Submodular Functions Maximization Problems

Niv Buchbinder
School of Mathematical Sciences
Tel Aviv University
Tel-Aviv 6997801, Israel
E-mail: niv.buchbinder@gmail.com

Moran Feldman
Depart. of Mathematics and Computer Science
The Open University of Israel
Raanana 4353701, Israel
E-mail: moranfe@openu.ac.il

November 6, 2017

1.1 Introduction

In this chapter we study fundamental results on maximizing a special class of functions called *submodular functions* under various combinatorial constraints. The study of submodular functions is motivated both by their many real world applications and by their frequent occurrence in more theoretical fields such as economy and algorithmic game theory. In particular, submodular functions and submodular maximization play a major role in combinatorial optimization as several well known combinatorial functions turn out to be submodular. A few examples of such functions include cuts functions of graphs and hypergraphs, rank functions of matroids and covering functions. We discuss some of these examples further in the following.

Let us begin by providing basic notation used throughout the chapter. We then give two definitions of submodular functions and prove that they are equivalent. Let $\mathcal{N} = \{u_1, u_2, \ldots, u_n\}$ be a ground set of elements. For a set A and an element $u \in \mathcal{N}$ we denote the union $A \cup \{u\}$ by $A + u$. Similarly, we denote $A \setminus \{u\}$ as $A - u$. The following is the first definition of submodular functions.

Definition 1.1. *(Submodular function: Definition 1) A function $f : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ is* submodular *if:*

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B) \qquad \forall A, B \subseteq \mathcal{N} \ . \tag{1.1}$$

In many cases it is more convenient to use the following (equivalent) definition.

Definition 1.2. *(Submodular function: Definition 2) A function $f : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ is submodular if:*

$$f(A + u) - f(A) \geq f(B + u) - f(B) \qquad \forall A \subseteq B \subseteq \mathcal{N}, u \in \mathcal{N} \setminus B \ . \tag{1.2}$$

The second definition illustrates an important property of submodular functions known as diminishing returns. Informally, Definition 1.2 states that adding an element to a larger set results in smaller marginal increase in the value of f (compared to adding the element to a smaller set). This property makes submodular functions natural candidates for modeling production profit (returns), utility functions, accuracy of a learning algorithm, etc.

We now prove that the above two definitions are indeed equivalent.

Lemma 1.1. *Definition 1.1 and Definition 1.2 are equivalent.*

Proof. Assume f satisfies Definition 1.1, and consider arbitrary sets $A \subseteq B \subseteq \mathcal{N}$ and element $u \in \mathcal{N} \setminus B$. By applying Definition 1.1 with $A+u$ and B, and observing that $(A+u) \cup B = B+u$ and $(A+u) \cap B = A$, we get

$$f(A+u) + f(B) \geq f((A+u) \cup B) + f((A+u) \cap B) = f(B+u) + f(A) \ .$$

Rearranging the terms yields that f satisfies Definition 1.2.

On the other hand, assume that f satisfies Definition 1.2, and consider arbitrary sets $A, B \subseteq \mathcal{N}$. Our objective is to show that $f(A)+f(B) \geq f(A \cup B)+f(A \cap B)$. We prove this claim by induction on $|A \cup B| - |A \cap B|$. If $|A \cup B| - |A \cap B| = 0$ (which implies $A = B$), then the claim is trivial. Next, suppose that $|A \cup B| - |A \cap B| = k$, and let us assume without loss of generality that there exists an element $u \in A \setminus B$. Then,

$$\begin{aligned} f(A) + f(B) &= [f(A) - f(A-u)] + [f(A-u) + f(B)] \\ &\geq [f(A) - f(A-u)] + [f((A \cup B) - u) + f(A \cap B)] \qquad (1.3) \\ &\geq [f(A \cup B) - f((A \cup B) - u)] + [f((A \cup B) - u) + f(A \cap B)] \qquad (1.4) \\ &= f(A \cup B) + f(A \cap B) \ , \end{aligned}$$

where Inequality (1.3) follows by the induction hypothesis, and inequality (1.4) follows by applying Definition 1.2. □

Recall that this chapter deals with algorithms maximizing submodular functions subject to combinatorial constraints. In addition to being submodular, the function to be maximized often has additional natural properties that can be used by the algorithm and often lead to an improved performance.

