Approximation Algorithms for Unconstrained Submodular Maximization

Nahom Seyoum[1]

Department of Computer Science
Yale University
2024

[1]nahom.seyoum@yale.edu

Contents

Chapter 1

1/2 Tight Approximation

1.1 Paper introduction

The paper I am reviewing is titled *A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization* by Buchbinder et al. published in 2015 [BFNS15]. The paper outlines various approximation algorithms for Unconstrained Submodular Maximization. The goal of this report is to describe the intuition for some of the algorithms and establish, in varying degrees of technical detail, the mathematics behind the algorithms and their respective justifications.

I will be assuming a rudimentary understanding of submodularity, defined as follows.

Definition 1 (Submodular Function) *Let $f : 2^X \to \mathbb{R}$ be a function. The function f is called* submodular *if for any subsets $S, T \subseteq X$,*

$$f(S \cup T) + f(S \cap T) \leq f(S) + f(T).$$

An alternative definition of submodularity is the property of *decreasing marginal values*: For any $A \subseteq B \subseteq X$ and $x \in X \setminus B$,

$$f(B \cup \{x\}) - f(B) \leq f(A \cup \{x\}) - f(A).$$

The problem we are ultimately interested in studying, as aformentioned, is **Unconstrained Submodular Maximization (USM)**. In USM, we are given a non-negative submodular function, $f : 2^N \to \mathbb{R}_+$ and the goal is to identify a subset $S \subseteq N$ that maximizes the value of $f(S)$. We are imposing no restrictions on the nature of the subset S which makes it convenient to model multiple combinatorial problems like Max-Cut and Max-SAT. It also has practical applications in optimization and game theory. Since the problems that USM captures are usually NP hard, there is an impetus to come up with approximation algorithms for it. The paper we are reviewing today presents multiple approximation algorithms which are significant improvements from the previous state of the art algorithms.

1.2 Motivation

1.2.1 Previous Algorithms

The paper, in their literaure review, rehash some of the work done on USM by Feige et al [FMV07]. Feige et al. showed that selecting a subset $S \subseteq N$ uniformly at random achieves a $\frac{1}{4}$- approximation to the optimal solution. They also introduced two local search algorithms. The first local search algorithm directly optimizes the original submodular function f achieving a $\frac{1}{3}$ approximation. The second algorithm incorporates a noisy version of f and achieves a $\frac{2}{5}$-approximation as it navigates local minimas slightly better. Subsequent authors have introduced simulated annealing and revisions of it which have yielded, at best, a 0.42-approximation

1.2.2 Theoretical Hard Bound

Feige et al., in the same work, established a fundamental limitation in the context of optimizing symmetric submodular functions within the value oracle model. Specifically, they proved that any algorithm attempting to achieve a better than $\frac{1}{2}$-approximation for such functions must make an exponential number of queries to the value oracle. This result implies that the $\frac{1}{2}$-approximations are tight.

Despite this limitation, there remains a gap between the theoretical bound established by the paper and the best performance achieved by the algorithms discussed in Section 1.2.1. Thus, he primary goal ofBuchbinder et al. is to close this gap through an algorithm that matches the theoretical bound. They outline a deterministic algorithm from which they build the randomized algorithm, which ultimately achieves the $\frac{1}{2}$- tight bound.

1.3 Deterministic Algorithm

1.3.1 Algorithm Setup and Explanation

The algorithm is essentially a greedy algorithm. It begins with two sets: X_0, initially empty, representing a growing solution, and Y_0, initially set to the entire ground set N, representing a shrinking complement. It processes each element u_i in a fixed order, iteratively deciding whether to include it in X or exclude it from Y. At each step, the algorithm calculates two marginal gains: a_i, the gain from adding u_i to X_{i-1}, and b_i, the gain from removing u_i from Y_{i-1}. Based on which marginal gain is greater, the algorithm either adds u_i to X and leaves Y unchanged, or excludes u_i from Y and leaves X unchanged. At the end, we get, $X_n = Y_n$, which is the final solution.

