

# Deterministic Algorithms for Submodular Maximization Problems

Niv Buchbinder\*

Moran Feldman<sup>†</sup>

August 11, 2015

## Abstract

Randomization is a fundamental tool used in many theoretical and practical areas of computer science. We study here the role of randomization in the area of submodular function maximization. In this area most algorithms are randomized, and in almost all cases the approximation ratios obtained by current randomized algorithms are superior to the best results obtained by known deterministic algorithms. Derandomization of algorithms for general submodular function maximization seems hard since the access to the function is done via a value oracle. This makes it hard, for example, to apply standard derandomization techniques such as conditional expectations. Therefore, an interesting fundamental problem in this area is whether randomization is inherently necessary for obtaining good approximation ratios.

In this work we give evidence that randomization is not necessary for obtaining good algorithms by presenting a new technique for derandomization of algorithms for submodular function maximization. Our high level idea is to maintain explicitly a (small) distribution over the states of the algorithm, and carefully update it using marginal values obtained from an extreme point solution of a suitable linear formulation. We demonstrate our technique on two recent algorithms for unconstrained submodular maximization and for maximizing submodular function subject to a cardinality constraint. In particular, for unconstrained submodular maximization we obtain an optimal deterministic  $1/2$ -approximation showing that randomization is unnecessary for obtaining optimal results for this setting.

---

\*Department of Statistics and Operations Research, Tel Aviv University, Israel. e-mail: niv.buchbinder@gmail.com

<sup>†</sup>School of Computer and Communication Sciences, EPFL, Switzerland. e-mail: moran.feldman@epfl.ch.

# 1 Introduction

Randomization is a fundamental tool used in many theoretical and practical areas of computer science [2, 32]. It is widely used, for example, in cryptology, coding, and the design of approximation and online algorithms. In the area of approximation algorithms randomization is used both to improve the running time of algorithms and to obtain improved approximation ratios. However, in the majority of cases randomization can be removed to obtain an equivalent deterministic algorithm with the same approximation ratio (albeit, in some cases, with a longer running time). One central field of combinatorial optimization in which it is unclear whether it is possible to avoid randomization is the field of submodular function maximization.

A set function  $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}$  is submodular if for every  $A, B \in \mathcal{N}$ :  $f(A \cap B) + f(A \cup B) \leq f(A) + f(B)$ . An equivalent definition of submodularity, which is perhaps more intuitive, is that of diminishing returns:  $f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B)$  for every  $A \subseteq B \subseteq \mathcal{N}$  and  $u \notin B$ . The concept of diminishing returns is widely used in economics, and thus, it should come as no surprise that utility functions in economics are often submodular. Additionally, submodular functions are ubiquitous in various other disciplines, including combinatorics, optimization, information theory, operations research, algorithmic game theory and machine learning. A few well known examples of submodular functions from these disciplines include cut functions of graphs and hypergraphs, rank functions of matroids, entropy, mutual information, coverage functions and budget additive functions. Moreover, submodular maximization problems capture well known combinatorial optimization problems such as: Max-Cut [24, 26, 28, 29, 33], Max-DiCut [15, 24, 25], Generalized Assignment [10, 11, 17, 22] and Max-Facility-Location [1, 12, 13].

For a general submodular function the explicit representation of the function might be exponential in the size of its ground set. Hence, it is standard practice to assume that the function is accessed via a value oracle returning the value of  $f(S)$  when given a set  $S \subseteq \mathcal{N}$ . The use of a value oracle makes it difficult to use the standard derandomization method of conditional expectations (see, *e.g.*, [2]). Moreover, other standard methods, such as using a small sample space, seem to fail as well. Indeed, for most scenarios the best current approximation algorithms are randomized, while known deterministic algorithms achieve only inferior approximation ratios. One example is the problem of unconstrained submodular maximization [4] for which the best (optimal) randomized algorithm has an approximation ratio of  $1/2$ , while the approximation ratio of the best known deterministic algorithm is only  $1/3$ . Another example is maximizing a monotone submodular function subject to a matroid constraints [21, 8, 19]. For this problem the best deterministic algorithm has an approximation ratio of  $1/2$ , while the best (optimal) randomized algorithm has an improved approximation ratio of  $1 - 1/e$ . An interesting fundamental problem is whether randomization is inherently necessary for obtaining good approximation ratio in the field of submodular function maximization.

## 1.1 Our results

In this paper we give evidence that randomization is not necessary for obtaining good algorithms for submodular maximization. We present a new technique for derandomization of algorithms for submodular maximization based on the following idea. In a typical randomized algorithm the size of the support of the distribution implied by the algorithm is exponential (as otherwise it can often be trivially enumerated). We show that in certain cases the size of the distribution can be kept small (polynomial) throughout the execution of the algorithm. This is done by formulating the set of “good” updates to the distribution in each iteration of the algorithm as a linear formulation, and then choosing an update step that corresponds on an (optimal) **extreme** point of the linear

program. We then use the fact that an extreme point for our formulations does not have many non-integral variables to control the increase in the size of the support of the distribution. This allows us to maintain the distribution explicitly throughout the execution. We are not aware of any previous results that obtain derandomization via such an approach, and believe that this idea may be applicable for other settings as well.

We demonstrate our technique on two recent algorithms. The first of them is a randomized  $1/2$ -approximation algorithm presented by [4] for the problem of unconstrained submodular maximization. It is known that the approximation ratio of the last algorithm cannot be improved by any polynomial time algorithm [16]. Using our technique we obtain the following result. In this result, and throughout the rest of the paper, we use  $n$  to denote the size of the ground set  $\mathcal{N}$ .

**Theorem 1.1.** *Let  $f$  be a submodular function. There exists a **deterministic** algorithm with an approximation ratio of  $1/2$  for the problem  $\max_{S \subseteq \mathcal{N}} \{f(S)\}$ . The algorithm makes  $O(n^2)$  value oracle queries.*

Notice that this result shows that randomization is not necessary for obtaining the best possible approximation ratio for the problem. However, this comes at a cost, as the number of oracle queries made by our deterministic algorithm is  $O(n^2)$ , while the randomized algorithm only needs  $O(n)$  oracle queries.

Our second result is for maximizing a (non-monotone) submodular function subject to a cardinality constraint. For this problem the best known randomized algorithm has an approximation guarantee of  $1/e + 0.004$  [5]. This algorithm is based on a combination of two randomized algorithms; one of which is an elegant randomized greedy algorithm that has approximation ratio  $1/e$  (on its own). We show that one can obtain an equivalent deterministic algorithm.

**Theorem 1.2.** *Let  $f$  be a submodular function. There exists a **deterministic** algorithm that has an approximation ratio of  $1/e$  for the problem of  $\max_{|S| \leq k} \{f(S)\}$ . The algorithm makes  $O(k^2n)$  value oracle queries.*

Again, note that our deterministic algorithm makes  $O(k^2n)$  oracle queries, while the randomized algorithm only needs  $O(kn)$  queries.

## 1.2 Previous Results

The literature on submodular maximization problems is very large, and therefore, we mention below only a few of the most relevant works. Randomization is widely used in submodular maximization. In particular, many of the recent algorithms use an extension of submodular functions to fractional vectors known as the *multilinear* extension (see, e.g., [8, 23, 19, 30, 5]). Any algorithm using this extension must be randomized since the only known way to (approximately) evaluate this extension is via random sampling. A few examples of randomized algorithms for submodular maximization that do not use the multilinear extension can be found in [14, 3, 4, 20].

The first provable approximation algorithms for unconstrained submodular maximization were described by Feige et al. [16]. Their best algorithms for the problem achieve a randomized approximation ratio of  $0.4 - o(1)$ , and a deterministic approximation ratio of  $1/3 - \varepsilon$  (where  $\varepsilon$  is an arbitrarily small positive constant). On the negative side, [16] showed that no algorithm has an approximation ratio of  $1/2 + \varepsilon$  for the problem. The randomized approximation ratio was improved gradually [23, 18], eventually leading to an optimal linear time  $1/2$ -approximation randomized algorithm given by [4]. However, the deterministic approximation ratio has not been improved since the work of [16]. Interestingly, [4] described a different  $1/3$ -approximation deterministic algorithm for the problem, which led some to conjecture that no deterministic algorithm can do better. Huang

and Borodin [27] strengthened this conjecture by showing that a large class of deterministic algorithms resembling the optimal  $1/2$ -approximation randomized algorithm of [4] cannot achieve  $1/2$ -approximation for unconstrained submodular maximization.