Definition 1.3. *Consider a function $f \colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$.*

- *f is* monotone *if $f(A) \leq f(B)$ for every two sets $A \subseteq B \subseteq \mathcal{N}$.*
- *f is* symmetric *if $f(A) = f(\mathcal{N} \setminus A)$ for every set $A \subseteq \mathcal{N}$.*
- *f is* normalized *if $f(\varnothing) = 0$.*

The following lemma provides basic properties of submodular functions used throughout the chapter. These properties follow quite easily from the definition of submodularity, and thus, we leave the proof of most of them to the reader. In this lemma, and in the rest of the chapter, we use the shorthand $f(u \mid A)$ to denote $f(A+u) - f(A)$ (*i.e.*, the marginal increase in $f(A)$ following the addition of u to A).

Lemma 1.2 (Basic properties)**.** *Let $f, g : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ be submodular functions, and let $B \subseteq \mathcal{N}$ be a subset of $\mathcal{N}$. Then,*

1. *For every $c \in \mathbb{R}_{\geq 0}$, $h(A) = c \cdot f(A)$ is a submodular function.*
2. *$h(A) = f(A) + g(A)$ is a submodular function.*
3. *$h(A) = f(A \cap B)$ is a submodular function.*
4. *$h(A) = f(\mathcal{N} \setminus A)$ is a submodular function.*
5. *If f is monotone and $c \in \mathbb{R}_{\geq 0}$, then $h(A) = \min\{f(A), c\}$ is a monotone submodular function.*
6. *$\sum_{u \in B} f(u \mid A) \geq f(B \cup A) - f(A)$.*

Proof. We only prove here Property 6. The proof relies on the frequently used idea of a telescopic sum. Let $B = \{u_1, \ldots, u_k\}$.

$$f(B \cup A) - f(A) = \sum_{i=1}^{k} f(u_i \mid A \cup \{u_1, \ldots, u_{i-1}\}) \leq \sum_{i=1}^{k} f(u_i \mid A) \ ,$$

where the equality follows by a telescopic sum, and the inequality follows by the submodularity of f. □

1.1.1 Examples of Submodular Functions

In this section we survey briefly several important submodular functions.

Additive function, Budget additive: Suppose each $u_i \in \mathcal{N}$ is associated with a weight $w_i \geq 0$. Then, it is easy to see that the function $f(A) = \sum_{u_i \in A} w_i$ is a normalized monotone submodular function. A slight generalization of this idea is captured by the following function, called budget additive.

$$f(A) = \min \left\{ \sum_{u_i \in A} w_i, b \right\} \qquad \text{(Budget additive function)} \tag{1.5}$$

Budget additive functions are used, for example, to model the amount of money a buyer (with budget $b \geq 0$) is willing to pay for a set of items. Lemma 1.2 shows that these function are also normalized, monotone and submodular.

Coverage function: Let $E = \{e_1, \ldots, e_m\}$ be a set of elements. Let $\mathcal{S} = \{s_1, \ldots, s_n\}$ be a collection of subsets of E (*i.e.*, $s_i \subseteq E$). We define $f : 2^{\mathcal{S}} \to \mathbb{Z}_{\geq 0}$ to be a function assigning for every subcollection of subsets the the total number of elements covered by subsets of this collection. Formally,

$$f(A) = \left| \bigcup_{s_i \in A} s_i \right| \qquad \text{(Coverage function)} \tag{1.6}$$

