A Tight Linear Time $(1/2)$-Approximation for Unconstrained Submodular Maximization

Niv Buchbinder[†]
Statistics and Operations Research Dept.
Tel Aviv University
Tel Aviv, Israel
niv.buchbinder@gmail.com

Moran Feldman
Computer Science Dept.
Technion
Haifa, Israel
moranfe@cs.technion.ac.il

Joseph (Seffi) Naor[*]
Computer Science Dept.
Technion
Haifa, Israel
naor@cs.technion.ac.il

Roy Schwartz
Microsoft Research
Redmond, WA
roysch@microsoft.com

***Abstract*—We consider the Unconstrained Submodular Maximization problem in which we are given a non-negative submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, and the objective is to find a subset $S \subseteq \mathcal{N}$ maximizing $f(S)$. This is one of the most basic submodular optimization problems, having a wide range of applications. Some well known problems captured by Unconstrained Submodular Maximization include Max-Cut, Max-DiCut, and variants of Max-SAT and maximum facility location. We present a simple randomized linear time algorithm achieving a tight approximation guarantee of $1/2$, thus matching the known hardness result of Feige et al. [11]. Our algorithm is based on an adaptation of the greedy approach which exploits certain symmetry properties of the problem. Our method might seem counterintuitive, since it is known that the greedy algorithm fails to achieve any bounded approximation factor for the problem.**

***Keywords*-Submodular Functions, Approximation Algorithms**

I. Introduction

The study of combinatorial problems with submodular objective functions has recently attracted much attention, and is motivated by the principle of economy of scale, prevalent in real world applications. Submodular functions are also commonly used as utility functions in economics and algorithmic game theory. Submodular functions and submodular maximization play a major role in combinatorial optimization, where several well known examples of submodular functions include cuts in graphs and hypergraphs, rank functions of matroids, and covering functions. A function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$ is called *submodular* if for every $A \subseteq B \subseteq \mathcal{N}$ and $u \in \mathcal{N}$, $f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B)$[1]. Submodularity captures the principle of *diminishing returns* in economics.

Perhaps the most basic submodular maximization problem is Unconstrained Submodular Maximization (USM). Given a non-negative submodular fucntion f, the goal is to find a subset $S \subseteq \mathcal{N}$ maximizing $f(S)$. Note that there is no restriction on the choice of S, as any subset of $\mathcal{N}$ is a feasible solution. USM captures many well studied problems such as Max-Cut, Max-DiCut [15], [20], [22], [25], [27], [36], variants of Max-SAT, and maximum facility location [1], [7], [8]. Moreover, USM has various applications in more practical settings such as marketing in social networks [21], revenue maximization with discrete choice [2], and algorithmic game theory [10], [34].

USM has been studied starting since the sixties in the operations research community [2], [6], [16], [17], [18], [26], [29], [33]. Not surprisingly, as USM captures NP-hard problems, all these works provide algorithms that either solve special cases of the problem, or provide exact algorithms whose time complexity cannot be efficiently bounded, or provide efficient algorithms whose output has no provable guarantee.

The first rigorous study of the USM problem was conducted by Feige et al. [11], who provided several constant approximation factor algorithms. They proved that a subset S chosen uniformly at random is a $(1/4)$-approximation. Additionally, they also described two local search algorithms. The first uses f as the objective function, and provides an approximation of $1/3$. The second uses a noisy version of f as the objective function, and achieves an improved approximation guarantee of $2/5$. Gharan and Vondrák [14] showed that an extension of the last method, known as *simulated annealing*, can provide an improved approximation of roughly 0.41. Their algorithm, like that of Feige et al. [11], uses local search with a noisy objective function. However, in [14] the noise decreases as the algorithm advances, as opposed to being constant as in [11]. Feldman et al. [12] observed that if the simulated annealing algorithm of [14] outputs a relatively poor solution, then it must generate at some point a set S which is *structurally similar* to some optimal solution. Moreover, they showed that this structural similarity can be traded for value, providing an overall improved approximation of roughly 0.42.

It is important to note that for many special cases of USM, better approximation factors are known. For example, the

[†]Research supported in part by ISF grant 954/11, and by BSF grant 2010426.

[*]Research supported in part by the Google Inter-university center for Electronic Markets, by ISF grant 954/11, and by BSF grant 2010426.

[1]An equivalent definition is that for every $A, B \subseteq \mathcal{N}$: $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$.

seminal work of Goemans and Williamson [15] provides an 0.878-approximation for Max-Cut based on a semidefinite programming formulation, and Ageev and Sviridenko [1] provide an approximation of 0.828 for the maximum facility location problem.

On the negative side, Feige et al. [11] studied the hardness of USM assuming the function f is given via a *value oracle*[2]. They proved that for any constant $\varepsilon > 0$, any algorithm achieving an approximation of $(1/2 + \varepsilon)$ requires an exponential number of oracle queries. This hardness result holds even if f is symmetric, in which case it is known to be tight [11]. Recently, Dobzinski and Vondrák [9] proved that even if f has a compact representation (which is part of the input), the latter hardness still holds assuming $RP \neq NP$.

A. Our Results

In this paper we resolve the approximability of the Unconstrained Submodular Maximization problem. We design a tight linear time randomized $(1/2)$-approximation for the problem. We begin by presenting a simple greedy-based deterministic algorithm that achieves a $(1/3)$-approximation for USM.

Theorem I.1. *There exists a deterministic linear time $(1/3)$-approximation algorithm for the Unconstrained Submodular Maximization problem.*

We show that our analysis of the algorithm is tight by providing an instance for which the algorithm achieves an approximation of $1/3 + \varepsilon$ for an arbitrary small $\varepsilon > 0$. The improvement in the performance is achieved by incorporating randomness into the choices of the algorithm. We obtain a tight bound for USM without increasing the time complexity of the algorithm.

Theorem I.2. *There exists a randomized linear time $(1/2)$-approximation algorithm for the Unconstrained Submodular Maximization problem.*

In both Theorems I.1 and I.2 we assume that a single query to the value oracle takes $O(1)$ time. Thus, a linear time algorithm is an algorithm which makes $O(n)$ oracle queries plus $O(n)$ other operations, where n is the size of the ground set $\mathcal{N}$.

Building on the above two theorems, we provide two additional approximation algorithms for Submodular Max-SAT and Submodular Welfare with 2 players (for the exact definition of these problems please refer to Section IV). Both problems are already known to have a tight approximation [13]. However, our algorithms run in linear time, thus, significantly improving the time complexity, while achieving the same performance guarantee.

Theorem I.3. *There exists a randomized linear time $(3/4)$-approximation algorithm for the Submodular Max-SAT problem.*

Theorem I.4. *There exists a randomized linear time $(3/4)$-approximation algorithm for the Submodular Welfare problem with 2 players .*

B. Techniques

The algorithms we present are based on a greedy approach. It is known that a straightforward greedy approach fails for USM. In contrast, Feldman et al. [13] recently showed that a continuous greedy approach does provide a $(1/e)$-approximation for maximizing any non-monotone submodular function over a matroid. Recall that better bounds than $1/e$ are already known for USM.