The problem of maximizing a (non-monotone) submodular function subject to a matroid independence constraint (which generalizes a cardinality constraint) was given a deterministic  $(1/4 - \epsilon)$ -approximation algorithm by Lee et al. [31] and a randomized  $0.309$ -approximation algorithm by Vondrák [34]. Later, the randomized approximation ratio was improved to  $0.325$  using a simulated annealing technique [23], and then to  $1/e - o(1)$  [19] via an extension of the continuous greedy algorithm of [8]. All the above randomized results are based on the multilinear extension, and thus, are quite inefficient. Buchbinder et al. [5] described a simple randomized greedy algorithm designed specifically for a cardinality constraint, and used this algorithm to achieve a clean approximation ratio of  $1/e$  with a significantly better time complexity. Additionally, [5] also showed that  $1/e$  is not the correct approximation ratio for the case of a cardinality constraint by describing an (inefficient) polynomial time  $(1/e + 0.004)$ -approximation algorithm for it. On the hardness side, [23] showed that no polynomial time algorithm can have an approximation ratio better than  $0.491$  for the problem of maximizing a (non-monotone) submodular function subject to a cardinality constraint.

Recent works consider online and streaming variants of the above problems [7, 9] as well as faster algorithms [6].

## 2 Additional Notation

For every set  $S$  and element  $u$ , we denote the union  $S \cup \{u\}$  by  $S + u$ , and the expression  $S \setminus \{u\}$  by  $S - u$ . Given a submodular function  $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}$ , the marginal contribution of  $u$  to  $S$  is denoted by  $f(u \mid S) = f(S + u) - f(S)$ . Our algorithms **explicitly** maintain in each iteration  $i$  a (finite) distribution  $\mathcal{D}_i$  over possible states of the algorithm. Each distribution is represented as a multiset of tuples  $\{(p, S)\}$ , where  $S$  is a state and  $p$  is the probability of this state. Naturally, we require all the  $p$ -values to be positive and to add up to 1. We denote by  $|\mathcal{D}_i|$  the number of tuples in the distribution  $\mathcal{D}_i$ , and by  $\text{supp}(\mathcal{D}_i)$  the set of states represented by these tuples (which is also the set of states having a positive probability).

To simplify the presentation of our algorithms, we allow our distributions to contain multiple tuples with the same state. Moreover, there might even be multiple identical tuples (this is why the distributions have to be multisets). Whenever this happens, the meaning is that the probability of a state  $S$  is the sum of the  $p$ -values of tuples containing it. An implementation of our algorithm can either keep identical state tuples separate, or unify them. Our proofs are independent of such details. The pseudocode of our algorithms uses the first option, which requires us to assume the following semantic rules regarding multisets:

- Given multisets  $A$  and  $B$  of tuples, the multiplicity of a tuple in the union  $A \cup B$  is the sum of its multiplicities in the original sets.
- Given an expression of the form  $\{(p(x), S(x)) \mid x \in A\}$ , where  $A$  is a set and  $(p(x), S(x))$  is a tuple which is a function of an element  $x \in A$ . If there are multiple  $x$  values resulting a single tuple  $(p, S)$ , then we assume that the multiplicity of this tuple is equal to the number of such  $x$  values in  $A$ .

### 3 Unconstrained Submodular Maximization

In this section we present a deterministic  $1/2$ -approximation algorithm for the problem  $\max\{f(S) : S \subseteq \mathcal{N}\}$  whose formal description is given as Algorithm 1. Each state in the distribution maintained by our algorithm is a pair  $(X, Y)$  of sets. Technically, this means that the distribution should be a multiset of tuples  $(p, (X, Y))$ . However, we abuse notation and use tuples of the form  $(p, X, Y)$  instead. Additionally, we write  $\mathbb{E}_{\mathcal{D}_i}$  instead of  $\mathbb{E}_{(X,Y) \sim \mathcal{D}_i}$  to avoid visual cluttering.

---

**Algorithm 1:** Deterministic Unconstrained( $f$ )

---

```

1 Initialize:  $\mathcal{D}_0 = \{(1, \emptyset, \mathcal{N})\}$ .
2 Denote the elements of  $\mathcal{N}$  by  $u_1, u_2, \dots, u_n$ , in an arbitrary order (recall that  $n = |\mathcal{N}|$ ).
3 for  $i = 1$  to  $n$  do
4    $\forall (X, Y) \in \text{supp}(\mathcal{D}_{i-1})$ , let  $a_i(X) = f(X + u_i) - f(X)$ ,  $b_i(Y) = f(Y - u_i) - f(Y)$ .
5   Find an extreme point solution of the following linear formulation:

           (P)   $\mathbb{E}_{\mathcal{D}_{i-1}}[z(X, Y)a_i(X) + w(X, Y)b_i(Y)] \geq 2 \cdot \mathbb{E}_{\mathcal{D}_{i-1}}[z(X, Y)b_i(Y)]$ 
                 $\mathbb{E}_{\mathcal{D}_{i-1}}[z(X, Y)a_i(X) + w(X, Y)b_i(Y)] \geq 2 \cdot \mathbb{E}_{\mathcal{D}_{i-1}}[w(X, Y)a_i(X)]$ 
                 $z(X, Y) + w(X, Y) = 1 \quad \forall (X, Y) \in \text{supp}(\mathcal{D}_{i-1})$ 
                 $z(X, Y), w(X, Y) \geq 0 \quad \forall (X, Y) \in \text{supp}(\mathcal{D}_{i-1})$ 

6   Construct a new distribution:

        $\mathcal{D}_i \leftarrow \{(z(X, Y) \cdot \Pr_{\mathcal{D}_{i-1}}[(X, Y)], (X + u_i, Y)) \mid (X, Y) \in \text{supp}(\mathcal{D}_{i-1}), z(X, Y) > 0\}$ 
        $\cup \{(w(X, Y) \cdot \Pr_{\mathcal{D}_{i-1}}[(X, Y)], (X, Y - u_i)) \mid (X, Y) \in \text{supp}(\mathcal{D}_{i-1}), w(X, Y) > 0\}$  .

7 return  $\arg \max_{(X,Y) \in \text{supp}(\mathcal{D}_n)} \{f(X)\}$  (equivalent to  $\arg \max_{(X,Y) \in \text{supp}(\mathcal{D}_n)} \{f(Y)\}$ ).

```

---

We begin the analysis of Algorithm 1 with the following simple observations.

**Observation 3.1.** *The following holds for every iteration  $1 \leq i \leq n$  of Algorithm 1:*

1. *For every state  $(X, Y) \in \text{supp}(\mathcal{D}_i)$ ,  $X \cap \{u_{i+1}, \dots, u_n\} = \emptyset$ ,  $\{u_{i+1}, \dots, u_n\} \subseteq Y$  and  $X \cap \{u_1, \dots, u_i\} = Y \cap \{u_1, \dots, u_i\}$ .*
2. *The total sum of the probabilities in  $\mathcal{D}_i$  is 1, and thus,  $\mathcal{D}_i$  is a valid distribution.*
3. *The formulation (P) is feasible. In particular, one feasible solution assigns for every state  $(X, Y) \in \text{supp}(\mathcal{D}_{i-1})$ :*

$$z(X, Y) = \frac{\max\{0, a_i(X)\}}{\max\{0, a_i(X)\} + \max\{0, b_i(Y)\}} \quad (\text{or } 1 \text{ if the denominator is } 0)$$

and

$$w(X, Y) = 1 - z(X, Y) = \frac{\max\{0, b_i(Y)\}}{\max\{0, a_i(X)\} + \max\{0, b_i(Y)\}} \quad (\text{or } 0 \text{ if the denominator is } 0) .$$

4. *For any extreme point of (P) there are at most  $2 + |\mathcal{D}_{i-1}|$  non-zero variables. Thus,  $|\mathcal{D}_i| \leq 2 + |\mathcal{D}_{i-1}|$ , and  $|\mathcal{D}_n| \leq 2n + 1$ .*

**Remark:** Item 4 of Observation 3.1 is the only place in the analysis of Algorithm 1 where we use the fact that the algorithm finds an extreme point solution of  $(P)$ . Claim A.1 shows that one can in fact compute a solution for  $(P)$  having at most  $1 + |\mathcal{D}_{i-1}|$  non-zero variables. Moreover, it is possible to compute such a solution in near linear time, implying that implementation of Algorithm 1 does not require an LP solver.