Algorithm 1: DeterministicUSM

```
1  Initialize $X_0 \leftarrow \emptyset$, $Y_0 \leftarrow N$;
2  for $i = 1$ to $n$ do
3      Compute $a_i \leftarrow f(X_{i-1} \cup \{u_i\}) - f(X_{i-1})$;
4      Compute $b_i \leftarrow f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1})$;
5      if $a_i \geq b_i$ then
6          $X_i \leftarrow X_{i-1} \cup \{u_i\}$;
7          $Y_i \leftarrow Y_{i-1}$;
8      else
9          $X_i \leftarrow X_{i-1}$;
10         $Y_i \leftarrow Y_{i-1} \setminus \{u_i\}$;
11 return $X_n$ (or equivalently $Y_n$);
```

1.3.2 Example Runthrough

Below we will work through a toy example to illustrate how the algorithm works. Consider f, a signed area. In this case, intersections are counted only once, so it is clearly submodular. The colored regions contribute positive area, and the white "donut" contributes negative area. The function f is bounded below by 0 to ensure non-negativity. We will apply the deterministic algorithm to the above setup.

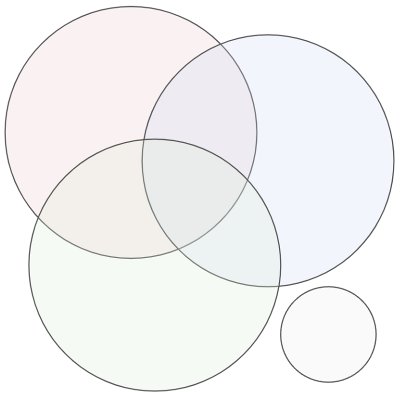

Figure 1.1: Toy example For Deterministic Algorithm

Initial State

$X_0 = \emptyset$ (The growing solution starts empty.)

$Y_0 = \{R, G, B, W\}$ (Initially contains all regions: Red R, Green G, Blue B, and White W.)

Process $u_1 = R$ (Red Region)

Marginal gain for adding R to X:

$$a_1 = f(\{R\}) - f(\emptyset) \geq 0 \quad \text{(since adding } R \text{ increases the area.)}$$

Marginal gain for removing R from Y:

$$b_1 = f(\{G, B, W\}) - f(\{R, G, B, W\}) \leq 0 \quad \text{(since removing } R \text{ amplifies the negative contribution of } W.)$$

Since $a_1 \geq b_1$, R is included in X:

$$X_1 = \{R\}, \quad Y_1 = \{R, G, B, W\}.$$

Process $u_2 = G$ (Green Region) and $u_3 = B$ (Blue Region)

For both G and B: Adding the region ($a_i \geq 0$) increases the positive area and removing the region ($b_i \leq 0$) reduces the total positive contribution.

Both G and B are added to X, resulting in:

$$X_3 = \{R, G, B\}, \quad Y_3 = \{R, G, B, W\}.$$

Process $u_4 = W$ (White Donut)

Marginal gain for adding W to X:

$a_4 = f(\{R, G, B, W\}) - f(\{R, G, B\}) \leq 0$ (as W contributes negatively to the total signed area.)

Marginal gain for removing W from Y:

$b_4 = f(\{R, G, B\}) - f(\{R, G, B, W\}) \geq 0$ (as removing W eliminates its negative effect.)

Since $a_4 < b_4$, W is excluded:

$$X_4 = \{R, G, B\}, \quad Y_4 = \{R, G, B\}.$$

1.3.3 Proof of $\frac{1}{3}$ Approximation

Ultimately, we are interested in the performance of the above algorithm, and the paper establishes that this achieves a $\frac{1}{3}$ approximation.

There are two lemmas we need to prove to show this result.

Lemma 2 *For every $1 \leq i \leq n$, $a_i + b_i \geq 0$.*

The above lemma ultimately argues that the combined effect of adding an element to one set and removing it from another related set does not decrease the overall function value. We will not prove this result but it is easy to see that it follows directly from submodularity.

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

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

This lemma simply provides an upper bound on how much the value of the optimal solution, $f(\text{OPT}_i)$, decreases as the algorithm progresses through each iteration i. Specifically, it states that the "loss" in the value of OPT (the optimal solution set), at step i, which is $f(\text{OPT}_{i-1}) - f(\text{OPT}_i)$, is at most the combined increases in the values of the two solutions maintained by the algorithm (X_i and Y_i) during that same step.

In the following proof, we establish this in more technical detail.