To understand the main ideas of our algorithm, consider a non-negative submodular function f. Let us examine the complement of f, denoted by $\bar{f}$, defined as $\bar{f}(S) \triangleq f(\mathcal{N} \setminus S)$, for any $S \subseteq \mathcal{N}$. Note that since f is submodular, $\bar{f}$ is also submodular. Additionally, given an optimal solution $OPT \subseteq \mathcal{N}$ for USM with input f, $\mathcal{N} \setminus OPT$ is an optimal solution for $\bar{f}$, and both solutions have the exact same value. Consider, for example, the greedy algorithm. For f, it starts from an empty solution and iteratively adds elements to it in a greedy fashion. However, when applying the greedy algorithm to $\bar{f}$, one gets an algorithm for f that effectively starts with the solution $\mathcal{N}$ and iteratively removes elements from it. Both algorithms are equally reasonable, but, unfortunately, both fail.

Despite the failure of the greedy algorithm when applied separately to either f or $\bar{f}$, we show that a correlated execution on both f and $\bar{f}$ provides a much better result. That is, we start with two solutions $\emptyset$ and $\mathcal{N}$. The algorithm considers the elements (in arbitrary order) one at a time. For each element it determines whether it should be added to the first solution or removed from the second solution. Thus, after a single pass over the ground set $\mathcal{N}$, both solutions completely coincide. This is the solution that the algorithm outputs. We show that a greedy choice in each step obtains an approximation guarantee of $1/3$, hence proving Theorem I.1. To get Theorem I.2, we use a natural extension of the greedy rule, together with randomization, thus improving the approximation guarantee to $1/2$.

C. Related Work

Many works consider the problem of maximizing a non-negative submodular function subject to various combinatorial constraints defining the feasible solutions [13], [19], [30], [31], [39], or minimizing a submodular function subject to such constraints [23], [24], [35]. Interestingly, the problem

[2]The explicit representation of a submodular function might be exponential in the size of its ground set. Thus, it is standard to assume that the function is accessed via a value oracle. For a submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, given a set $S \subseteq \mathcal{N}$, the value oracle returns the value of $f(S)$.

of unconstrained submodular minimization can be solved in polynomial time [32].

Another line of work deals with maximizing normalized *monotone* submodular functions, again, subject to various combinatorial constraints. A continuous greedy algorithm was given by Calinescu et al. [3] for maximizing a normalized monotone submodular function subject to a matroid constraint. Later, Lee et al. [31] gave a local search algorithm achieving $1/p-\varepsilon$ approximation for maximizing such functions subject to the intersection of p matroids. Kulik et al. [28] showed a $1-1/e-\varepsilon$ approximation algorithm for maximizing a normalized monotone submodular function subject to multiple knapsack constraints. Recently, Chekuri et al. [5] and Feldman et al. [13] gave non-monotone counterparts of the continuous greedy algorithm of [3], [38]. These results improve several non-monotone submodular optimization problems. Some of the above results were generalized by Chekuri et al. [4], who provide a dependent rounding technique for various polytopes, including matroid and matroid-intersection polytops. The advantage of this rounding technique is that it guarantees strong concentration bounds for submodular functions. Additionally, Chekuri et al. [5] define a contention resolution rounding scheme which allows one to obtain approximations for different combinations of constraints.

II. A Deterministic $(1/3)$-Approximation Algorithm

In this section we present a deterministic linear time algorithm for USM. The algorithm proceeds in n iterations that correspond to some arbitrary order $u_1, \ldots, u_n$ of the ground set $\mathcal{N}$. The algorithm maintains two solutions X and Y. Initially, we set the solutions to $X_0 = \emptyset$ and $Y_0 = \mathcal{N}$. In the ith iteration the algorithm either adds u_i to X_{i-1} or removes u_i from Y_{i-1}. This decision is done greedily based on the marginal gain of each of the two options. Eventually, after n iterations both solutions coincide, and we get $X_n = Y_n$; this is the output of the algorithm. A formal description of the algorithm appears as Algorithm 1.

```
Algorithm 1: DeterministicUSM($f, \mathcal{N}$)
1 $X_0 \leftarrow \emptyset$, $Y_0 \leftarrow \mathcal{N}$.
2 for $i$ = 1 to $n$ do
3     $a_i \leftarrow f(X_{i-1} \cup \{u_i\}) - f(X_{i-1})$.
4     $b_i \leftarrow f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1})$.
5     if $a_i \geq b_i$ then $X_i \leftarrow X_{i-1} \cup \{u_i\}$, $Y_i \leftarrow Y_{i-1}$.
6     else $X_i \leftarrow X_{i-1}$, $Y_i \leftarrow Y_{i-1} \setminus \{u_i\}$.
7 return $X_n$ (or equivalently $Y_n$).
```

The rest of this section is devoted for proving Theorem I.1, *i.e.*, we prove that the approximation guarantee of Algorithm 1 is $1/3$. Denote by a_i the change in value of the first solution if element u_i is added to it in the ith iteration, *i.e.*, $f(X_{i-1} \cup \{u_i\}) - f(X_{i-1})$. Similarly, denote by b_i the change in value of the second solution if element u_i is removed from it in the ith iteration, *i.e.*, $f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1})$. We start with the following useful lemma.

Lemma II.1. *For every* $1 \leq i \leq n$, $a_i + b_i \geq 0$.

Proof: Notice that $(X_{i-1} \cup \{u_i\}) \cup (Y_i \setminus \{u_i\}) = Y_{i-1}$ and $(X_{i-1} \cup \{u_i\}) \cap (Y_i \setminus \{u_i\}) = X_{i-1}$. By combining both observations with submodularity, one gets:

$$\begin{aligned} a_i + b_i &= [f(X_{i-1} \cup \{u_i\}) - f(X_{i-1})] + \\ &\quad [f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1})] \\ &= [f(X_{i-1} \cup \{u_i\}) + f(Y_{i-1} \setminus \{u_i\})] - \\ &\quad [f(X_{i-1}) + f(Y_{i-1})] \geq 0 \ . \end{aligned}$$

■

Let OPT denote an optimal solution. Define $OPT_i \triangleq (OPT \cup X_i) \cap Y_i$. Thus, OPT_i coincides with X_i and Y_i on elements $1, \ldots, i$, and it coincides with OPT on elements $i+1, \ldots, n$. Note that $OPT_0 = OPT$ and the output of the algorithm is $OPT_n = X_n = Y_n$. Examine the sequence $f(OPT_0), \ldots, f(OPT_n)$, which starts with $f(OPT)$ and ends with the value of the output of the algorithm. The main idea of the proof is to bound the total loss of value along this sequence. This goal is achieved by the following lemma which upper bounds the loss in value between every two consecutive steps in the sequence. Formally, the loss of value, *i.e.*, $f(OPT_{i-1}) - f(OPT_i)$, is no more than the *total* increase in value of both solutions maintained by the algorithm, *i.e.*, $[f(X_i) - f(X_{i-1})] + [f(Y_i) - f(Y_{i-1})]$.