*Proof.* The proof of the observation is by induction on  $i$ . Assume the observation holds for every  $1 \leq i' < i$ , and let us prove it for  $i$ . Observe that item (1) holds for  $i - 1$  either by the induction hypothesis or (for  $i = 1$ ) by the fact that  $\mathcal{D}_0$  trivially obeys it. Additionally, it is easy to see that every state of  $\mathcal{D}_i$  is obtained from a state of  $\mathcal{D}_{i-1}$  by either adding  $u_i$  to  $X$  or removing  $u_i$  from  $Y$ . Hence item (1) is maintained. Similarly, item (2) holds for  $i - 1$ . Since  $z(X, Y) + w(X, Y) = 1$  for every state  $(X, Y) \in \text{supp}(\mathcal{D}_{i-1})$ , the distribution  $\mathcal{D}_i$  contains up two tuples resulting from  $(X, Y)$ , and these tuples exactly split the probability  $(X, Y)$  has in  $\mathcal{D}_{i-1}$ . Thus, the sum of the probabilities in  $\mathcal{D}_i$  is equal to their sum in  $\mathcal{D}_{i-1}$ .

To see why item (3) holds, observe first that by (1)  $X \subseteq Y \setminus \{u_i\}$  for each  $(X, Y) \in \text{supp}(\mathcal{D}_{i-1})$ , and therefore, by submodularity,

$$a_i(X) + b_i(Y) = [f(X + u_i) - f(X)] + [f(Y - u_i) - f(Y)] \geq 0 .$$

Next, we prove that the first constraint of  $(P)$  is satisfied by the assignment suggested by (3). Proving that the second constraint of  $(P)$  is satisfied by the assignment can be done similarly.

If  $a_i(X) = b_i(Y) = 0$  for some state  $(X, Y)$ , then  $(X, Y)$  does not contribute to either side of the first constraint of  $(P)$ , and we may ignore it. Thus, we may assume that either  $a_i(X)$  or  $b_i(Y)$  is strictly non-zero for every state. Plugging the assignment suggested by (3) under this assumption into the first constraint in  $(P)$ , we get:

$$\begin{aligned} & \mathbb{E}_{\mathcal{D}_{i-1}}[z(X, Y) \cdot a_i(X) + w(X, Y) \cdot b_i(Y)] \\ &= \mathbb{E}_{\mathcal{D}_{i-1}} \left[ \frac{\max\{0, a_i(X)\} \cdot a_i(X)}{\max\{0, a_i(X)\} + \max\{0, b_i(Y)\}} + \frac{\max\{0, b_i(Y)\} \cdot b_i(Y)}{\max\{0, a_i(X)\} + \max\{0, b_i(Y)\}} \right] \\ &\geq 2 \cdot \mathbb{E}_{\mathcal{D}_{i-1}} \left[ \frac{\max\{0, a_i(X)\} \cdot b_i(Y)}{\max\{0, a_i(X)\} + \max\{0, b_i(Y)\}} \right] = 2 \cdot \mathbb{E}_{\mathcal{D}_{i-1}}[z(X, Y) \cdot b_i(Y)] . \end{aligned}$$

The final inequality holds even without the expectation due to the following argument: if either  $a_i(X) < 0$  or  $b_i(Y) < 0$ , then the LHS is non-negative while the RHS is non-positive. On the other hand, if  $a_i(X) \geq 0$  and  $b_i(Y) \geq 0$  the inequality reduces to  $a_i^2(X) + b_i^2(Y) \geq 2a_i(X)b_i(Y)$ , which clearly holds.

Item (4) follows immediately by the properties of an extreme point. Since  $(P)$  has  $2 + |\mathcal{D}_{i-1}|$  constraints, an extreme point of  $(P)$  has at most  $2 + |\mathcal{D}_{i-1}|$  non-zero variables. Since a single tuple is added to  $\mathcal{D}_i$  for every non-zero variable, the size of  $\mathcal{D}_i$  is upper bounded by  $2 + |\mathcal{D}_{i-1}|$ .  $\square$

Let  $OPT \subseteq \mathcal{N}$  be the optimal solution for the problem  $\{f(S) : S \subseteq \mathcal{N}\}$  that we want to approximate, and let  $OPT(X, Y)$  be a shorthand for the set  $((OPT \cup X) \cap Y)$ . The following is the main lemma we need in order to analyze Algorithm 1.

**Lemma 3.2.** *For any iteration  $1 \leq i \leq n$  of Algorithm 1,*

$$\mathbb{E}_{\mathcal{D}_i}[f(X) + f(Y)] - \mathbb{E}_{\mathcal{D}_{i-1}}[f(X) + f(Y)] \geq 2 \cdot (\mathbb{E}_{\mathcal{D}_{i-1}}[f(OPT(X, Y))] - \mathbb{E}_{\mathcal{D}_i}[f(OPT(X, Y))]) .$$

*Proof.* Observe that whenever  $u_i \notin OPT$ :

$$\begin{aligned} \mathbb{E}_{\mathcal{D}_{i-1}}[f(OPT(X, Y))] - \mathbb{E}_{\mathcal{D}_i}[f(OPT(X, Y))] \\ = \mathbb{E}_{\mathcal{D}_{i-1}}[z(X, Y) \cdot (f(OPT(X, Y)) - f(OPT(X + u_i, Y)))] \\ \leq \mathbb{E}_{\mathcal{D}_{i-1}}[z(X, Y) \cdot (f(Y - u_i) - f(Y))] = \mathbb{E}_{\mathcal{D}_{i-1}}[z(X, Y) \cdot b_i(Y)] , \end{aligned}$$

where the inequality follows by submodularity since  $OPT(X, Y) \subseteq Y - u_i$ . A similar argument can be used to show that whenever  $u_i \in OPT$ :

$$\mathbb{E}_{\mathcal{D}_{i-1}}[f(OPT(X, Y))] - \mathbb{E}_{\mathcal{D}_i}[f(OPT(X, Y))] \leq \mathbb{E}_{\mathcal{D}_{i-1}}[w(X, Y) \cdot a_i(X)] .$$

The lemma now follows by combining the above observations with the next inequality:

$$\begin{aligned} \mathbb{E}_{\mathcal{D}_i}[f(X) + f(Y)] - \mathbb{E}_{\mathcal{D}_{i-1}}[f(X) + f(Y)] \\ = \mathbb{E}_{\mathcal{D}_{i-1}}[z(X, Y) \cdot (f(X + u_i) - f(X)) + w(X, Y) \cdot (f(Y - u_i) - f(Y))] \\ = \mathbb{E}_{\mathcal{D}_{i-1}}[(z(X, Y) \cdot a_i(X) + w(X, Y) \cdot b_i(Y))] \\ \geq 2 \cdot \max \{ \mathbb{E}_{\mathcal{D}_{i-1}}[z(X, Y) \cdot b_i(Y)], \mathbb{E}_{\mathcal{D}_{i-1}}[w(X, Y) \cdot a_i(X)] \} , \end{aligned}$$

where the inequality follows by the constraints of  $(P)$ .  $\square$

We can now prove the next theorem, which implies Theorem 1.1.