Proof Recall the definitions from the algorithm: Initially, $X_0 = \emptyset$ and $Y_0 = N$. For each $i = 1, \dots, n$, let

$$a_i := f(X_{i-1} \cup \{u_i\}) - f(X_{i-1}), \quad b_i := f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1}).$$

The algorithm compares a_i and b_i at each iteration i: If $a_i \geq b_i$, then $X_i = X_{i-1} \cup \{u_i\}$ and $Y_i = Y_{i-1}$. Otherwise, if $b_i > a_i$, then $X_i = X_{i-1}$ and $Y_i = Y_{i-1} \setminus \{u_i\}$.
We define:

$$\mathrm{OPT}_i := (\mathrm{OPT} \cup X_i) \cap Y_i.$$

Note that $\mathrm{OPT}_0 = \mathrm{OPT}$ and $\mathrm{OPT}_n = X_n = Y_n$.
We must show that for every $1 \leq i \leq n$,

$$f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i) \ \leq \ [f(X_i) - f(X_{i-1})] + [f(Y_i) - f(Y_{i-1})].$$

Case 1: $a_i \geq b_i$.
In this case, $X_i = X_{i-1} \cup \{u_i\}$ and $Y_i = Y_{i-1}$. Thus:

$$f(X_i) - f(X_{i-1}) = a_i \quad \text{and} \quad f(Y_i) - f(Y_{i-1}) = 0.$$

Since $Y_i = Y_{i-1}$, we have

$$\mathrm{OPT}_i = (\mathrm{OPT} \cup X_i) \cap Y_i = (\mathrm{OPT} \cup (X_{i-1} \cup \{u_i\})) \cap Y_{i-1}.$$

Recall that $\mathrm{OPT}_{i-1} = (\mathrm{OPT} \cup X_{i-1}) \cap Y_{i-1}$. Adding u_i to X_{i-1} while not changing Y_{i-1} gives:

$$\mathrm{OPT}_i = \mathrm{OPT}_{i-1} \cup \{u_i\}.$$

Hence:

$$f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i) = f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1} \cup \{u_i\}).$$

Rearranging, we must prove:

$$f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1} \cup \{u_i\}) \leq a_i.$$

Consider two subcases:

If $u_i \in$ **OPT**, then $f(\mathrm{OPT}_{i-1}) = f(\mathrm{OPT}_{i-1} \cup \{u_i\})$, so the left-hand side is 0. Since $a_i \geq b_i$ and $a_i + b_i \geq 0$ by Lemma II.1, it follows that $a_i \geq 0$. Thus $0 \leq a_i$.

If $u_i \notin$ **OPT**, then $u_i \notin \mathrm{OPT}_{i-1}$. By submodularity, for any $A \subseteq B$ and any element z,

$$f(A) - f(A \cup \{z\}) \leq f(B) - f(B \cup \{z\}).$$

Set $A = \mathrm{OPT}_{i-1}$ and $B = Y_{i-1} \setminus \{u_i\}$. Since $\mathrm{OPT}_{i-1} \subseteq Y_{i-1} \setminus \{u_i\}$, we get:

$$f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1} \cup \{u_i\}) \leq f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1}) = b_i.$$

Since $a_i \geq b_i$, we have $f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1} \cup \{u_i\}) \leq b_i \leq a_i$.

Thus, in both subcases under $a_i \geq b_i$, we have the desired inequality.

Case 2: $b_i > a_i$.
In this case, $X_i = X_{i-1}$ and $Y_i = Y_{i-1} \setminus \{u_i\}$. By symmetry, interchanging the roles of X

and Y and the roles of a_i and b_i, a completely analogous argument shows:

$$f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i) \leq b_i = [f(X_i) - f(X_{i-1})] + [f(Y_i) - f(Y_{i-1})],$$

since in this case $f(X_i) - f(X_{i-1}) = 0$ and $f(Y_i) - f(Y_{i-1}) = b_i$.

In both cases, we have proven that:

$$f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i) \ \leq \ [f(X_i) - f(X_{i-1})] + [f(Y_i) - f(Y_{i-1})],$$

completing the proof. □

Theorem 4 *There exists a deterministic linear time (1/3)- approximation algorithm for the Unconstrained Submodular Maximization problem.*

Proof From Lemma II.2, summing the inequality over all iterations $i = 1, \dots, n$, we have:

$$\sum_{i=1}^{n} [f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i)] \leq \sum_{i=1}^{n} [f(X_i) - f(X_{i-1})] + \sum_{i=1}^{n} [f(Y_i) - f(Y_{i-1})] \,.$$

We can use a simple telescoping argument for both sides of the inequality.