Lemma II.2. *For every* $1 \leq i \leq n$,

$$f(OPT_{i-1}) - f(OPT_i) \leq [f(X_i) - f(X_{i-1})] + [f(Y_i) - f(Y_{i-1})] \ .$$

Before proving Lemma II.2, let us show that Theorem I.1 follows from it.

Proof of Theorem I.1: Summing up Lemma II.2 for every $1 \leq i \leq n$ gives:

$$\sum_{i=1}^{n} [f(OPT_{i-1}) - f(OPT_i)] \leq \sum_{i=1}^{n} [f(X_i) - f(X_{i-1})] + \sum_{i=1}^{n} [f(Y_i) - f(Y_{i-1})] \ .$$

The above sum is telescopic. Collapsing it, we get:

$$f(OPT_0) - f(OPT_n) \leq [f(X_n) - f(X_0)] + [f(Y_n) - f(Y_0)] \leq f(X_n) + f(Y_n) \ .$$

Recalling the definitions of OPT_0 and OPT_n, we obtain that $f(X_n) = f(Y_n) \geq f(OPT)/3$. ■

It is left to prove Lemma II.2.

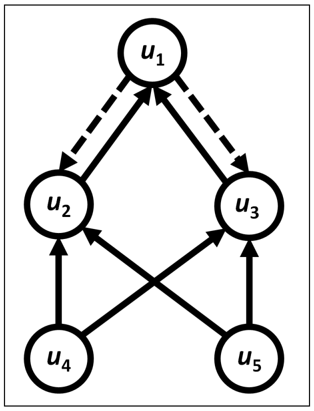

Figure 1. Tight example for Algorithm 1. The weight of the dashed edges is $1 - \varepsilon$. All other edges have weight of 1.

Proof of Lemma II.2: Assume without loss of generality that $a_i \geq b_i$, *i.e.*, $X_i \leftarrow X_{i-1} \cup \{u_i\}$ and $Y_i \leftarrow Y_{i-1}$ (the other case is analogous). Notice that in this case $OPT_i = (OPT \cup X_i) \cap Y_i = OPT_{i-1} \cup \{u_i\}$ and $Y_i = Y_{i-1}$. Thus the inequality that we need to prove becomes:

$$f(OPT_{i-1}) - f(OPT_{i-1} \cup \{u_i\}) \leq f(X_i) - f(X_{i-1}) = a_i \ .$$

We now consider two cases. If $u_i \in OPT$, then the left hand side of the last inequality is 0, and all we need to show is that a_i is non-negative. This is true since $a_i + b_i \geq 0$ by Lemma II.1, and $a_i \geq b_i$ by our assumption.

If $u_i \notin OPT$, then also $u_i \notin OPT_{i-1}$, and thus:

$$f(OPT_{i-1}) - f(OPT_{i-1} \cup \{u_i\}) \leq \\ f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1}) = b_i \leq a_i \ .$$

The first inequality follows by submodularity: $OPT_{i-1} = ((OPT \cup X_{i-1}) \cap Y_{i-1}) \subseteq Y_{i-1} \setminus \{u_i\}$ (recall that $u_i \in Y_{i-1}$ and $u_i \notin OPT_{i-1}$). The second inequality follows from our assumption that $a_i \geq b_i$. ■

A. Tight Example

We now show that the analysis of Algorithm 1 is tight.

Theorem II.3. *For an arbitrarily small constant $\varepsilon > 0$, there exists a submodular function for which Algorithm 1 provides only $(1/3 + \varepsilon)$-approximation.*

Proof: The proof follows by analyzing Algorithm 1 on the cut function of the weighted digraph given in Figure 1. The maximum weight cut in this digraph is $\{u_1, u_4, u_5\}$. This cut has weight of $6 - 2\varepsilon$. We claim that Algorithm 1 outputs the cut $\{u_2, u_3, u_4, u_5\}$, whose value is only 2 (assuming the nodes of the graph are considered at the given order). Hence, the approximation guarantee of Algorithm 1 on the above instance is:

$$\frac{2}{6 - 2\varepsilon} = \frac{1}{3 - \varepsilon} \leq \frac{1}{3} + \varepsilon \ .$$

Let us track the execution of the algorithm. Let X_i, Y_i be the solutions maintained by the algorithm. Initially $X_0 = \emptyset, Y_0 = \{u_1, u_2, u_3, u_4, u_5\}$. Note that in case of a tie ($a_i = b_i$), Algorithm 1 takes node u_i.

1) In the first iteration the algorithm considers u_1. Adding this node to X_0 increases the value of this solution by $2 - 2\varepsilon$. On the other hand, removing this node from Y_0 increases the value of Y_0 by 2. Hence, $X_1 \leftarrow X_0, Y_1 \leftarrow Y_0 \setminus \{u_1\}$.
2) Let us inspect the next two iterations in which the algorithm considers u_2, u_3. One can easily verify that these two iterations are independent, hence, we consider only u_2. Adding u_2 to X_1 increases its value by 1. On the other hand, removing u_2 from $Y_1 = \{u_2, u_3, u_4, u_5\}$ also increases the value of Y_1 by 1. Thus, the algorithm adds u_2 to X_1. Since u_2 and u_3 are symmetric, the algorithm also adds u_3 to X_1. Thus, at the end of these two iterations $X_3 = X_1 \cup \{u_2, u_3\} = \{u_2, u_3\}, Y_3 = \{u_2, u_3, u_4, u_5\}$.
3) Finally, the algorithm considers u_4 and u_5. These two iterations are also independent so we consider only u_4. Adding u_4 to X_3 does not increase the value of X_3. Also removing u_4 from Y_3 does not increase the value of Y_3. Thus, the algorithm adds u_4 to X_3. Since u_4 and u_5 are symmetric, the algorithm also adds u_5 to X_3. Thus, we get $X_5 = Y_5 = \{u_2, u_3, u_4, u_5\}$.

■

III. A Randomized (1/2)-Approximation Algorithm

In this section we present a randomized algorithm achieving a tight $(1/2)$-approximation for USM. Algorithm 1 compared the marginal profits a_i and b_i, and based on this comparison it made a greedy deterministic decision whether to include or exclude u_i from its output. The random algorithm we next present makes a "smoother" decision, based on the values a_i and b_i. In each step, it randomly chooses whether to include or exclude u_i from the output with probability derived from the values a_i and b_i. A formal description of the algorithm appears as Algorithm 2.

The rest of this section is devoted to proving that Algorithm 2 provides an approximation guarantee of $1/2$ to USM. In Appendix A, we describe another algorithm which can be thought of as the continuous counterpart of Algorithm 2. This algorithm achieves the same approximation ratio (up to low order terms), but keeps a fractional inner state.