**Theorem 3.3.** *Algorithm 1 is a  $\frac{1}{2}$ -approximation algorithm performing  $O(n^2)$  value oracle queries.*

*Proof.* Adding up Lemma 3.2 over  $1 \leq i \leq n$  we get:

$$\mathbb{E}_{\mathcal{D}_n}[f(X) + f(Y)] - \mathbb{E}_{\mathcal{D}_0}[f(X) + f(Y)] \geq 2 \cdot (\mathbb{E}_{\mathcal{D}_0}[f(OPT(X, Y))] - \mathbb{E}_{\mathcal{D}_n}[f(OPT(X, Y))]) .$$

The single state in the support of  $\mathcal{D}_0$  is  $(\emptyset, \mathcal{N})$ . Hence,  $\mathbb{E}_{\mathcal{D}_0}[f(OPT(X, Y))] = f(OPT(\emptyset, \mathcal{N})) = f(OPT)$ . On the other hand, for every state  $(X_n, Y_n) \in \text{supp}(\mathcal{D}_n)$  we have  $X_n = Y_n$  by Observation 3.1, and thus,  $OPT(X_n, Y_n) = X_n = Y_n$ . Plugging all these observations into the last inequality gives:

$$2 \cdot \mathbb{E}_{\mathcal{D}_n}[f(X)] - (f(\emptyset) + f(\mathcal{N})) \geq 2 \cdot (f(OPT) - \mathbb{E}_{\mathcal{D}_n}[f(X)]) .$$

Using an averaging argument and the non-negativity of  $f$  we now get:

$$\arg \max_{(X, Y) \in \text{supp}(\mathcal{D}_n)} \{f(X)\} \geq \mathbb{E}_{\mathcal{D}_n}[f(X)] \geq \frac{f(OPT)}{2} + \frac{f(\emptyset) + f(\mathcal{N})}{4} \geq \frac{f(OPT)}{2} .$$

Observation 3.1 shows that  $|\mathcal{D}_i| \leq 2i + 1$  for every  $1 \leq i \leq n$ . Since Algorithm 1 performs 2 oracle queries for every state in  $\text{supp}(\mathcal{D}_{i-1})$ , the number of such queries done during the  $i$ -th iteration is at most  $4i - 2$ . Adding up the last bound over all iterations we get a bound  $O(n^2)$  on the total number of oracle queries made by the algorithm.  $\square$

## 4 Cardinality Constraints

In this section we present a deterministic  $1/e$ -approximation algorithm for the problem  $\max\{f(S) : |S| \leq k\}$  whose formal description is given as Algorithm 2. Each state in the distribution maintained by our algorithm is a set  $S$ .

We first make the following simple observations.

---

**Algorithm 2:** Deterministic Cardinality( $f, k$ )

---

```

1 Initialize:  $\mathcal{D}_0 = \{(1, \emptyset)\}$ .
2 for  $i = 1$  to  $k$  do
3   Let  $M_i \subseteq \mathcal{N}$  be a subset of at most  $k$  elements maximizing  $\sum_{u \in M_i} \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[f(u \mid S)]$ .
4   Find an optimal extreme point solution of the following linear formulation:

      (P)  max  $\sum_{u \in M_i} \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[x(u, S) \cdot f(u \mid S)]$ 
             $\mathbb{E}_{S \sim \mathcal{D}_{i-1}}[x(u, S)] \leq 1/k \cdot \Pr_{S \sim \mathcal{D}_{i-1}}[u \notin S] \quad \forall u \in M_i$ 
             $\sum_{u \in M_i} x(u, S) + \ell(S) = 1 \quad \forall S \in \text{supp}(\mathcal{D}_{i-1})$ 
             $x(u, S), \ell(S) \geq 0 \quad \forall u \in M_i, S \in \text{supp}(\mathcal{D}_{i-1})$ 

5   Construct a new distribution:

       $\mathcal{D}_i \leftarrow \{(x(u, S) \cdot \Pr_{\mathcal{D}_{i-1}}[S], S + u) \mid u \in M_i, S \in \text{supp}(\mathcal{D}_{i-1}), x(u, S) > 0\}$ 
       $\cup \{(\ell(S) \cdot \Pr_{\mathcal{D}_{i-1}}[S], S) \mid S \in \text{supp}(\mathcal{D}_{i-1}), \ell(S) > 0\}$  .

6 Return  $\arg \max_{S \in \text{supp}(\mathcal{D}_k)} \{f(S_k)\}$ .
```

---

**Observation 4.1.** *The following holds for every iteration  $i = 1, \dots, k$  of Algorithm 2:*

1. *The assignment  $x(u, S) = 1/k \cdot \mathbf{1}[u \notin S]$  and  $\ell(S) = 1 - |M_i \setminus S|/k$  for every  $S \in \text{supp}(\mathcal{D}_{i-1})$  and  $u \in M_i$  is a feasible assignment for the formulation (P).*
2. *The total sum of the probabilities in  $\mathcal{D}_i$  is 1, and thus,  $\mathcal{D}_i$  is a valid distribution.*
3.  *$|\mathcal{D}_i| \leq k + |\mathcal{D}_{i-1}|$ . Thus,  $|\mathcal{D}_k| \leq k^2 + 1$ .*

*Proof.* The proof of the observation is by induction on  $i$ . Assume the observation holds for every  $1 \leq i' < i$ , and let us prove it for  $i$ . It is easy to verify that item (1) holds given that  $\mathcal{D}_{i-1}$  is a valid distribution. To see why item (2) holds, observe that the sum of the probabilities in  $\mathcal{D}_i$  is:

$$\sum_{S \in \text{supp}(\mathcal{D}_{i-1})} \Pr_{\mathcal{D}_{i-1}}[S] \cdot \left[ \sum_{u \in M_i} x(u, S) + \ell(S) \right] = \sum_{S \in \text{supp}(\mathcal{D}_{i-1})} \Pr_{\mathcal{D}_{i-1}}[S] = 1 .$$

Finally, to prove item (3) notice that the number of constraints in (P) at iteration  $i$  is at most  $k + |\text{supp}(\mathcal{D}_{i-1})| \leq k + |\mathcal{D}_{i-1}|$ . By the properties of extreme point solutions the total number of variables that are strictly greater than zero is upper bounded by the number of (tight) constraints. Since a single set is added to  $\mathcal{D}_i$  for every non-zero  $x(u, S)$  or  $\ell(S)$  variable, the size of  $\mathcal{D}_i$  is also upper bounded by  $k + |\mathcal{D}_{i-1}|$ .  $\square$

The next lemma upper bounds the probability of an item to be in a set chosen according to the distributions defined by Algorithm 2.

**Lemma 4.2.** *For every element  $u \in \mathcal{N}$  and  $0 \leq i \leq k$ :*

$$\Pr_{S \sim \mathcal{D}_i}[u \notin S] \geq \left(1 - \frac{1}{k}\right)^i .$$



*Proof.* The proof of the lemma is by induction on  $i$ . The distribution  $\mathcal{D}_0$  gives a probability 1 to the empty set, and thus,  $\Pr_{S \sim \mathcal{D}_0}[u \notin S] = 1$  for every  $u \in \mathcal{N}$ , i.e., the base case  $i = 0$  holds. Next, assume the lemma holds for  $0 \leq i - 1$ , and let us prove it for  $i$ .

For simplicity of notation, let us define  $x(u, S)$  to be 0 for every  $u \notin M_i$ . A set  $S \in \text{supp}(\mathcal{D}_i)$  contains the element  $u$  in two cases: if it is constructed from a set in the support of  $\mathcal{D}_{i-1}$  that contains  $u$ , or it is constructed by adding  $u$  to a set in the support of  $\mathcal{D}_{i-1}$ . Using this observation we get the bound:

$$\begin{aligned} \Pr_{S \sim \mathcal{D}_i}[u \notin S] &= \sum_{\substack{S \in \text{supp}(\mathcal{D}_{i-1}) \\ u \notin S}} \Pr_{\mathcal{D}_{i-1}}[S] \cdot \left[ \sum_{u' \in M_i - u} x(u', S) + \ell(S) \right] \\ &\geq \sum_{\substack{S \in \text{supp}(\mathcal{D}_{i-1}) \\ u \notin S}} \Pr_{\mathcal{D}_{i-1}}[S] \cdot \left[ \sum_{u' \in M_i} x(u', S) + \ell(S) \right] - \sum_{S \in \text{supp}(\mathcal{D}_{i-1})} x(u, S) \cdot \Pr_{\mathcal{D}_{i-1}}[S] \\ &= \Pr_{S \sim \mathcal{D}_{i-1}}[u \notin S] - \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[x(u, S)] \geq (1 - 1/k) \cdot \Pr_{S \sim \mathcal{D}_{i-1}}[u \notin S] \geq \left(1 - \frac{1}{k}\right)^i, \end{aligned}$$

where the second equality holds by the second constraint of  $(P)$ , the second inequality holds by the first constraint of  $(P)$  and the last inequality holds by the induction hypothesis.  $\square$

The following lemma is an immediate implication of Lemma 2.2 of [5]. For completeness, we give an independent proof of it Appendix A.

**Lemma 4.3.** *For any subset  $T$  and a distribution  $\mathcal{D}$ ,*

$$\mathbb{E}_{S \sim \mathcal{D}}[f(T \cup S)] \geq f(S) \cdot \min_{u \in N} \Pr_{S \sim \mathcal{D}}[u \notin S].$$

The next lemma is the last component we need in order to analyze Algorithm 2.

