[\bigvee_{i' \neq i} F_{i',j} | \mathcal{E}_{i,j}] \leq \frac{1}{72}$, and so: $\Pr[\bigwedge_{i' \neq i} \neg F_{i',j} | \mathcal{E}_{i,j}] \geq \frac{71}{72}$. Consequently:

$$\Pr[\mathcal{E}_{i,j} \wedge (\bigwedge_{i' \neq i} \neg F_{i',j})] \geq \frac{71}{72} \Pr[\mathcal{E}_{i,j}] \geq \frac{71}{72} \cdot \frac{9}{100} = \frac{71}{800}$$

To complete the proof, we observe $\Pr[H_{i,j} | \mathcal{E}_{i,j} \wedge (\bigwedge_{i' \neq i} \neg F_{i',j})] \geq \frac{1/20k}{2/5} = \frac{1}{8k}$ and so:

$$\Pr[H_{i,j} \wedge \mathcal{E}_{i,j} \wedge (\bigwedge_{i' \neq i} \neg F_{i',j})] \geq \frac{71}{800} \cdot \frac{1}{8k} = \frac{71}{6400k}$$

Finally, we select this highest element when we encounter it with probability $q = 1/200$, and so we select the highest element from group i in epoch j with probability at least $\frac{71}{1280000k}$, which completes the proof. □

D Lower Bounds for the Constrained Submodular Maximization Problem in the Secretary Setting

In this section we show lower-bounds for the secretary problem over submodular functions. We first note that Kleinberg [Kle05] showed that for additive functions, the maximization problem in the on-line setting with a k-uniform matroid constraint can be approximated within a factor of $1 - \frac{5}{\sqrt{k}}$. We show that this is not the case for submodular functions, even in the information theoretic, semi-online setting by exhibiting a gap for arbitrarily large k.

Theorem D.1. *No algorithm approximates submodular maximization in the semi-online setting with a k-uniform matroid constraint better than a factor of $\frac{8}{9}$ for $k = 2$ or $\frac{35}{36}$ for any even k.*

We note that no non-trivial bound is possible for $k = 1$ because the algorithm knows OPT. Thus the standard secretary lower bounds will not work.

We first prove the case for $k = 2$ with a small example and case analysis. Let $b \in \{1, 2\}$. We define the set cover problem instance COVER(b) as follows: Let the base elements be $\{1, 2\} \times \{\flat, \sharp\}$. Let the sets be $\{(1, \flat)\}$, $\{(2, \flat)\}$, and $\{(b, \flat), (b, \sharp)\}$.

We consider the semi-online maximization problem where b is random, and the arriving elements are the sets of the COVER(b) instance. The objective function f is the number of base elements covered. The constraint is that only 2 sets can be selected.

Claim D.2. No algorithm approximates the above COVER(b) submodular maximization in the semi-online setting better than $\frac{8}{9}$.

Proof. We proceed by case analysis.

In the case where the first element that arrives is $\{(b, \flat), (b, \sharp)\}$, the algorithm knows what b is and can obtain $OPT = 3$. This happens with probability $\frac{1}{3}$.

In the case where the first element that arrives is $\{(b', \flat)\}$, the algorithm can accept or reject the element. If the algorithm rejects, then he may as well take the next two elements that arrive. In this case his expected payoff is $\frac{5}{2}$ because there is a fifty/fifty chance that $b' \neq b$ and so rejecting the first element was a good idea. Thus the total payoff is $\frac{2}{3}\frac{5}{2} + \frac{1}{3}3 = \frac{8}{3}$

If the algorithm accepts, then if the next element that arrives is also of the form $\{(b'', \flat)\}$ the algorithm should reject it. For, if it takes it, it will get payoff 2, but it can always get that payoff by accepting the third set. By rejecting it and instead taking the third set, then the algorithm will get payoff 2 with probability 1/2 (if $b' = b$) and 3 with probability 1/2 (if $b' \neq b$) for an expected payment of $\frac{5}{2}$. If the next element is of the form $\{(b, \flat), (b, \sharp)\}$ then the algorithm knows what b is. If $b' = b$, then taking it or not, the algorithm can receive 2. If $b' \neq b$ then the algorithm receives 3 by taking it. Each of these events happens with probability 1/2. Thus the total expected payoff is the same as in the above, rejecting case, of $\frac{8}{3}$.

$OPT = 3$, and thus no algorithm can do better than $\frac{8}{9}$ fraction of OPT.

□

Let k be an even integer, let $T \subseteq [k]$ (recall that $[k]$ denotes the set $\{1, \ldots, k\}$) such that $|T| = k/2$. We define the COVER(k, T) as follows: Let the base elements be $[k] \times \{\flat, \sharp\}$. Let the sets be $\{(i, \flat)\}$ for $i \in [k]$ and $\{(i, \flat), (i, \sharp)\}$ for $i \in T$.

We define *classes* by saying that two elements are in the same class if their first coordinates are equal. It will be helpful to think of a COVER(k, T) as being generated as follows: take $k/2$ instances of COVER(b) and jumble them together by replacing the $\{1, 2\}$ indices with an element of $[k]$. This give us a natural pairing of classes into $k/2$ *puzzles*.

Note that $OPT = \frac{3k}{2}$. Fix an semi-online algorithm A. Let S be the set of elements chosen by A. Let P_0, P_1, P_2, P_3 be the set of puzzles such that S contains respectively, 0, 1, 2, and 3 elements from the puzzle. Let x_0, x_1, x_2, x_3 be the expected sizes of P_0, P_1, P_2, P_3 (over the randomness of the assignments of puzzles, the ordering, and A).

Claim D.3. The puzzles in P_0, P_1, P_2 contribute at most $\frac{4k}{3}$ to the objective.

Proof. Say, for the sake of contradiction, that these is an algorithm A such that the expected utility from the puzzles with exactly two elements chosen from them was more than $\frac{4}{3}k$. We will show a reduction. We are given an instance of COVER(b) that we would like to solve. We create a instance of COVER($k - 2, T$). Then we select 3 positions at random. At each of these positions, we will insert one element of the given COVER(b) instance. Randomly relabel the classes. Note that the distribution we have created is identical to a COVER(k, T) instance. We run A on this instance, and accept the same sets (of the three designated sets) that A accepts. The expected contribution of puzzles with at most two elements selected is more than $\frac{4k}{3}$. Thus on average each such puzzle contributes more than $\frac{8}{3}$, and so in expectation, this reduction yields a payoff of more than $\frac{8}{3}$ on the given instance of COVER(b), which is a contradiction. □

The puzzles in P_3 contribute $3x_3$ to the objectively function (in expectation). Let S be the set chosen by A. By the claim we see that $\mathrm{E}[f(S)] \leq \frac{4}{3}k + 3x_3$.

Note also that $\mathrm{E}[f(S)] \leq OPT - 3x_0 - x_1$ because if the algorithm fails to pick two element from a given puzzle, it has missed at least 1 from **OPT**; and if it failed to pick even 1 element, it has missed 3.

Finally, because $x_3 \leq 2x_0 + x_1$ because of the k uniform constraint. These three equations imply that $\mathrm{E}[f(S)] \leq \frac{35}{24}k$. Thus we get a bound of $\frac{\mathrm{E}[f(S)]}{\mathbf{OPT}} \leq \frac{35}{36}$.