Let us first introduce some notation. Notice that for every $1 \leq i \leq n$, X_i and Y_i are random variables denoting the sets of elements in the two solutions generated by the algorithm at the end of the ith iteration. As in Section II, let us define the following random variables: $OPT_i \triangleq (OPT \cup X_i) \cap Y_i$. Note that as before, $X_0 = \emptyset$, $Y_0 = \mathcal{N}$ and $OPT_0 = OPT$. Moreover, the following

Algorithm 2: RandomizedUSM$(f, \mathcal{N})$

```
1 $X_0 \leftarrow \emptyset$, $Y_0 \leftarrow \mathcal{N}$.
2 for $i$ = 1 to $n$ do
3     $a_i \leftarrow f(X_{i-1} \cup \{u_i\}) - f(X_{i-1})$.
4     $b_i \leftarrow f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1})$.
5     $a'_i \leftarrow \max\{a_i, 0\}$, $b'_i \leftarrow \max\{b_i, 0\}$.
6     with probability $a'_i/(a'_i + b'_i)^*$ do:
      $X_i \leftarrow X_{i-1} \cup \{u_i\}$, $Y_i \leftarrow Y_{i-1}$.
7     else (with the compliment probability $b'_i/(a'_i + b'_i)$)
      do: $X_i \leftarrow X_{i-1}$, $Y_i \leftarrow Y_{i-1} \setminus \{u_i\}$.
8 return $X_n$ (or equivalently $Y_n$).
```

* If $a'_i = b'_i = 0$, we assume $a'_i/(a'_i + b'_i) = 1$.

always holds: $OPT_n = X_n = Y_n$. The proof idea is similar to that of the deterministic algorithm in Section II when considering the sequence $\mathbb{E}[f(OPT_0)], \ldots, \mathbb{E}[f(OPT_n)]$. This sequence starts with $f(OPT)$ and ends with the expected value of the algorithm's output. The following lemma upper bounds the loss of two consecutive elements in the sequence. Formally, $\mathbb{E}[f(OPT_{i-1}) - f(OPT_i)]$ is upper bounded by the *average* expected change in the value of the two solutions maintained by the algorithm, *i.e.*, $1/2 \cdot \mathbb{E}\left[(f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})\right]$.

Lemma III.1. *For every $1 \leq i \leq n$,*

$$\mathbb{E}\left[f(OPT_{i-1}) - f(OPT_i)\right] \leq \frac{1}{2} \cdot \mathbb{E}\left[(f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})\right] \quad , \tag{1}$$

where expectations are taken over the random choices of the algorithm.

Before proving Lemma III.1, let us show that Theorem I.2 follows from it.

Proof of Theorem I.2: Summing up Lemma III.1 for every $1 \leq i \leq n$ yields:

$$\sum_{i=1}^{n} \mathbb{E}\left[f(OPT_{i-1}) - f(OPT_i)\right] \leq \frac{1}{2} \cdot \sum_{i=1}^{n} \mathbb{E}\left[f(X_i) - F(X_{i-1}) + f(Y_i) - F(Y_{i-1})\right] \quad .$$

The above sum is telescopic. Collapsing it, we get:

$$\mathbb{E}\left[f(OPT_0) - f(OPT_n)\right] \leq \frac{1}{2} \cdot \mathbb{E}\left[f(X_n) - f(X_0) + f(Y_n) - f(Y_0)\right] \leq \frac{\mathbb{E}[f(X_n) + f(Y_n)]}{2} \quad .$$

Recalling the definitions of OPT_0 and OPT_n, we obtain that $\mathbb{E}[f(X_n)] = \mathbb{E}[f(Y_n)] \geq f(OPT)/2$. ■

It is left to prove Lemma III.1.

Proof of Lemma III.1: Notice that it suffices to prove Inequality (1) conditioned on any event of the form $X_{i-1} = S_{i-1}$, when $S_{i-1} \subseteq \{u_1, \ldots, u_{i-1}\}$ and the probability that $X_{i-1} = S_{i-1}$ is non-zero. Hence, fix such an event for a given S_{i-1}. The rest of the proof implicitly assumes everything is conditioned on this event. Notice that due to the conditioning, the following quantities become constants:

- $Y_{i-1} = S_{i-1} \cup \{u_i, \ldots, u_n\}$.
- $OPT_{i-1} \triangleq (OPT \cup X_{i-1}) \cap Y_{i-1} = S_{i-1} \cup (OPT \cap \{u_i, \ldots, u_n\})$.
- a_i and b_i.

Moreover, by Lemma II.1, $a_i + b_i \geq 0$. Thus, it cannot be that both a_i, b_i are strictly less than zero. Hence, we only need to consider the following 3 cases:

Case 1 ($\mathbf{a_i} \geq \mathbf{0}$ and $\mathbf{b_i} \leq \mathbf{0}$): In this case $a'_i/(a'_i + b'_i) = 1$, and so the following always happen: $Y_i = Y_{i-1} = S_{i-1} \cup \{u_i, \ldots, u_n\}$ and $X_i \leftarrow S_{i-1} \cup \{u_i\}$. Hence, $f(Y_i) - f(Y_{i-1}) = 0$. Also, by our definition $OPT_i = (OPT \cup X_i) \cap Y_i = OPT_{i-1} \cup \{u_i\}$. Thus, we are left to prove that:

$$f(OPT_{i-1}) - f(OPT_{i-1} \cup \{u_i\}) \leq \frac{1}{2} \cdot [f(X_i) - f(X_{i-1})] = \frac{a_i}{2} \quad .$$

If $u_i \in OPT$, then the left hand side of the last expression is 0, which is clearly no larger than the non-negative $a_i/2$. If $u_i \notin OPT$, then:

$$f(OPT_{i-1}) - f(OPT_{i-1} \cup \{u_i\}) \leq f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1}) = b_i \leq 0 \leq a_i/2 \quad .$$

The first inequality follows from submodularity since $OPT_{i-1} \triangleq (OPT \cup X_{i-1}) \cap Y_{i-1} \subseteq Y_{i-1} \setminus \{u_i\}$ (note that $u_i \in Y_{i-1}$ and $u_i \notin OPT_{i-1}$).

Case 2 ($\mathbf{a_i} < \mathbf{0}$ and $\mathbf{b_i} \geq \mathbf{0}$): This case is analogous to the previous one, and therefore, we omit its proof.