LHS:

$$\sum_{i=1}^{n} [f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i)] = f(\mathrm{OPT}_0) - f(\mathrm{OPT}_n).$$

RHS:

$$\sum_{i=1}^{n} [f(X_i) - f(X_{i-1})] = f(X_n) - f(X_0),$$

$$\sum_{i=1}^{n} [f(Y_i) - f(Y_{i-1})] = f(Y_n) - f(Y_0).$$

Simplifying,

$$f(\mathrm{OPT}_0) - f(\mathrm{OPT}_n) \leq (f(X_n) - f(X_0)) + (f(Y_n) - f(Y_0)) \,.$$

Recall that $\mathrm{OPT}_0 = \mathrm{OPT}$, $\mathrm{OPT}_n = X_n = Y_n$, and $X_0 = \emptyset$, hence:

$$f(\mathrm{OPT}) - f(X_n) \ \leq \ f(X_n) + f(Y_n).$$

As $X_n = Y_n$, we get:

$$f(\mathrm{OPT}) - f(X_n) \ \leq \ 2f(X_n) \implies f(X_n) \geq \frac{f(\mathrm{OPT})}{3}.$$

Thus, the algorithm achieves a (1/3)-approximation. □

1.3.4 Proof of Tightness

In the paper, they outline a way of proving that this analysis is tight through the following weighted graph:

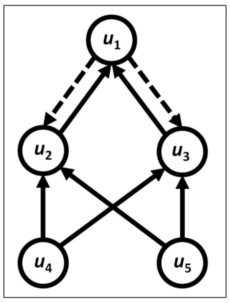

Figure 1.2: Weighted Graph For Proof of Tightness

The dashed edges (u_1, u_2) and (u_1, u_3) have weights of $1 - \epsilon$ and all other edges have weights of 1.
The optimal cut in the graph is the set $S = \{u_1, u_4, u_5\}$, where:

- u_1 is included to maximize the contribution of edges (u_1, u_2) and (u_1, u_3),
- u_4 and u_5 are included to maximize the contributions of edges leaving these nodes.

The weight of the edges in this cut is as follows:

$$f(\{u_1, u_4, u_5\}) = w(u_1, u_2) + w(u_1, u_3) + w(u_4, u_2) + w(u_4, u_3) + w(u_5, u_2) + w(u_5, u_3).$$

Substituting the weights:

$$f(\{u_1, u_4, u_5\}) = (1 - \epsilon) + (1 - \epsilon) + 1 + 1 + 1 + 1 = 6 - 2\epsilon.$$

Thus, the optimal solution has a cut weight of $6 - 2\epsilon$.
Now, we simply have to think about the cut produced by the deterministic algorithm.
Applying the deterministic algorithm, we get the set: $\{u_2, u_3, u_4, u_5\}$.
Substituting the weights, we get 2.

$$\text{Approximation Ratio} = \frac{f(\text{Algorithm's Cut})}{f(\text{Optimal Cut})} = \frac{2}{6 - 2\epsilon} \leq \frac{1}{3} + \epsilon$$

1.4 Randomized Algorithm

This algorithm is a probabilistic modification of the earlier deterministic algorithm. While the deterministic version always makes a greedy choice by selecting u_i based on whether $a_i \geq b_i$, this version introduces randomness by selecting u_i with a probability proportional to its marginal gain: $\frac{a_i}{a_i+b_i}$ for inclusion in X and $\frac{b_i}{a_i+b_i}$ for exclusion from Y. Since we are dealing with probability, it ensures non-negative marginal gains by setting $a_i = \max(a_i, 0)$ and $b_i = \max(b_i, 0)$,. It also defaults to a deterministic choice if both values are zero.

Algorithm 2: RandomizedUSM

```
    Input: $f$, $N$
    Output: $X_n$ (or equivalently $Y_n$)
 1  Initialize $X_0 \leftarrow \emptyset$, $Y_0 \leftarrow N$;
 2  for $i = 1$ to $n$ do
 3      Compute $a_i \leftarrow f(X_{i-1} \cup \{u_i\}) - f(X_{i-1})$;
 4      Compute $b_i \leftarrow f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1})$;
 5      Set $a_i \leftarrow \max\{a_i, 0\}$ and $b_i \leftarrow \max\{b_i, 0\}$;
 6      With probability $\frac{a_i}{a_i+b_i}$ do
 7          $X_i \leftarrow X_{i-1} \cup \{u_i\}$, $Y_i \leftarrow Y_{i-1}$;
 8      Otherwise (with complementary probability $\frac{b_i}{a_i+b_i}$):
 9          $X_i \leftarrow X_{i-1}$, $Y_i \leftarrow Y_{i-1} \setminus \{u_i\}$;
10  Note: If $a_i$ and $b_i = 0$, $\frac{a_i}{a_i+b_i} = 1$
11  return $X_n$ (or equivalently $Y_n$);
```

Ultimately, we are interested in the guarantee of the algorithm, which the authors claim is $\frac{1}{2}$. To prove this result, they rely on Lemma 2 and the following lemma.