**Lemma 4.4.** *For any iteration  $1 \leq i \leq k$  of Algorithm 2,*

$$\mathbb{E}_{S \sim \mathcal{D}_i}[f(S)] - \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[f(S)] \geq 1/k \cdot \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[f(OPT \cup S) - f(S)].$$

*Proof.* One can view the construction of  $\mathcal{D}_i$  in the following way: the probability of every set  $S \in \text{supp}(\mathcal{D}_{i-1})$  is split. A fraction of  $\ell(S)$  of this probability is kept for  $S$ , and for every  $u \in M_i$  a fraction of  $x(u, S)$  of this probability is transferred to  $S + u$ . Using this view we get:

$$\begin{aligned} \mathbb{E}_{S \sim \mathcal{D}_i}[f(S)] - \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[f(S)] &= \sum_{u \in M_i} \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[x(u, S) \cdot f(u \mid S)] \geq \frac{1}{k} \cdot \sum_{u \in M_i} \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[f(u \mid S)] \\ &\geq \frac{1}{k} \cdot \sum_{u \in OPT} \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[f(u \mid S)] \geq \frac{1}{k} \cdot \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[f(OPT \cup S) - f(S)], \end{aligned}$$

where the first inequality holds since the solution found by Algorithm 2 must be at least as good as the feasible solution given by Observation 4.1, the second inequality holds by the definition of  $M_i$  and the last inequality holds by submodularity.  $\square$

We can now prove the next theorem, which implies Theorem 1.2 (note that Theorem 1.2 is trivial for  $k = 1$ ).

**Theorem 4.5.** For  $k \geq 2$ , the approximation ratio of Algorithm 2 is at least  $(1 - \frac{1}{k})^{k-1} \geq 1/e$ , and it performs  $O(k^2n)$  oracle queries.

*Proof.* By combining Lemmata 4.2, 4.3 and 4.4, we get

$$\mathbb{E}_{S \sim \mathcal{D}_i}[f(S)] - \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[f(S)] \geq \frac{1}{k} \cdot \mathbb{E}_{S \sim \mathcal{D}_{i-1}} \left[ \left(1 - \frac{1}{k}\right)^{i-1} \cdot f(OPT) - f(S) \right] . \quad (1)$$

Next, we prove by induction that:

$$\mathbb{E}_{S \sim \mathcal{D}_i}[f(S)] \geq \frac{i}{k} \cdot \left(1 - \frac{1}{k}\right)^{i-1} f(OPT) .$$

For  $i = 0$ , this is true since  $f(\emptyset) \geq 0 = (0/k) \cdot (1 - 1/k)^{-1} \cdot f(OPT)$ . Assume now that the claim holds for every  $i' < i$ , let us prove it for  $i > 0$ .

$$\begin{aligned} \mathbb{E}_{S \sim \mathcal{D}_i}[f(S)] &\geq \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[f(S)] + \frac{1}{k} \cdot \mathbb{E}_{S \sim \mathcal{D}_{i-1}} \left[ \left(1 - \frac{1}{k}\right)^{i-1} \cdot f(OPT) - f(S) \right] \\ &= \left(1 - \frac{1}{k}\right) \cdot \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[f(S)] + \frac{1}{k} \cdot \left(1 - \frac{1}{k}\right)^{i-1} \cdot f(OPT) \\ &\geq \left(1 - \frac{1}{k}\right) \cdot \frac{i-1}{k} \cdot \left(1 - \frac{1}{k}\right)^{i-2} \cdot f(OPT) + \frac{1}{k} \cdot \left(1 - \frac{1}{k}\right)^{i-1} \cdot f(OPT) \\ &= \frac{i}{k} \cdot \left(1 - \frac{1}{k}\right)^{i-1} \cdot f(OPT) , \end{aligned}$$

where the first inequality follows by inequality (1), and the second inequality follows by the induction hypothesis.

The approximation ratio guaranteed by the theorem follows immediately by plugging  $k$  into the induction hypothesis. Finally, Observation 4.1 implies  $|\mathcal{D}_i| \leq ik + 1$ , and thus, in the  $i$ -th iteration Algorithm 2 makes at most  $n \cdot \text{supp}(\mathcal{D}_{i-1}) \leq n \cdot |\mathcal{D}_{i-1}| \leq nik$  oracle queries. Thus, the total number of oracle queries in all the iterations is at most  $O(k^2n)$ .  $\square$

## 4.1 A Tight Example for Algorithm 2

In this section we give an example of a “bad” instance for which Algorithm 2 has an approximation ratio of at most  $e^{-1} + O(\frac{1}{k})$ . Specifically, the optimal solution for the instance we describe has a value of at least 1, while Algorithm 2 may produce a set of value at most  $e^{-1} + O(\frac{1}{k})$ . In the rest of this section we assume  $k$  is larger than some arbitrary constant  $\ell$  (to be determined later).

The ground set  $\mathcal{N}$  of our bad instance is the union of two sets  $O$  and  $Y$ , both of size  $k$  (if one wishes to have  $n > 2k$ , it is possible to add an arbitrary number of elements that do not affect the objective function). The objective function of the instance is the function  $f: 2^{\mathcal{N}} \rightarrow \mathbb{R}^+$  defined as follows,

$$f(S) = \frac{|S \cap O|}{k} \cdot \left(1 - \frac{|S \cap Y|}{k}\right) + \left[ g\left(\frac{|S \cap Y|}{k}\right) + \frac{\ell \cdot |S \cap Y|}{k^2} \right] \cdot \left(1 - \frac{|S \cap O|}{k}\right) ,$$

where  $g: [0, 1] \rightarrow [0, 1]$  is a function given by the following formula:

$$g(x) = \begin{cases} (x-1) \cdot \ln(1-x) & \text{for } 0 \leq x \leq 1 - e^{-1} , \\ e^{-1} & \text{for } 1 - e^{-1} \leq x \leq 1 . \end{cases}$$

Observe that  $g(x)$  is a continuous function. Additionally, we note that,

$$g'(x) = \begin{cases} 1 + \ln(1-x) & \text{for } 0 \leq x \leq 1 - e^{-1} , \\ 0 & \text{for } 1 - e^{-1} \leq x \leq 1 , \end{cases}$$

and

$$g''(x) = \begin{cases} \frac{1}{x-1} & \text{for } 0 \leq x < 1 - e^{-1} , \\ 0 & \text{for } 1 - e^{-1} < x \leq 1 . \end{cases}$$

Observe that  $g'$  is always non-negative and  $g''$  is always non-positive. Thus,  $g$  is a non-decreasing continuous concave function with  $g(0) = 0$  and  $g(1) = e^{-1}$ .

It is useful to find expressions for the marginal contribution of an element  $u \in \mathcal{N}$  to a set  $S \subseteq \mathcal{N}$  given the objective  $f$ . Let  $y = |S \cap Y|/k$  and  $x = |S \cap O|/k$ , then

$$f(u \mid S) = -\frac{x}{k} + (1-x) \left( g(y + 1/k) - g(y) + \frac{\ell}{k^2} \right) \quad \forall u \in Y \setminus S \quad (2)$$

$$f(u \mid S) = \frac{1}{k} \left( (1-y) - \left( g(y) + \frac{\ell y}{k} \right) \right) \quad \forall u \in O \setminus S \quad (3)$$

**Observation 4.6.** *The function  $f$  is a submodular function.*

*Proof.* The marginal (2) of an element  $u \in Y$  is a decreasing function of  $x$  since  $g$  is non-decreasing and of  $y$  since  $g$  is concave. On the other hand, the marginal (3) of an element  $u \in O$  is independent of  $x$  and a decreasing function of  $y$  since  $g$  is non-decreasing.  $\square$

In Appendix B we complete the proof. We show that given the above bad instance, Algorithm 2 may terminate with a distribution over subsets of  $Y$ . The value of  $f$  for any such set  $S$  is at most:

$$f(S) = g\left(\frac{|S|}{k}\right) + \frac{\ell \cdot |S|}{k^2} \leq e^{-1} + \frac{\ell}{k} = e^{-1} + O\left(\frac{1}{k}\right) .$$

On the other hand,  $O$  is a feasible solution and  $f(O) = 1$ . Thus, completing the proof.