Case 3 ($\mathbf{a_i} \geq \mathbf{0}$ and $\mathbf{b_i} > \mathbf{0}$): In this case $a'_i = a_i, b'_i = b_i$. Therefore, with probability $a_i/(a_i + b_i)$ the following events happen: $X_i \leftarrow X_{i-1} \cup \{u_i\}$ and $Y_i \leftarrow Y_{i-1}$, and with probability $b_i/(a_i + b_i)$ the following events happen: $X_i \leftarrow X_{i-1}$ and $Y_i \leftarrow Y_{i-1} \setminus \{u_i\}$. Thus,

$$\mathbb{E}[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})] = \frac{a_i}{a_i + b_i} \cdot [f(X_{i-1} \cup \{u_i\}) - f(X_{i-1})] + \frac{b_i}{a_i + b_i} \cdot [f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1})] = \frac{a_i^2 + b_i^2}{a_i + b_i} \quad . \tag{2}$$

Next, we upper bound $\mathbb{E}[f(OPT_{i-1}) - f(OPT_i)]$. As $OPT_i = (OPT \cup X_i) \cap Y_i$, we get,

$$\begin{aligned}&\mathbb{E}[f(OPT_{i-1}) - f(OPT_i)] = \\ &\quad \frac{a_i}{a_i+b_i} \cdot [f(OPT_{i-1}) - f(OPT_{i-1} \cup \{u_i\})] + \\ &\quad \frac{b_i}{a_i+b_i} \cdot [f(OPT_{i-1}) - f(OPT_{i-1} \setminus \{u_i\})] \\ &\le \frac{a_ib_i}{a_i+b_i} \quad . \qquad (3)\end{aligned}$$

The final inequality follows by considering two cases. Note first that $u_i \in Y_{i-1}$ and $u_i \notin X_{i-1}$. If $u_i \notin OPT_{i-1}$ then the second term of the left hand side of the last inequality equals zero. Moreover, $OPT_{i-1} \triangleq (OPT \cup X_{i-1}) \cap Y_{i-1} \subseteq Y_{i-1} \setminus \{u_i\}$, and therefore, by submodularity,

$$\begin{aligned}f(OPT_{i-1}) - f(OPT_{i-1} \cup \{u_i\}) &\le \\ f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1}) &= b_i \quad .\end{aligned}$$

If $u_i \in OPT_{i-1}$ then the first term of the left hand side of the inequality equals zero, and we also have $X_{i-1} \subseteq ((OPT \cup X_{i-1}) \cap Y_{i-1}) \setminus \{u_i\} = OPT_{i-1} \setminus \{u_i\}$. Thus, by submodularity,

$$\begin{aligned}f(OPT_{i-1}) - f(OPT_{i-1} \setminus \{u_i\}) &\le \\ f(X_{i-1} \cup \{u_i\}) - f(X_{i-1}) &= a_i \quad .\end{aligned}$$

Plugging (2) and (3) into Inequality (1), we get the following:

$$\frac{a_ib_i}{a_i+b_i} \le \frac{1}{2} \cdot \left(\frac{a_i^2+b_i^2}{a_i+b_i}\right) \quad ,$$

which can be easily verified. ■

IV. Approximation Algorithms for Submodular Max-SAT and Submodular Welfare with 2 Players

In this section we build upon ideas from the previous sections to obtain linear time tight (3/4)-approximation algorithms for both the Submodular Max-SAT (SSAT) and Submodular Welfare (SW) with 2 players problems. In contrast to USM, tight approximations are already known for the above two problems [13]. However, the algorithms we present in this work, in addition to providing tight approximations, also run in linear time. We require the definition of a monotone submodular function: a submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$ is monotone if for every $A \subseteq B \subseteq \mathcal{N}$, $f(A) \le f(B)$. Moreover, a monotone submodular function f is normalized if $f(\emptyset) = 0$.

A. A (3/4)-Approximation for Submodular Max-SAT

In the Submodular Max-SAT problem we are given a CNF formula Ψ with a set $\mathcal{C}$ of clauses over a set $\mathcal{N}$ of variables, and a normalized monotone submodular function $f : 2^{\mathcal{C}} \to \mathbb{R}^+$ over the set of clauses. Given an assignment $\phi : \mathcal{N} \to \{0,1\}$, let $C(\phi) \subseteq \mathcal{C}$ be the set of clauses satisfied by ϕ. The goal is to find an assignment ϕ that maximizes $f(C(\phi))$.

Usually, an assignment ϕ can give each variable exactly a single truth value. However, for the sake of the algorithm we extend the notion of assignments, and think of an extended assignment ϕ' which is a relation $\phi' \subseteq \mathcal{N} \times \{0,1\}$. That is, the assignment ϕ' can assign up to 2 truth values to each variable. A clause C is satisfied by an (extended) assignment ϕ', if there exists a positive literal in the clause which is assigned to truth value 1, or there exists a negative literal in the clause which is assigned a truth value 0. Note again that it might happen that some variable is assigned both 0 and 1. Note also, that an assignment is a feasible solution to the original problem if and only if it assigns exactly one truth value to every variable of $\mathcal{N}$. Let $C(\phi')$ be the set of clauses satisfied by ϕ'. We define $g : \mathcal{N} \times \{0,1\} \to \mathbb{R}^+$ using $g(\phi') \triangleq f(C(\phi'))$. Using this notation, we can restate SSAT as the problem of maximizing the function g over the set of feasible assignments.

Observation IV.1. *The function g is a normalized monotone submodular function.*

The algorithm we design for SSAT conducts n iterations. It maintains at each iteration $1 \le i \le n$ two assignments X_i and Y_i which always satisfy that: $X_i \subseteq Y_i$. Initially, $X_0 = \emptyset$ assigns no truth values to the variables, and $Y_0 = \mathcal{N} \times \{0,1\}$ assigns both truth values to all variables. The algorithm considers the variables in an arbitrary order $u_1, u_2, \ldots, u_n$. For every variable u_i, the algorithm evaluates the marginal profit from assigning it only the truth value 0 in both assignments, and assigning it only the truth value 1 in both assignments. Based on these marginal values, the algorithm makes a random decision on the truth value assigned to u_i. After the algorithm considers a variable u_i, both assignments X_i and Y_i agree on the single truth value assigned to u_i. Thus, when the algorithm terminates both assignments are identical and feasible. A formal statement of the algorithm appears as Algorithm 3.

The proof of the Theorem I.3 uses similar ideas to previous proofs in this paper, and is therefore, deferred to Appendix B. Note, however, that unlike for the previous algorithms, here, the fact that the algorithm runs in linear time requires a proof. The source of the difficulty is that we have an oracle access to f, but use the function g in the algorithm. Therefore, we need to prove that we may implement all oracle queries of g that are conducted during the execution of the algorithm in linear time.

A Note on Max-SAT: The well known Max-SAT problem is in fact a special case of SSAT where f is a linear function. We note that Algorithm 3 can be applied to Max-SAT in order to achieve a (3/4)-approximation in linear time, however, this is not immediate. This result is summarized in the following theorem. The proof of this theorem is deferred to a full version of the paper.