Lemma 5 *For every* $1 \leq i \leq n$,

$$\mathbb{E}\big[f(OPT_{i-1}) - f(OPT_i)\big] \leq \frac{1}{2} \cdot \mathbb{E}\big[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})\big].$$

High-Level Idea of Proof: The proof presented below is quite lengthy so I will attempt to outline a road map and what we are doing in the proof. We need to establish the desired inequality under the condition $X_{i-1} = S_{i-1}$, where S_{i-1} is some subset of $\{u_1, \ldots, u_{i-1}\}$ that occurs with non-zero probability. Once we condition on this event, some variables become fixed, simplifying the problem. We then consider three distinct cases based on the signs of a_i and b_i. In each case, submodularity is used to bound the change in the value of OPT$_{i-1}$ when u_i is either added or removed.

Proof
Consider an event of the form $X_{i-1} = S_{i-1}$ with $\Pr(X_{i-1} = S_{i-1}) > 0$. Since this event has positive probability, we can validly condition on it. Under this conditioning:

- We know $X_{i-1} = S_{i-1}$.
- Since $Y_0 = N$ and at each step we either remove an element from Y or leave Y unchanged, after $i-1$ steps:

$$Y_{i-1} = S_{i-1} \cup \{u_i, \ldots, u_n\}.$$

- Also,

$$\text{OPT}_{i-1} = (\text{OPT} \cup X_{i-1}) \cap Y_{i-1} = (\text{OPT} \cup S_{i-1}) \cap (S_{i-1} \cup \{u_i, \ldots, u_n\}) = S_{i-1} \cup (\text{OPT} \cap \{u_i, \ldots, u_n\}).$$

Define:

$$a_i := f(X_{i-1} \cup \{u_i\}) - f(X_{i-1}), \quad b_i := f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1}).$$

These values become constants once we condition on $X_{i-1} = S_{i-1}$.

By Lemma II.1, we have $a_i + b_i \geq 0$. Hence, we cannot have both $a_i < 0$ and $b_i < 0$.

We have the following three cases

$$\textbf{(i)}\ a_i \geq 0 \text{ and } b_i \leq 0, \quad \textbf{(ii)}\ a_i < 0 \text{ and } b_i \geq 0, \quad \textbf{(iii)}\ a_i \geq 0 \text{ and } b_i > 0.$$

Case 1: $a_i \geq 0$ **and** $b_i \leq 0$

Since $b_i \leq 0 \leq a_i$, it follows that $a_i \geq b_i$. The algorithm chooses to add u_i to X with probability $\frac{a_i}{a_i+b_i} = 1$. Thus:

$$X_i = X_{i-1} \cup \{u_i\} = S_{i-1} \cup \{u_i\}, \quad Y_i = Y_{i-1}.$$

Since $Y_i = Y_{i-1}$, we have $f(Y_i) - f(Y_{i-1}) = 0$. Also:

$\mathrm{OPT}_i = (\mathrm{OPT} \cup X_i) \cap Y_i = (\mathrm{OPT} \cup (S_{i-1} \cup \{u_i\})) \cap (S_{i-1} \cup \{u_i, \ldots, u_n\}) = \mathrm{OPT}_{i-1} \cup \{u_i\}$.

We need to show:

$$f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1} \cup \{u_i\}) \leq \tfrac{1}{2}[f(X_i) - f(X_{i-1})] = \tfrac{a_i}{2}.$$

Consider two subcases:

1. If $u_i \in \mathrm{OPT}$, then adding u_i does not decrease the value of OPT_{i-1}:

   $$f(\mathrm{OPT}_{i-1}) = f(\mathrm{OPT}_{i-1} \cup \{u_i\}) \implies f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1} \cup \{u_i\}) = 0.$$

   Since $a_i \geq 0$, we have $0 \leq \frac{a_i}{2}$.