## 5 Conclusions

In this paper we proposed a new technique for derandomization of algorithms in the area of submodular function maximization. For unconstrained submodular maximization we showed that randomization is not necessary for obtaining the best possible approximation ratio. For submodular maximization with a cardinality constraint we obtained nearly the best known result.

The main interesting open question is whether algorithms that are based on the multilinear extension can be derandomized. In particular, it is interesting whether the continuous greedy approach [8, 19] used to obtain optimal results for maximizing a monotone submodular function subject to a matroid independence constraint can be derandomized. One possible direction is to try to approximate the multilinear extension function deterministically using its special properties. Another interesting question is whether the number of oracle calls of the deterministic algorithms can be reduced. A possible way to speed up algorithms produced by our method is to keep the size of the distributions small by avoiding splitting sets when this results in sets of a too low probability. As long as only low probability sets are affected, this should not significantly decrease the quality of the output, while reducing the number of oracle queries needed (and speeding up the algorithm).

## References

- [1] A. A. Ageev and M. I. Sviridenko. An 0.828 approximation algorithm for the uncapacitated facility location problem. *Discrete Appl. Math.*, 93:149–156, July 1999.
- [2] Noga Alon and Joel H. Spencer. *The Probabilistic Method*. Wiley Interscience Series in Discrete Mathematics and Optimization. John Wiley and Sons, Inc., second edition, 2000.
- [3] Yossi Azar, Iftah Gamzu, and Ran Roth. Submodular max-sat. In *ESA*, pages 323–334, 2011.
- [4] Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. A tight linear time  $(1/2)$ -approximation for unconstrained submodular maximization, 2012. FOCS 2012.
- [5] Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *SODA*, pages 1433–1452, 2014.
- [6] Niv Buchbinder, Moran Feldman, and Roy Schwartz. Comparing apples and oranges: Query tradeoff in submodular maximization. In *SODA*, pages 1149–1168, 2015.
- [7] Niv Buchbinder, Moran Feldman, and Roy Schwartz. Online submodular maximization with preemption. In *SODA*, pages 1202–1216, 2015.
- [8] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.
- [9] Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *ICALP*, pages 318–330, 2015.
- [10] Chandra Chekuri and Sanjeev Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM J. Comput.*, 35(3):713–728, September 2005.
- [11] Reuven Cohen, Liran Katzir, and Danny Raz. An efficient approximation for the generalized assignment problem. *Information Processing Letters*, 100(4):162–166, 2006.
- [12] G. Cornuejols, M. L. Fisher, and G. L. Nemhauser. Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. *Management Sciences*, 23:789–810, 1977.
- [13] G. Cornuejols, M. L. Fisher, and G. L. Nemhauser. On the uncapacitated location problem. *Annals of Discrete Mathematics*, 1:163–177, 1977.
- [14] Shahar Dobzinski and Michael Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In *SODA*, pages 1064–1073, 2006.
- [15] Uriel Feige and Michel X. Goemans. Approximating the value of two prover proof systems, with applications to max 2sat and max dicut. In *ISTCS*, pages 182–189, 1995.
- [16] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.
- [17] Uriel Feige and Jan Vondrák. Approximation algorithms for allocation problems: Improving the factor of  $1 - 1/e$ . In *FOCS*, pages 667–676, 2006.

- [18] Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. Nonmonotone submodular maximization via a structural continuous greedy algorithm. In *ICALP*, pages 342–353, 2011.
- [19] Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *FOCS*, 2011.
- [20] Yuval Filmus and Justin Ward. Monotone submodular maximization over a matroid via non-oblivious local search. *SIAM J. Comput.*, 43(2):514–542, 2014.
- [21] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions – ii. In *Polyhedral Combinatorics*, volume 8 of *Mathematical Programming Studies*, pages 73–87. Springer Berlin Heidelberg, 1978.
- [22] Lisa Fleischer, Michel X. Goemans, Vahab S. Mirrokni, and Maxim Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *SODA*, pages 611–620, 2006.
- [23] Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *SODA*, pages 1098–1117, 2011.
- [24] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.
- [25] Eran Halperin and Uri Zwick. Combinatorial approximation algorithms for the maximum directed cut problem. In *SODA*, pages 1–7, 2001.
- [26] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48:798–859, July 2001.
- [27] Norman Huang and Allan Borodin. Bounds on double-sided myopic algorithms for unconstrained non-monotone submodular maximization. In *ISAAC*, pages 528–539, 2014.
- [28] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.
- [29] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O’Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM J. Comput.*, 37:319–357, April 2007.
- [30] Ariel Kulik, Hadas Shachnai, and Tami Tamir. Approximations for monotone and non-monotone submodular maximization with knapsack constraints. *Math. Oper. Res.*, 38(4):729–739, 2013.
- [31] Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Maximizing non-monotone submodular functions under matroid or knapsack constraints. *SIAM Journal on Discrete Mathematics*, 23(4):2053–2078, 2010.
- [32] Michael Mitzenmacher and Eli Upfal. *Probability and Computing, Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.
- [33] Luca Trevisan, Gregory B. Sorkin, Madhu Sudan, and David P. Williamson. Gadgets, approximation, and linear programming. *SIAM J. Comput.*, 29:2074–2097, April 2000.
- [34] Jan Vondrák. Symmetry and approximability of submodular maximization problems. *SIAM J. Comput.*, 42(1):265–304, 2013.

## A Omitted Proofs

The following claim slightly improves item 4 of Observation 3.1, and shows that every iteration of Algorithm 1 can be implemented in near linear time.

**Claim A.1.** *Let  $(P)$  be the formulation in Algorithm 1. There always exists a solution to  $(P)$  with at most  $1 + |\mathcal{D}_{i-1}|$  non-zero variables. Moreover, this solution can be computed in near linear time.*

*Proof.* To get the desired solution of  $(P)$  we need make a few simple manipulations to  $(P)$ . First, we replace the first constraint of  $(P)$  with an objective function asking to maximize:

$$\mathbb{E}_{\mathcal{D}_{i-1}}[z(X, Y) \cdot a_i(X) + w(X, Y) \cdot b_i(Y)] - 2 \cdot \mathbb{E}_{\mathcal{D}_{i-1}}[z(X, Y)b_i(Y)] ,$$

Since  $(P)$  is feasible by Observation 3.1, any optimal solution for the new formulation is a feasible solution for  $(P)$ . We can simplify the new objective of  $(P)$  by removing constants and using the fact that  $w(X, Y)$  is fully determined by  $z(X, Y)$  due to the equality  $z(X, Y) + w(X, Y) = 1$ . This yields:

$$\mathbb{E}_{\mathcal{D}_{i-1}}[z(X, Y) \cdot (a_i(X) - 3b_i(Y))] .$$

Similarly, by exchanging terms, the second constraint of  $(P)$  can be replaced with the equivalent form:

$$\mathbb{E}_{\mathcal{D}_{i-1}}[z(X, Y) \cdot (b_i(Y) - 3 \cdot a_i(X))] \leq \mathbb{E}_{\mathcal{D}_{i-1}}[b_i(Y) - 2 \cdot a_i(X)] .$$

The resulting linear program, for which we need to find an optimal solution, is a variant of the fractional knapsack problem of the following form (where the number  $m$  of items is  $|\text{supp}(\mathcal{D}_{i-1})| \leq |\mathcal{D}_{i-1}|$ ).

$$\begin{aligned} \max \quad & \sum_{j=1}^m v_j \cdot z_j \\ & \sum_{j=1}^m s_j \cdot z_j \leq B \\ & 0 \leq z_j \leq 1 \quad \forall 1 \leq j \leq m \end{aligned}$$

The only change compared to a standard (fractional) knapsack problem is that  $v_j$  and  $s_j$  may have negative values (such values can be interpreted as an option to buy additional knapsack space). However, a simple modification of the, so called, density rule can be used to solve the problem optimally. First, take to the solution all items with  $v_j > 0$  and  $s_j < 0$ . Also, omit all items of  $v_j < 0$  and  $s_j > 0$ . This leaves us with two types of items: “positive” items having  $v_j, s_j \geq 0$ , and “negative” items having  $v_j, s_j < 0$ . We sort the positive items in decreasing order of  $v_j/s_j$  (intuitively, the value that we can earn per unit of the knapsack). Similarly, we sort the negative items increasingly by  $v_j/s_j$  (intuitively, the price we need to pay to buy a unit of the knapsack). The algorithm then starts by (fractionally) taking the first positive items until the knapsack becomes full (or we are out of positive items). We then continue (fractionally) taking positive items and negative items in parallel as long as the value per unit gained by the positive item is at least equal to the price per unit paid for the negative item. It is easy to see that this algorithm produces an optimal solution for the above fractional knapsack problem, and whenever it terminates there is only a single (positive or negative) item taken fractionally.