It is easy to see that f is normalized and monotone. Additionally, f is also submodular since, for every $A \subseteq B \subseteq \mathcal{S}$ and $s \in \mathcal{S} \setminus B$,

$$\begin{aligned} f(A+s) - f(A) &= \left| \left\{ e \in E \mid e \in s \setminus \left(\bigcup_{s_i \in A} s_i \right) \right\} \right| \\ &\geq \left| \left\{ e \in E \mid e \in s \setminus \left(\bigcup_{s_i \in B} s_i \right) \right\} \right| = f(B+s) - f(B) \ . \end{aligned}$$

Cut function: Let $G = (V, E)$ be a directed graph with capacities $c_e \geq 0$ on the edges. For every subset of vertices $A \subseteq V$, let $\delta(A) = \{e = uv \mid u \in A, v \in V \setminus A\}$. The *cut capacity function* is defined as the total capacity of edges that cross the cut $(A, V \setminus A)$. Formally,

$$f(A) = \sum_{e \in \delta(A)} c_e \qquad \text{(Cut function)} \tag{1.7}$$

Clearly f is normalized, and for undirected graphs it is also symmetric. Additionally, f is submodular for every graph since, for every $A \subseteq B \subseteq V$ and $r \in V \setminus B$,

$$\begin{aligned} f(A+r) - f(A) &= \sum_{e \in \delta(A+r)} c_e \quad - \sum_{e \in \delta(A)} c_e \\ &= \sum_{e=rv \mid v \in V \setminus A} c_e \; - \sum_{e=ur \mid u \in A} c_e \\ &\geq \sum_{e=rv \mid v \in V \setminus B} c_e \; - \sum_{e=ur \mid u \in B} c_e \qquad (1.8) \\ &= \sum_{e \in \delta(B+r)} c_e \quad - \sum_{e \in \delta(B)} c_e = f(B+r) - f(B) \;, \end{aligned}$$

where Inequality (1.8) follows since we have both $\{e = rv | v \in V \setminus B\} \subseteq \{e = rv | v \in V \setminus A\}$ and $\{e = ur | u \in A\} \subseteq \{e = ur | u \in B\}$.

Rank function: Let $\{a_1, \ldots, a_m\}$ be a set of vectors in $\mathbb{R}^n$. For a subset $A \subseteq \{a_1, \ldots, a_m\}$ we define the *rank* of A as the dimension of the vector space spanned by the vectors in A. Let,

$$f(A) = \text{rank}(A) \qquad \text{(Independent set rank function)} \tag{1.9}$$

The function f is a normalized monotone submodular function, where submodularity follows by basic properties of linear algebra. In fact, this function is a special case of a more general family of matroid rank functions (which is outside the scope of this chapter). Informally, matroids are important combinatorial objects generalizing of the notion of linear independence of vector spaces. Matroids and submodular functions are closely related. In particular, each matroid is associated with a submodular rank function.

1.1.2 Maximizing Submodular Functions

Let $f \colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ be a non-negative submodular function. The basic problem we discuss in this chapter is maximizing the function f subject to a (possibly empty) set of constraints. Formally, let $\mathcal{I}$ be the set of subsets of $\mathcal{N}$ obeying the constraint. We are interested in the following problem.

$$\begin{array}{ll} \max & f(A) \\ \text{s.t.} & A \in \mathcal{I} \subseteq 2^{\mathcal{N}} \end{array} \tag{1.10}$$

We denote by $OPT \in \mathcal{I}$ a feasible subset with maximum value, and look for algorithms that can find a set whose value approximates the optimal value $f(OPT)$. Below we survey briefly important special cases of this problem, however, we first consider two technical issues with the above problem. First, we note that we restrict ourselves to non-negative functions. This restriction is necessary since we will be looking for algorithms with a multiplicative approximation ratio for

this problem. Moreover, as seen by the above examples, many important submodular functions are normalized and monotone, which implies that they are also non-negative.