2. If $u_i \notin \mathrm{OPT}$, we have $\mathrm{OPT}_{i-1} \subseteq Y_{i-1} \setminus \{u_i\}$. By submodularity, the marginal decrease from adding u_i to OPT_{i-1} is at most the marginal decrease from adding it to $Y_{i-1} \setminus \{u_i\}$. Thus:

   $$f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1} \cup \{u_i\}) \leq f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1}) = b_i.$$

   Since $b_i \leq 0 \leq a_i$, we immediately get:

   $$f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1} \cup \{u_i\}) \leq b_i \leq 0 \leq \frac{a_i}{2}.$$

Case 2: $a_i < 0$ **and** $b_i \geq 0$
This case is symmetric to Case 1 with the roles of X and Y reversed. By an analogous submodularity argument, we obtain the required inequality.

Case 3: $a_i \geq 0$ **and** $b_i > 0$
In this case, both a_i and b_i are non-negative, and at least one of them is strictly positive. The algorithm chooses its action with probabilities proportional to a_i and b_i:

- With probability $\frac{a_i}{a_i+b_i}$, we set $X_i = X_{i-1} \cup \{u_i\}$ and $Y_i = Y_{i-1}$.
- With probability $\frac{b_i}{a_i+b_i}$, we set $X_i = X_{i-1}$ and $Y_i = Y_{i-1} \setminus \{u_i\}$.

Taking expectations, we have:

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

By the definitions of a_i and b_i:

$$a_i = f(X_{i-1}\cup\{u_i\}) - f(X_{i-1}), \quad b_i = f(Y_{i-1}\setminus\{u_i\}) - f(Y_{i-1}),$$

this simplifies to:

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

Next, we consider the expected decrease in the value of OPT:

$$\mathbb{E}[f(\mathrm{OPT}_{i-1})-f(\mathrm{OPT}_i)] = \frac{a_i}{a_i+b_i}[f(\mathrm{OPT}_{i-1})-f(\mathrm{OPT}_{i-1}\cup\{u_i\})]+\frac{b_i}{a_i+b_i}[f(\mathrm{OPT}_{i-1})-f(\mathrm{OPT}_{i-1}\setminus\{u_i\}$$

We must upper bound this quantity by $\frac{a_ib_i}{a_i+b_i}$.
To do this, we analyze two scenarios depending on whether u_i is in OPT_{i-1}:

If $u_i \notin \mathbf{OPT}_{i-1}$:

In this case, $f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1}\setminus\{u_i\}) = 0$ because removing an element not in the set does not change the value. Thus, the second term vanishes:

$$\mathbb{E}[f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i)] = \frac{a_i}{a_i+b_i}[f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1}\cup\{u_i\})].$$

Also, since $u_i \in Y_{i-1}$ and $u_i \notin \mathrm{OPT}_{i-1}$, we have $\mathrm{OPT}_{i-1} \subseteq Y_{i-1}\setminus\{u_i\}$. This means, removing u_i from Y_{i-1} does not affect any elements already in OPT_{i-1}.

Since

$$\mathrm{OPT}_{i-1} = (\mathrm{OPT}\cup X_{i-1})\cap Y_{i-1},$$

and $u_i \notin \mathrm{OPT}_{i-1}$, the set OPT_{i-1} is contained within $Y_{i-1}\setminus\{u_i\}$ because u_i does not contribute to the intersection. By submodularity:

$$f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1}\cup\{u_i\}) \le f(Y_{i-1}\setminus\{u_i\}) - f(Y_{i-1}) = b_i.$$

Therefore:

$$\mathbb{E}[f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i)] \le \frac{a_i}{a_i+b_i}b_i = \frac{a_ib_i}{a_i+b_i}.$$

If $u_i \in \mathbf{OPT}_{i-1}$:

In this scenario, adding u_i to OPT_{i-1} has no effect because u_i is already in it. Thus:

$$f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1}\cup\{u_i\}) = 0.$$

Hence:

$$\mathbb{E}[f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i)] = \frac{b_i}{a_i+b_i}[f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1}\setminus\{u_i\})].$$

Since $u_i \in \mathrm{OPT}_{i-1}$, removing u_i might decrease its value. Using a similar submodularity argument but now with $X_{i-1} \cup \{u_i\}$ as the larger set containing $\mathrm{OPT}_{i-1} \setminus \{u_i\}$, we have:

$$f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1} \setminus \{u_i\}) \leq f(X_{i-1} \cup \{u_i\}) - f(X_{i-1}) = a_i.$$