Algorithm 3: RandomizedSSAT(f, Ψ)

```
1  $X_0 \leftarrow \emptyset$, $Y_0 \leftarrow \mathcal{N} \times \{0,1\}$.
2  for $i$ = 1 to $n$ do
3      $a_{i,0} \leftarrow g(X_{i-1} \cup \{u_i, 0\}) - g(X_{i-1})$.
4      $a_{i,1} \leftarrow g(X_{i-1} \cup \{u_i, 1\}) - g(X_{i-1})$.
5      $b_{i,0} \leftarrow g(Y_{i-1} \setminus \{u_i, 0\}) - g(Y_{i-1})$.
6      $b_{i,1} \leftarrow g(Y_{i-1} \setminus \{u_i, 1\}) - g(Y_{i-1})$.
7      $s_{i,0} \leftarrow \max\{a_{i,0} + b_{i,1}, 0\}$.
8      $s_{i,1} \leftarrow \max\{a_{i,1} + b_{i,0}, 0\}$.
9      with probability $s_{i,0}/(s_{i,0} + s_{i,1})^*$ do:
       $X_i \leftarrow X_{i-1} \cup \{u_i, 0\}$, $Y_i \leftarrow Y_{i-1} \setminus \{u_i, 1\}$.
10     else (with the compliment probability
       $s_{i,1}/(s_{i,0} + s_{i,1})$) do:
11         $X_i \leftarrow X_{i-1} \cup \{u_i, 1\}$, $Y_i \leftarrow Y_{i-1} \setminus \{u_i, 0\}$.
12 return $X_n$ (or equivalently $Y_n$).
```

* If $s_{i,0} = s_{i,1} = 0$, we assume $s_{i,0}/(s_{i,0} + s_{i,1}) = 1$.

Theorem IV.2. *Algorithm 3 has a linear time implementation for instances of Max-SAT.*

B. A $(3/4)$-Approximation for Submodular Welfare with 2 Players

The input for the Submodular Welfare problem consists of a ground set $\mathcal{N}$ of n elements and k players, each equipped with a normalized monotone submodular utility function $f_i : 2^{\mathcal{N}} \to \mathbb{R}^+$. The goal is to divide the elements among the players while maximizing the social welfare. Formally, the objective is to partition $\mathcal{N}$ into $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k$ while maximizing $\sum_{i=1}^{k} f_i(\mathcal{N}_i)$.

We give below two different short proofs of Theorem I.4 via reductions to SSAT and USM, respectively. The second proof is due to Vondrák [37].

Proof of Theorem I.4: We provide here two proofs.

Proof (1): Given an instance of SW with 2 players, construct an instance of SSAT as follows:

1) The set of variables is $\mathcal{N}$.
2) The CNF formula Ψ consists of $2|\mathcal{N}|$ singleton clauses; one for every possible literal.
3) The objective function $f : 2^{\mathcal{C}} \to \mathbb{R}^+$ is defined as following. Let $P \subseteq \mathcal{C}$ be the set of clauses of Ψ consisting of positive literals. Then, $f(C) = f_1(C \cap P) + f_2(C \cap (\mathcal{C} \setminus P))$.

Every assignment ϕ to this instance of SSAT corresponds to a solution of SW using the following rule: $\mathcal{N}_1 = \{u \in \mathcal{N} | \phi(u) = 0\}$ and $\mathcal{N}_2 = \{u \in \mathcal{N} | \phi(u) = 1\}$. One can easily observe that this correspondence is reversible, and that each assignment has the same value as the solution it corresponds to. Hence, the above reduction preserves approximation ratios.

Moreover, queries of f can be answered in constant time using the following technique. We track for every subset $C \subseteq \mathcal{C}$ in the algorithm the subsets $C \cap P$ and $C \cap (\mathcal{C} \setminus P)$. For Algorithm 3 this can be done without effecting its running time. Then, whenever the value of $f(C)$ is queried, answering it simply requires making two oracle queries: $f_1(C \cap P)$ and $f_2(C \cap (\mathcal{C} \setminus P))$.

Proof (2): In any feasible solution to SW with two players, the set $\mathcal{N}_1$ uniquely determines the set $\mathcal{N}_2 = \mathcal{N} - \mathcal{N}_1$. Hence, the value of the solution as a function of $\mathcal{N}_1$ is given by $g(\mathcal{N}_1) = f_1(\mathcal{N}_1) + f_2(\mathcal{N} - \mathcal{N}_1)$. Thus, SW with two players can be restated as the problem of maximizing the function g over the subsets of $\mathcal{N}$.

Observe that the function g is a submodular function, but unlike f_1 and f_2, it is possibly non-monotone. Moreover, we can answer queries to the function g using only two oracle queries to f_1 and f_2[3]. Thus, we obtain an instance of USM. We apply Algorithm 2 to this instance. Using the analysis of Algorithm 2 as is, provides only a $(1/2)$-approximation for our problem. However, by noticing that $g(\varnothing) + g(\mathcal{N}) \geq f_1(\mathcal{N}) + f_2(\mathcal{N}) \geq g(OPT)$, the claimed $(3/4)$-approximation is obtained. ∎

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] Shabbir Ahmed and Alper Atamtürk. Maximizing a class of submodular utility functions. *Mathematical Programming*, 128:149–169, 2011.

[3] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. To appear in SIAM Journal on Computing.

[4] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *FOCS*, pages 575–584, 2010.

[5] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *STOC*, pages 783–792, 2011.

[6] V. P. Cherenin. Solving some combinaotiral problems of optimal planning by the method of successive calculations. Proceedings of the Conference on Experiences and Perspectives on the Applications of Mathematical Methods and Electronic Computers in Planning, Mimeograph, Novosibirsk, 1962 (in Russian).

[7] 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.

[3]For every algorithm, assuming a representation of sets allowing addition and removal of only a single element at a time, one can maintain the complement sets of all sets maintained by the algorithm without changing the running time. Hence, we need not worry about the calculation of $\mathcal{N} - \mathcal{N}_1$.

[8] G. Cornuejols, M. L. Fisher, and G. L. Nemhauser. On the uncapacitated location problem. *Annals of Discrete Mathematics*, 1:163–177, 1977.

[9] Shahar Dobzinski and Jan Vondrák. From query complexity to computational complexity. In *STOC*, 2012.

[10] Shaddin Dughmi, Tim Roughgarden, and Mukund Sundararajan. Revenue submodularity. In *Proceedings of the 10th ACM conference on Electronic commerce*, EC '09, pages 243–252, 2009.

[11] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. In *FOCS*, pages 461–471, 2007.

[12] Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. Nonmonotone submodular maximization via a structural continuous greedy algorithm. In *ICALP*, pages 342–353, 2011.

[13] Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *FOCS*, 2011.

[14] Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *SODA*, pages 1098–1117, 2011.

[15] 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.

[16] Boris Goldengorin and Diptesh Ghosh. A multilevel search algorithm for the maximization of submodular functions applied to the quadratic cost partition problem. *J. of Global Optimization*, 32(1):65–82, May 2005.

[17] Boris Goldengorin, Gerard Sierksma, Gert A. Tijssen, and Michael Tso. The data-correcting algorithm for the minimization of supermodular functions. *Manage. Sci.*, 45(11):1539–1551, November 1999.

[18] Boris Goldengorin, Gert A. Tijssen, and Michael Tso. The maximization of submodular functions : old and new proofs for the correctness of the dichotomy algorithm. Research Report 99A17, University of Groningen, Research Institute SOM (Systems, Organisations and Management), 1999.

[19] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: offline and secretary algorithms. In *Proceedings of the 6th international conference on Internet and network economics*, WINE'10, pages 246–257. Springer-Verlag, 2010.