To complete the proof of the claim observe that when translating a solution for the fractional knapsack problem into a solution for the original formulation  $(P)$  we get two non-zero variables for every item taken fractionally, and one non-zero variable for every other item.  $\square$

Lemma 4.4 is an immediate implication of Lemma 2.2 of [5]. For completeness, we give an independent proof of it.

**Lemma 4.3.** *For any subset  $T$  and a distribution  $\mathcal{D}$ ,*

$$\mathbb{E}_{S \sim \mathcal{D}}[f(T \cup S)] \geq f(S) \cdot \min_{u \in N} \Pr_{S \sim \mathcal{D}}[u \notin S] .$$

*Proof.* Let  $u_1, u_2, \dots, u_n$  denote the elements of  $N$  sorted by a non-decreasing order of the probability  $\Pr_{S \sim \mathcal{D}}[u \notin S]$ . Additionally, let  $A_i = \{u_1, u_2, \dots, u_i\}$  be the set of the first  $i$  elements in this order (for every  $0 \leq i \leq n$ ). Then,

$$\begin{aligned} \mathbb{E}_{S \sim \mathcal{D}}[f(T \cup S)] &= f(T) + \sum_{i=1}^n \mathbb{E}_{S \sim \mathcal{D}}[1[u_i \in S] \cdot f(u_i \mid T \cup (A_{i-1} \cap S))] \\ &\geq f(T) + \sum_{i=1}^n \mathbb{E}_{S \sim \mathcal{D}}[1[u_i \in S] \cdot f(u_i \mid T \cup A_{i-1})] = f(T) + \sum_{i=1}^n f(u_i \mid T \cup A_{i-1}) \cdot \Pr_{S \sim \mathcal{D}}[u_i \in S] \\ &= \Pr_{S \sim \mathcal{D}}[u_1 \notin S] \cdot f(T) + \sum_{i=1}^{n-1} (\Pr_{S \sim \mathcal{D}}[u_i \in S] - \Pr_{S \sim \mathcal{D}}[u_{i+1} \in S]) \cdot f(T \cup A_i) \\ &\quad + \Pr_{S \sim \mathcal{D}}[u_n \in S] \cdot f(N) \geq \Pr_{S \sim \mathcal{D}}[u_1 \notin S] \cdot f(T) , \end{aligned}$$

where the first inequality holds by submodularity and the second inequality is based on the fact that  $\Pr_{S \sim \mathcal{D}}[u_i \in S]$  is a non-increasing function of  $i$ .  $\square$

## B Proof of the Tight Example for Algorithm 2

In this section we continue the proof of the tight example for Algorithm 2. Let  $f$  be the submodular function defined in Section 4.1. We analyze a possible execution of the algorithm given this function.

Let us denote the elements of  $Y$  by  $u_1, u_2, \dots, u_k$ . For notational convenience, given an index  $i > k$ , we denote by  $u_i$  the element  $u_j$  having  $i \equiv j \pmod{k}$ . Given a value  $z \in [0, 1]$ , let us characterize a distribution  $\mathcal{D}(z)$  as follows. The support of the distribution  $\mathcal{D}(z)$  contains at most  $2k$  states:

- For every  $1 \leq i \leq k$ , the state  $S_i^L = \{u_j\}_{j=i}^{i+\lfloor kz \rfloor}$  has a probability of  $z - \lfloor kz \rfloor / k$ .
- For every  $1 \leq i \leq k$ , the state  $S_i^S = \{u_j\}_{j=i}^{i+\lfloor kz \rfloor - 1}$  has a probability of  $\lfloor kz + 1 \rfloor / k - z$ .

It can be verified that all the above probabilities add up to 1, and thus,  $\mathcal{D}(z)$  is a valid distribution. Technically the above definition of  $\mathcal{D}(z)$  sometimes defines multiple identical states (for example, the states  $S_i^S$  are identical when  $z < 1/k$ ). Whenever this happens, we formally unify these states and give the unified state a probability equal to the sum of the probabilities of the unified states. In the rest of the proof we ignore that possibility for simplicity.

Intuitively,  $\mathcal{D}(z)$  is a distribution over two types of subsets of  $Y$ : cyclically continuous states of size  $1 + \lfloor kz \rfloor$  and cyclically continuous states of size  $\lfloor kz \rfloor$ . The distribution is symmetrical in the sense that all the cyclically continuous states of a given length have equal probabilities.

**Observation B.1.** *For every  $z \in [0, 1]$  and element  $u \in \mathcal{N}$ ,*

$$\Pr_{S \sim \mathcal{D}(z)}[u \in S] = \begin{cases} z & \text{if } u \in Y , \\ 0 & \text{if } u \in O . \end{cases}$$

*Proof.* It is clear that elements of  $O$  never appear in a set distributed like  $\mathcal{D}(z)$ , thus, we consider only an element  $u \in Y$ . By symmetry  $u$  appears in  $\lfloor kz \rfloor + 1$  cyclically continuous states of size  $\lfloor kz \rfloor + 1$  and  $\lfloor kz \rfloor$  cyclically continuous states of size  $\lfloor kz \rfloor$ . Thus,

$$\begin{aligned} \Pr_{S \sim \mathcal{D}(z)}[u \in S] &= (\lfloor kz \rfloor + 1) \cdot (z - \lfloor kz \rfloor / k) + (\lfloor kz \rfloor) \cdot (\lfloor kz \rfloor + 1 / k - z) \\ &= \lfloor kz \rfloor \cdot (\lfloor kz \rfloor + 1 - \lfloor kz \rfloor) / k + (z - \lfloor kz \rfloor / k) = z. \end{aligned} \quad \square$$

**Lemma B.2.** *For every  $0 \leq i \leq k$ , Algorithm 2 may set  $\mathcal{D}_i = \mathcal{D}(z_i)$ , where  $z_i = 1 - (1 - 1/k)^i$ .*

*Proof.* We prove the lemma by induction. Notice that  $\mathcal{D}(z_0)$  puts all the probability on the empty set, thus, it is identical to  $\mathcal{D}_0$ . This completes the proof of the base case. Next, assume that Algorithm 2 choosed  $\mathcal{D}_{i-1} = \mathcal{D}(z_{i-1})$  for some  $1 \leq i \leq k$ , and let us prove that it can end up with  $\mathcal{D}_i = \mathcal{D}(z_i)$ . Let us start with analyzing the marginal contributions of the various elements to a random set from  $\mathcal{D}(z_{i-1})$ .

Let  $z'_{i-1} = \lfloor kz_{i-1} \rfloor / k$ , and note that  $z'_{i-1} \leq z_{i-1} \leq z'_{i-1} + 1/k$ . Consider an arbitrary element  $u_o \in O$ . Every set  $S \in \text{supp}(\mathcal{D}_{i-1})$  contains either  $kz'_{i-1}$  or  $kz'_{i-1} + 1$  elements of  $Y$ , and thus, by submodularity, we can lower bound the expected marginal contribution of  $u_o$  to a random set of  $\mathcal{D}_{i-1}$  with its marginal contribution to a set containing  $kz'_{i-1}$  elements of  $Y$ . By (3) we now get:

$$\begin{aligned} \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[f(u_o | S)] &\leq \frac{1}{k} \left( (1 - z'_{i-1}) - \left( g(z'_{i-1}) + \frac{\ell z'_{i-1}}{k} \right) \right) \\ &\leq \frac{1}{k} \left( (1 - z'_{i-1}) - (z'_{i-1} - 1) \ln(1 - z'_{i-1}) \right) = \frac{1}{k} (1 - z'_{i-1}) \cdot (1 + \ln(1 - z'_{i-1})) . \end{aligned}$$