Thus:

$$\mathbb{E}[f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i)] \leq \frac{b_i}{a_i + b_i} a_i = \frac{a_i b_i}{a_i + b_i}.$$

In both subcases, we have:

$$\mathbb{E}[f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i)] \leq \frac{a_i b_i}{a_i + b_i}.$$

Finally, substituting our bounds back into the main inequality, we get:

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

Multiplying both sides by $a_i + b_i$ (which is non-negative), we get:

$$a_i b_i \leq \tfrac{1}{2}(a_i^2 + b_i^2).$$

This inequality is equivalent to:

$$a_i^2 + b_i^2 \geq 2 a_i b_i,$$

which follows from the arithmetic mean-geometric mean inequality or simply expanding $(a_i - b_i)^2 \geq 0$.

Therefore, our inequality is valid in all three cases under the conditioning event $X_{i-1} = S_{i-1}$. Since this conditioning was arbitrary (but had non-zero probability), the inequality holds unconditionally.

□

Theorem 6 *There exists a randomized linear time (1/2)-approximation algorithm for the Unconstrained Submodular Maximization problem.*

A similar telescoping argument used in Theorem I can be applied to prove that the $\frac{1}{2}$ approximation follows from the above lemma (Lemma 5).

Proof We sum the inequality over all iterations $i = 1, \ldots, n$. On the left-hand side, we have a telescoping sum of the form

$$\sum_{i=1}^{n} [f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i)].$$

On the right-hand side, we have

$$\frac{1}{2} \sum_{i=1}^{n} [f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})].$$

The telescoping nature simplifies these sums. The left-hand side collapses to $f(\mathrm{OPT}_0) -$

$f(\mathrm{OPT}_n)$ since consecutive terms cancel out. Similarly, the right-hand side becomes

$$\frac{1}{2}\left[(f(X_n) - f(X_0)) + (f(Y_n) - f(Y_0))\right].$$

Therefore, we have:

$$\mathbb{E}\left[f(\mathrm{OPT}_0) - f(\mathrm{OPT}_n)\right] \leq \frac{1}{2}\mathbb{E}\left[(f(X_n) - f(X_0)) + (f(Y_n) - f(Y_0))\right].$$

Since $\mathrm{OPT}_0 = \mathrm{OPT}$ and $\mathrm{OPT}_n = X_n = Y_n$, we get:

$$\mathbb{E}\left[f(\mathrm{OPT}) - f(X_n)\right] \leq \frac{1}{2}\mathbb{E}\left[f(X_n) + f(Y_n)\right].$$

Because $X_n = Y_n$, this simplifies to:

$$f(\mathrm{OPT}) - \mathbb{E}\left[f(X_n)\right] \leq 2\mathbb{E}\left[f(X_n)\right].$$

Rearranging, it follows that:

$$\mathbb{E}\left[f(X_n)\right] \geq \frac{f(\mathrm{OPT})}{2}.$$

$\square$

All of the above analysis forms the crux of the paper. However, they also present results and algorithms for various other types of submodular problems. I will outline the main results and some of the intuition behind them but not delve deeply into the proofs/justifications for why they work.

1.5 Submodular Max-SAT (SSAT)

The Submodular Max-SAT problem is a generalization of the classic Max-SAT problem. Instead of a linear objective (counting satisfied clauses), we have a normalized monotone submodular function f defined over the set of clauses. For a given assignment φ, let $C(\varphi)$ be the set of clauses satisfied by φ. The goal is to find an assignment φ that maximizes $f(C(\varphi))$.

Theorem 7 (Theorem I.3) *There exists a $(3/4)$-approximation algorithm for SSAT. In other words, the algorithm finds an assignment φ whose value $f(C(\varphi))$ is at least $(3/4)$ times the value of an optimal assignment.*

Algorithm 3: RandomizedSSAT

```
1  Initialize $X_0 \leftarrow \emptyset, Y_0 \leftarrow N \times \{0,1\}$;
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 $\frac{s_{i,0}}{s_{i,0}+s_{i,1}}$ do
10     $X_i \leftarrow X_{i-1} \cup u_i, 0, Y_i \leftarrow Y_{i-1} \setminus u_i, 1$.
11     Otherwise (with complementary probability $\frac{b_i}{a_i+b_i}$):
12         $X_i \leftarrow X_{i-1} \cup u_i, 1, Y_i \leftarrow Y_{i-1} \setminus u_i, 0$
13 Note: If $s_{i,0} = s_{i,1} = 0$, we assume $\frac{s_{i,0}}{s_{i,0}+s_{i,1}} = 1$.
14 return $X_n$ (or equivalently $Y_n$);
```