[20] Eran Halperin and Uri Zwick. Combinatorial approximation algorithms for the maximum directed cut problem. In *SODA*, pages 1–7, 2001.

[21] Jason Hartline, Vahab Mirrokni, and Mukund Sundararajan. Optimal marketing strategies over social networks. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 189–198, 2008.

[22] Johan Hȧstad. Some optimal inapproximability results. *J. ACM*, 48:798–859, July 2001.

[23] Satoru Iwata and Kiyohito Nagano. Submodular function minimization under covering constraints. In *FOCS*, pages 671–680, 2009.

[24] Stefanie Jegelka and Jeff Bilmes. Cooperative cuts: Graph cuts with submodular edge weights. Technical Report 189-03-2010, Max Planck Institute for Biological Cybernetics, Tuebingen, 2010.

[25] 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.

[26] V. R. Khachaturov. Some problems of the consecutive calculation method and its applications to location problems. Ph.D. thesis, Central Economics & Mathematics Institute, Russian Academy of Sciences, Moscow , 1968 (in Russian).

[27] 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.

[28] Ariel Kulik, Hadas Shachnai, and Tami Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *SODA*, pages 545–554, 2009.

[29] Heesang Lee, George L. Nemhauser, and Yinhua Wang. Maximizing a submodular function by integer programming: Polyhedral results for the quadratic case. *European Journal of Operational Research*, 94(1):154 – 166, 1996.

[30] 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*, (4):2053–2078, 2010.

[31] Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. In *APPROX*, pages 244–257, 2009.

[32] L. Lovász M. Grötschel and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatoria*, 1(2):169–197, 1981.

[33] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In J. Stoer, editor, *Optimization Techniques*, volume 7 of *Lecture Notes in Control and Information Sciences*, pages 234–243. Springer Berlin / Heidelberg, 1978.

[34] A. S. Schulz and N. A. Uhan. Approximating the least core and least core value of cooperative games with supermodular costs. Working paper, 2010.

[35] Zoya Svitkina and Lisa Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. In *FOCS*, pages 697–706, 2008.

[36] Luca Trevisan, Gregory B. Sorkin, Madhu Sudan, and David P. Williamson. Gadgets, approximation, and linear programming. *SIAM J. Comput.*, 29:2074–2097, April 2000.

[37] Jan Vondrák. personal communication.

[38] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC*, pages 67–74, 2008.

[39] Jan Vondrák. Symmetry and approximability of submodular maximization problems. In *FOCS*, pages 651–670, 2009.

Appendix

A. A $(1/2)$-Approximation for USM Using Fractional Values

In this section we present Algorithm 4, which is the continuous counterpart of Algorithm 2, presented in Section III.

In order to describe Algorithm 4, we need some notation. Given a submodular function $f : \mathcal{N} \to \mathbb{R}^+$, its multilinear extension is a function $F : [0,1]^{\mathcal{N}} \to \mathbb{R}^+$, whose value at point $x \in [0,1]^{\mathcal{N}}$ is the expected value of f over a random subset $R(x) \subseteq \mathcal{N}$. The subset $R(x)$ contains each element $u \in \mathcal{N}$ independently with probability x_u. Formally, for every $x \in [0,1]^{\mathcal{N}}$, $F(x) \triangleq \mathbb{E}[R(x)] = \sum_{S\subseteq\mathcal{N}} f(S)\prod_{u\in S} x_u \prod_{u\notin S}(1-x_u)$. For two vectors $x, y \in [0,1]^{\mathcal{N}}$, we use $x \vee y$ and $x \wedge y$ to denote the coordinate-wise maximum and minimum, respectively, of x and y (formally, $(x \vee y)_u = \max\{x_u, y_u\}$ and $(x \wedge y)_u = \min\{x_u, y_u\}$).

We abuse notations both in the description of the algorithm and in its analysis, and unify a set with its characteristic vector. We also assume that we have an oracle access to F. If this is not the case, then one can approximate the value of F arbitrarily well using sampling. This makes the approximation ratio deteriorate by a low order term only (see, *e.g.*, [3] for details).

Algorithm 4: MultilinearUSM$(f, \mathcal{N})$

```
1 $x_0 \leftarrow \emptyset$, $y_0 \leftarrow \mathcal{N}$.
2 for $i$ = 1 to $n$ do
3     $a'_i \leftarrow F(x_{i-1} + \{u_i\}) - F(x_{i-1})$.
4     $b'_i \leftarrow F(y_{i-1} - \{u_i\}) - F(y_{i-1})$.
5     $a'_i \leftarrow \max\{a_i, 0\}$, $b'_i \leftarrow \max\{b_i, 0\}$.
6     $x_i \leftarrow x_{i-1} + \frac{a'_i}{a'_i+b'_i} \cdot \{u_i\}^*$.
7     $y_i \leftarrow y_{i-1} - \frac{b'_i}{a'_i+b'_i} \cdot \{u_i\}^*$.
8 return a random set $R(x_n)$ (or equivalently $R(y_n)$).
```

* If $a'_i = b'_i = 0$, we assume $a'_i/(a'_i + b'_i) = 1$ and $b'_i/(a'_i + b'_i) = 0$.

The main difference between Algorithms 2 and 4 is that Algorithm 2 chooses each element with some probability, whereas Algorithm 4 assigns a fractional value to the element. This requires the following modifications to the algorithm:

- The sets $X_i, Y_i \subseteq 2^{\mathcal{N}}$ are replaced by the vectors $x_i, y_i \in [0,1]^{\mathcal{N}}$.
- Algorithm 4 uses the multilinear extension F instead of the original submodular function f.

Theorem A.1. *If one has an oracle access to F, Algorithm 4 is a* $(1/2)$*-approximation algorithm for* NSM. *Otherwise, it can be implemented using sampling, and achieves an approximation ratio of* $(1/2) - o(1)$.

The rest of this section is devoted to proving Theorem A.1. Similarly to Section III, define $OPT_i \triangleq (OPT \vee x_i) \wedge y_i$. Examine the sequence $F(OPT_0), \dots, F(OPT_n)$. Notice that $OPT_0 = OPT$, *i.e.*, the sequence starts with the value of an optimal solution, and that $OPT_n = x_n = y_n$, *i.e.*, the sequence ends at a fractional point whose value is the expected value of the algorithm's output. The following lemma upper bounds the loss of two consecutive elements in the sequence. Formally, $F(OPT_{i-1}) - F(OPT_i)$ is upper bounded by the *average* change in the value of the two solutions maintained by the algorithm, *i.e.*,

$$1/2 \cdot [F(x_i) - F(x_{i-1}) + F(y_i) - F(y_{i-1})].$$

Lemma A.2. *For every* $1 \le i \le n$,

$$F(OPT_{i-1}) - F(OPT_i) \le \frac{1}{2} \cdot [F(x_i) - F(x_{i-1}) + F(y_i) - F(y_{i-1})] \ .$$

Before proving Lemma A.2, let us show that Theorem A.1 follows from it.