On the other hand, consider an element  $u_y \in Y$ . A random set from  $\mathcal{D}_{i-1}$  contains  $u_y$  with probability  $z_{i-1}$ , in which case the marginal contribution of  $u_y$  is 0. Every other set  $S$  in the distribution contains either  $kz'_{i-1}$  or  $kz'_{i-1} + 1$  elements of  $Y$ , and thus, by submodularity, we can upper bound the expected marginal contribution of  $u_y$  to such a set with its marginal contribution to a set containing  $kz'_{i-1} + 1$  elements of  $y$ . By (2) we now get:

$$\begin{aligned} \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[f(u_y | S)] &= (1 - z_{i-1}) \cdot \left( g(z'_{i-1} + 2/k) - g(z'_{i-1} + 1/k) + \frac{\ell}{k^2} \right) \\ &\geq \frac{1}{k} \cdot (1 - z_{i-1}) \cdot \left( g'(z'_{i-1} + 2/k) + \frac{\ell}{k} \right) \\ &\geq \frac{1}{k} \cdot (1 - z'_{i-1} - 1/k) \cdot \left( 1 + \ln(1 - z'_{i-1} - 2/k) + \frac{\ell}{k} \right) , \end{aligned} \quad (4)$$

where Inequality (4) follows by the concavity of  $g$ . Using the two inequalities we get,

$$\begin{aligned} &\mathbb{E}_{S \sim \mathcal{D}_{i-1}}[f(u_y | S)] - \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[f(u_o | S)] \\ &\geq \frac{1}{k} \cdot (1 - z'_{i-1}) \left( \ln \left( 1 - \frac{2}{k(1 - z'_{i-1})} \right) + \frac{\ell}{k} \right) - \frac{1}{k^2} \cdot \left( 1 + \ln(1 - z'_{i-1} - 2/k) + \frac{\ell}{k} \right) \\ &\geq \frac{1}{e \cdot k} \left( \ln \left( 1 - \frac{2e}{k} \right) + \frac{\ell}{k} \right) - \frac{2}{k^2} \end{aligned} \quad (5)$$

$$\geq \frac{1}{e \cdot k} \left( -\frac{2e}{k(1 - \frac{2e}{k})} + \frac{\ell}{k} \right) - \frac{2}{k^2} \quad (6)$$

$$\geq \frac{1}{e \cdot k} \left( -\frac{4e}{k} + \frac{\ell}{k} \right) - \frac{2}{k^2} \geq 0 , \quad (7)$$



where Inequality (5) follows for a large enough  $\ell$  ( $\geq 4e$ ) since  $k \geq \ell$  by our assumption and  $0 \leq z'_{i-1} \leq 1 - e^{-1}$ . Inequality (6) follows by the inequality  $\ln(1 - y) \geq -\frac{y}{1-y}$ , which holds for  $y \in [0, 1)$ . Finally, Inequality (7) and the last inequality both follow by considering a large enough  $\ell$  ( $\geq 6e$ ) and recalling that  $k \geq \ell$ .

Since the last inequality holds for every pair of elements  $u_o \in O$  and  $u_y \in Y$ , it implies that the set  $M_i$  chosen by the algorithm is exactly  $Y$ . Given this observation, the formulation (P) of Algorithm 2 in iteration  $i$  becomes:

$$(P) \quad \begin{aligned} \max \quad & \sum_{u \in Y} \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[x(u, S) \cdot f(u \mid S)] \\ & \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[x(u, S)] \leq 1/k \cdot (1 - z_{i-1}) \quad \forall u \in Y \\ & \sum_{u \in M_i} x(u, S) + \ell(S) = 1 \quad \forall S \in \text{supp}(\mathcal{D}_{i-1}) \\ & x(u, S), \ell(S) \geq 0 \quad \forall u \in M_i, S \in \text{supp}(\mathcal{D}_{i-1}) \end{aligned}$$

We need to show that there exists an optimal extreme point solution for this formulation which makes the algorithm set  $\mathcal{D}_i = \mathcal{D}(z_i)$ . There are two cases to consider. If  $\mathcal{D}(z_{i-1})$  and  $\mathcal{D}(z_i)$  have the same states (*i.e.*,  $\lfloor kz_{i-1} \rfloor = \lfloor kz_i \rfloor$ ), then Algorithm 2 can come up with a solution  $x^*$  for (P) assigning  $x(u_{j+\lfloor kz_i \rfloor}, S_j^S) = (z_i - z_{i-1}) / \Pr_{\mathcal{D}_{i-1}}[S_j^S]$  for every  $1 \leq j \leq k$  and the value 0 to the other  $x$  variables (the values of the  $\ell$  variables are induced by the values of the  $x$  variables, and thus, we do not state their assignment). The solution  $x^*$  is feasible since, for every  $1 \leq j \leq k$ :

$$\begin{aligned} \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[x(u_j, S)] &= \Pr_{\mathcal{D}_{i-1}}[S_j^S] \cdot \frac{z_i - z_{i-1}}{\Pr_{\mathcal{D}_{i-1}}[S_j^S]} = [1 - (1 - 1/k)^i] - [1 - (1 - 1/k)^{i-1}] \\ &= (1 - 1/k)^{i-1} [1 - (1 - 1/k)] = 1/k \cdot (1 - z_{i-1}) . \end{aligned}$$

It can be checked that  $x^*$  indeed leads Algorithm 2 to set  $\mathcal{D}_i = \mathcal{D}(z_i)$ . To see that  $x^*$  is optimal, notice that it adds elements only to the smaller sets (which results in a larger marginal gain by submodularity), and it adds every element to the maximal extent allowed by the first type of constraints. Finally, to see that  $x^*$  is an extreme point solution notice that it is the only solution maximizing the objective function  $c \cdot x$ , where  $c$  is a vector taking the value 1 exactly in the coordinates for which  $x^*$  is non-zero.

The second case we need to consider is when  $\mathcal{D}(z_{i-1})$  and  $\mathcal{D}(z_i)$  have different states (*i.e.*,  $1 + \lfloor kz_{i-1} \rfloor = \lfloor kz_i \rfloor$ ). In this case Algorithm 2 can come up with a solution  $x^*$  for (P) assigning  $x(u_{j+\lfloor kz_{i-1} \rfloor}, S_j^S) = 1$  and  $x(u_{j+\lfloor kz_i \rfloor}, S_j^L) = k^{-1}(kz_{i-1} - z_{i-1} - \lfloor kz_{i-1} \rfloor) / \Pr[S^L]$  for every  $1 \leq j \leq k$  and the value 0 to the other  $x$  variables. The solution  $x^*$  is feasible since, for every  $1 \leq j \leq k$ :

$$\begin{aligned} \mathbb{E}_{S \sim \mathcal{D}_{i-1}}[x(u_j, S)] &= \Pr_{\mathcal{D}_{i-1}}[S_j^S] + \Pr_{\mathcal{D}_{i-1}}[S_j^L] \cdot \frac{kz_{i-1} - z_{i-1} - \lfloor kz_{i-1} \rfloor}{k \cdot \Pr_{\mathcal{D}_{i-1}}[S_j^L]} \\ &= \lfloor kz_{i-1} + 1 \rfloor / k - z_{i-1} + \frac{kz_{i-1} - z_{i-1} - \lfloor kz_{i-1} \rfloor}{k} = 1/k \cdot (1 - z_{i-1}) . \end{aligned}$$

It can be checked that  $x^*$  again leads Algorithm 2 to set  $\mathcal{D}_i = \mathcal{D}(z_i)$ . To see that  $x^*$  is optimal, notice that it adds as much as possible elements to the smaller sets (which results in a larger marginal gain by submodularity), and only the remaining capacity given by the first type of constraints is used to add elements to the larger sets. Finally, to see that  $x^*$  is an extreme point solution notice that it is the only solution maximizing the objective function  $c \cdot x$ , where  $c$  is a vector taking the value  $k + 1$  in the coordinates for which  $x^*$  is 1 and the value 1 in the other coordinates for which  $x^*$  is non-zero.  $\square$

To complete the analysis of our bad instance, notice that  $O$  is a feasible solution and  $f(O) = 1$ . On the other hand, Lemma B.2 shows that Algorithm 2 may terminate with a distribution over

subsets of  $Y$ . The value of  $f$  for any such set  $S$  is at most:

$$f(S) = g\left(\frac{|S|}{k}\right) + \frac{\ell \cdot |S|}{k^2} \leq e^{-1} + \frac{\ell}{k} = e^{-1} + O\left(\frac{1}{k}\right) .$$