The second technical issue is that the representation of a general submodular function might be exponential in the size $|\mathcal{N}|$ of the ground set (we reserve n throughout the chapter to denote this size). To deal with this issue the algorithms we present assume access to a procedure that given any subset $A \subseteq \mathcal{N}$ returns the value $f(A)$. Such a procedure is called *value oracle*. In many cases we measure the complexity of algorithms by the number of oracle queries they performs. To obtain the real time complexity one needs to take into account the time complexity of the value oracle itself, as well as the time required for additional calculations performed by the algorithm. However, counting the number of value oracle queries is a clean measure that in many cases represents (and dominates up to poly-logarithmic terms) the actual time complexity of the algorithm.

As promised, we now survey special cases of the above basic problem.

Maximum coverage problem: Consider the coverage function (1.6). The *maximum coverage problem* is the problem of maximizing $f(A)$ subject to the constraint $|A| \leq k$ when f is a coverage function and k is a parameter. In other words, given a set of elements and a collection of subsets of these elements, we are looking for a sub-collection of k subsets maximizing the number of covered elements. The constraint that allows A to be of size at most k is called a *cardinality constraint*. Section 1.2.1 discusses in details algorithms for maximizing general non-negative submodular functions subject to a cardinality constraint.

Maximum cut problem: Consider the cut function (1.7). The *maximum cut problem* is the problem of maximizing $f(A)$ when f is a cut function. In other words, given a (possibly directed) graph $G = (V, E)$ with capacities on its edges we are interested in finding a subset A of the vertices that maximizes the total capacity of the edges crossing the cut $(A, V \setminus A)$. The maximum cut problem is a special case of the more general unconstrained submodular maximization problem in which one looks for an arbitrary set maximizing a given non-negative submodular function. We discuss this problem in details in Section 1.2.2.

Submodular welfare: In the `Submodular Welfare` problem there is a set of items, $\mathcal{N}^{\texttt{SW}}$, that we would like to allocate to a set of k buyers. Each buyer i is associated with a monotone submodular function $f_i : 2^{\mathcal{N}^{\texttt{SW}}} \rightarrow \mathbb{R}_{\geq 0}$ representing her utility from each subset of items. The problem is to partition the items between the buyers so as to maximize the total welfare measured as $\sum_{i=1}^{k} f_i(A_i)$, where A_i is the subset of items allocated to buyer i. At first sight, it looks like this problem is not a special case of the general problem (1.10). To see that `SW` can, in fact, be presented as a special case of this general problem, consider an instance of `SW`, and let us define a new ground set $\mathcal{N} = \mathcal{N}^{\texttt{SW}} \times \{1, 2, \ldots, k\}$, where $\mathcal{N}^{\texttt{SW}}$ is the ground set of the `SW` instance and k is the number of buyers in this instance. Intuitively, each element (u, i) of $\mathcal{N}$ should be understood as corresponding to assigning item u to buyer i. Using this interpretation, the objective function of the `SW` instance can be formulated as

$$f(A) = \sum_{i=1}^{k} f_i(\{u \in \mathcal{N}^{\texttt{SW}} : (u, i) \in A\}) \quad \forall\, A \subseteq \mathcal{N} \ .$$

It is not difficult to verify that the above function f is non-negative, monotone and submodular whenever the utility functions of the buyers have these properties. Additionally, the requirement of `SW` that each element is assigned to at most one buyer can be captured by the general problem (1.10) by choosing $\mathcal{I} = \{A \subseteq \mathcal{N} : |A \cap (\{u\} \times \{1, 2, \ldots, k\})| \leq 1 \ \ \forall u \in \mathcal{N}^{\texttt{SW}}\}$, which means that a solution

is feasible if and only if it contains at most one pair (u, i) involving every given element $u \in \mathcal{N}^{\mathtt{SW}}$. We discuss the `Submodular Welfare` problem further in Sections 1.2.3, 1.3.2 and 1.3.3.