Proof of Theorem A.1: Summing up Lemma A.2 for every $1 \le i \le n$ gives:

$$\sum_{i=1}^{n}[F(OPT_{i-1}) - F(OPT_i)] \le \frac{1}{2} \cdot \sum_{i=1}^{n}[F(x_i) - F(x_{i-1})] + \frac{1}{2} \cdot \sum_{i=1}^{n}[F(y_i) - F(y_{i-1})] \ .$$

The above sum is telescopic. Collapsing it, we get:

$$F(OPT_0) - F(OPT_n) \le \frac{1}{2} \cdot [F(x_n) - F(x_0)] + \frac{1}{2} \cdot [F(y_n) - F(y_0)] \le \frac{F(x_n) + F(y_n)}{2} \ .$$

Recalling the definitions of OPT_0 and OPT_n, we obtain that $F(x_n) = F(y_n) \ge f(OPT)/2$. ■

It is left to prove Lemma A.2.

Proof of Lemma A.2: By Lemma II.1, $a_i + b_i \ge 0$, therefore, it cannot be that both a_i, b_i are strictly less than zero. Thus, we have 3 cases to consider.

Case 1 ($\mathbf{a_i \ge 0}$ and $\mathbf{b_i \le 0}$): In this case $a'_i/(a'_i + b'_i) = 1$, and so $y_i = y_{i-1}$, $x_i \leftarrow x_{i-1} \vee \{u_i\}$. Hence, $F(y_i) - F(y_{i-1}) = 0$. Also, by our definition $OPT_i = (OPT \vee x_i) \wedge y_i = OPT_{i-1} \vee \{u_i\}$. Thus, we are left to prove that:

$$F(OPT_{i-1}) - F(OPT_{i-1} \vee \{u_i\}) \leq \frac{1}{2} \cdot [F(x_i) - F(x_{i-1})] = a_i/2 \ .$$

If $u_i \in OPT$, then the left hand side of the above equation is 0, which is clearly no larger than the non-negative $a_i/2$. If $u_i \notin OPT$, then:

$$F(OPT_{i-1}) - F(OPT_{i-1} \vee \{u_i\}) \leq F(y_{i-1} - \{u_i\}) - F(y_{i-1}) = b_i \leq 0 \leq a_i/2 \ .$$

The first inequality follows from submodularity since $OPT_{i-1} = ((OPT \vee x_{i-1}) \wedge y_{i-1}) \leq y_{i-1} - \{u_i\}$ (note that in this case $(y_{i-1})_{u_i} = 1$ and $(OPT_{i-1})_{u_i} = 0$).

Case 2 ($\mathbf{a_i < 0}$ and $\mathbf{b_i \geq 0}$): This case is analogous to the previous one, and therefore, we omit its proof.

Case 3 ($\mathbf{a_i \geq 0}$ and $\mathbf{b_i > 0}$): In this case $a'_i = a_i, b'_i = b_i$ and so, $x_i \leftarrow x_{i-1} + \frac{a_i}{a_i+b_i} \cdot \{u_i\}$ and $y_i \leftarrow y_{i-1} - \frac{b_i}{a_i+b_i} \cdot \{u_i\}$. Therefore, we have,

$$F(x_i) - F(x_{i-1}) = \frac{a_i}{a_i + b_i} \cdot [F(x_{i-1} \vee \{u_i\}) - F(x_{i-1})] = \frac{a_i^2}{a_i + b_i} \ . \quad (4)$$

A similar argument shows that:

$$F(y_i) - F(y_{i-1}) = \frac{b_i^2}{a_i + b_i} \ . \quad (5)$$

Next, we upper bound $F(OPT_{i-1}) - F(OPT_i)$. For simplicity, let us assume $u_i \notin OPT$ (the proof for the other case is similar). Recall, that $OPT_i = (OPT \vee x_i) \wedge y_i$.

$$F(OPT_{i-1}) - F(OPT_i) = \frac{a_i}{a_i + b_i} \cdot [F(OPT_{i-1}) - F(OPT_{i-1} \vee \{u_i\})] \leq \frac{a_i}{a_i + b_i} \cdot [F(y_{i-1} - \{u_i\}) - F(y_{i-1})] = \frac{a_i b_i}{a_i + b_i} \ . \quad (6)$$

The inequality follows from the submodularity of f since,

$$OPT_{i-1} = ((OPT \vee x_{i-1}) \wedge y_{i-1}) \leq y_{i-1} - \{u_i\}$$

(note again that in this case $(y_{i-1})_{u_i} = 1$ and $(OPT_{i-1})_{u_i} = 0$). Plugging (4), (5) and (6) into the inequality that we need to prove, we get the following:

$$\frac{a_i b_i}{a_i + b_i} \leq \frac{1}{2} \cdot \frac{a_i^2 + b_i^2}{a_i + b_i} \ ,$$

which can be easily verified. ■

B. Proof of Theorem I.3

In this section we prove that Algorithm 3 is a randomized linear time (3/4)-approximation algorithm for SSAT. As a first step we state the following useful lemma.

Lemma A.3. *For every $1 \leq i \leq n$,*

$$\mathbb{E}[g(OPT_{i-1}) - g(OPT_i)] \leq \frac{1}{2} \cdot \mathbb{E}\left[(g(X_i) - g(X_{i-1}) + g(Y_i) - g(Y_{i-1})\right] \ , \quad (7)$$

where expectations are taken over the random choices of the algorithm.

The proof of this lemma is deferred to a full version of the paper. Let us just note that the proof follows the lines of Lemma III.1. Let us show that the approximation guarantee of Theorem I.3 follows from the above lemma. The proof that Algorithm 3 can be implemented to run in linear time is deferred to a full version of the paper.

Proof of approximation guarantee of Algorithm 3: Summing up Lemma A.3 for every $1 \leq i \leq n$ we get,

$$\sum_{i=1}^{n} \mathbb{E}[g(OPT_{i-1}) - g(OPT_i)] \leq \frac{1}{2} \cdot \sum_{i=1}^{n} \mathbb{E}[g(X_i) - g(X_{i-1}) + g(Y_i) - g(Y_{i-1})] \ .$$

The above sum is telescopic. Collapsing it, we get:

$$\begin{aligned} &\mathbb{E}[g(OPT_0) - g(OPT_n)] \\ &\leq \frac{1}{2} \cdot \mathbb{E}[g(X_n) - g(X_0) + g(Y_n) - g(Y_0)] \\ &\leq \frac{\mathbb{E}[g(X_n) + g(Y_n) - g(Y_0)]}{2} \ . \end{aligned}$$

Recalling the definitions of OPT_0 and OPT_n, we obtain that:

$$\mathbb{E}[g(X_n)] = \mathbb{E}[g(Y_n)] \geq g(OPT)/2 + g(Y_0)/4.$$

The approximation ratio now follows from the observation that Y_0 satisfies all clauses of Ψ, and therefore, $g(Y_0) \geq g(OPT)$.