1.5.1 Intuition for the SSAT Algorithm

The SSAT algorithm starts by assigning both 0 and 1 to every variable, creating an extended assignment where all clauses have the potential to be satisfied. This provides maximum flexibility at the beginning. For each variable, the algorithm calculates the gain in the objective value if the variable is fixed to 0 while removing 1, and separately if it is fixed to 1 while removing 0. These gains are used to compare the two choices for each variable.

Similar to our randomized algorithm earlier, this algorithm does not always pick the larger gain. Instead, it makes a probabilistic decision, assigning the variable's value based on the relative sizes of the two gains. As the algorithm considers each variable, it gradually removes the extra assignments, leaving each variable with only one value, either 0 or 1. By the end, all variables are assigned single truth values, which gives us a feasible solution. The guarantee, as outlined in Theorem 7, is that this algorithm gives us a $\frac{3}{4}$-approximation.

The authors also establish the following result in the paper:

Theorem 8 (Theorem IV.2) *When the objective function f is linear (as in the classical Max-SAT problem), Algorithm 3 can be implemented in linear time.*

1.6 Submodular Welfare with Two Players (2-Player SW)

The final problem the authors engage with is the 2 Player submodular welfare problem. In this problem, we are given a ground set N of items and k players, each with a normalized monotone submodular utility function f_i. The goal is to partition N into $N_1, N_2, \ldots, N_k$ to maximize social welfare $\sum_{i=1}^{k} f_i(N_i)$.

Theorem 9 (Theorem I.4 for 2-player SW) *When there are only two players, there is a* $(3/4)$*-approximation algorithm for the Submodular Welfare problem. This result can be obtained by reductions to SSAT or to the problem of Unconstrained Submodular Maximization (USM).*

1.6.1 Intuition for the 2-Player SW Result

- With two players, each element is either assigned to player 1 or to player 2.
- We can encode this decision as a binary variable assignment problem: assigning a variable to 0 corresponds to giving the corresponding element to player 1, and assigning it to 1 corresponds to giving it to player 2.
- Using a suitable construction, we can represent the welfare objective as a submodular function that depends on these assignments. Applying the $(3/4)$ approximation algorithm developed for SSAT or the USM techniques directly leads to a $(3/4)$ approximation for the 2-player welfare problem.

1.7 Conclusion and Extensions

All in all, the paper outlines various approximation algorithms for USM and related problems, including Submodular Max-SAT(SSAT) and 2 Player Submodular Welfare. They establish that the deterministic and randomized algorithms for USM achieve approximation guarantees of 1/3 and 1/2, respectively, with the randomized algorithm matching the theoretical hardness bound for symmetric submodular functions. For SSAT, a generalized version of the classical Max-SAT problem, they establish a 3/4-approximation algorithm. Similarly, the reduction techniques applied to the 2-player SW problem extend the 3/4 approximation to welfare maximization, which is a testament to the versatility of these methods.

There is still significant work being done in the field of USM and there are multiple ways this can be extended. One, it would be interesting to investigate whether approximation guarantees depend intrinsically on the curvature or modularity ratio of the submodular function. Furthermore, we can try and study online or streaming versions of the problem using advanced probabilistic tools, like stochastic gradients or dynamic programming on submodular lattices. There is already some work being done in this regard [RW18] but there is still quite some unanswered questions when it comes to the generality of the developed algorithms. Finally, we can try and apply the architecture of the algorithms developed here to real world scenarious I encounter in my research such as network optimization(for trains!) and distributed systems.

Bibliography

[BFNS15] Buchbinder, Feldman, Naor, and Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *Siam J. Comput*, 44:1384, 2015.

[FMV07] Feige, Mirrokni, and Vondrak. Maximizing non-monotone submodular functions. *48th Annual IEEE Symposium on Foundations of Computer Science*, pages 461–471, 2007.

[RW18] Roughgarden and Wang. An optimal learning algorithm for online unconstrained submodular maximization. *Proceedings of Machine Learning Research*, 75:1–19, 2018.