1.1.3 Organization

In Section 1.2 we discuss basic discrete (combinatorial) algorithms for maximizing submodular functions subject to various constraints. In Section 1.2.1 we discuss maximization subject to a cardinality constraint. In Section 1.2.2 we discuss the unconstrained submodular maximization problem. Finally, in Section 1.2.3 we discuss a discrete algorithm for the submodular welfare problem.

In Section 1.3 we discuss continuous methods that are based on extensions of submodular functions to the n-dimensional cube. We discuss the basic properties of these extensions in Section 1.3.1. Section 1.3.2 and Section 1.3.3 describe basic algorithms obtaining fractional solutions which approximately maximize these extensions. Rounding these fractional solutions to obtain integeral solutions is discussed both in Sections 1.3.2 and 1.3.4. Finally, Section 1.3.5 describes a method for obtaining inapproximability results for submodular maximization problems which is based on similar techniques.

1.2 Discrete Greedy Algorithms

In this section we present discrete (combinatorial) algorithms for three submodular maximization problems: maximizing a submodular function subject to a cardinality constraint, unconstrained submodular maximization and submodular welfare.

1.2.1 Cardinality Constraint

One of the first submodular maximization problems that was considered is the problem of maximizing a non-negative monotone submodular function subject to a cardinality constraint. More specifically, in this problem the input is a non-negative monotone submodular function $f\colon \mathcal{N} \to \mathbb{R}_{\geq 0}$ and an integer k between 1 and n. The objective of the problem is to find a set A of size at most k maximizing f.

A natural greedy algorithm for this problem starts with the empty set, and then adds elements to it in k iterations. In each iteration the algorithm adds to its current solution the single element increasing the value of this solution by the most (*i.e.*, the element with the largest marginal value with respect to the current solution). In the context of submodular maximization this simple algorithm is usually referred to simply as "the greedy algorithm". A formal statement of the greedy algorithm is given as Algorithm 1. It is important to note that Algorithm 1 denotes by A_i the solution produced at the end of iteration i. The use of a different variable for the solution produced at the end of each iteration is not necessary for the algorithm, but is convenient for the analysis.

The approximation ratio of the greedy algorithm is $1 - 1/e \approx 0.632$ [1], and this turns out to be the best approximation ratio possible for the problem [2]. Here we give only the analysis of the approximation ratio of the greedy algorithm. More information about the matching hardness result can be found in Section 1.3.5.

Theorem 1.3 (Due to [1]). *The greedy algorithm is a $(1 - 1/e)$-approximation algorithm for maximizing a non-negative monotone submodular function subject to a cardinality constraint.*

Algorithm 1: The Greedy Algorithm(f, k)

```
Let A_0 ← ∅.
for i = 1 to k do
    Let u_i be the element of 𝒩 maximizing f(u_i | A_{i-1}).
    Let A_i ← A_{i-1} + u_i.
return A_k
```

Proof. Observe that, for every $1 \leq i \leq k$,

$$\begin{aligned} f(u_i \mid A_{i-1}) &\geq \max_{u \in OPT} f(u \mid A_{i-1}) \geq \frac{\sum_{u \in OPT} f(u \mid A_{i-1})}{k} \\ &\geq \frac{f(OPT \cup A_{i-1}) - f(A_{i-1})}{k} \geq \frac{f(OPT) - f(A_{i-1})}{k} \ , \end{aligned}$$

where the first inequality follows since the element u_i is chosen as the element with the largest marginal contribution with respect to A_{i-1}, the second inequality holds by an averaging argument since OPT contains at most k elements and the third inequality follows from the submodularity of f (Lemma 1.2, Property 6). Finally, the last inequality follows from the monotonicity of f.

Recall that $A_i = A_{i-1} + u_i$, and hence, $f(u_i \mid A_{i-1}) = f(A_i) - f(A_{i-1})$. Plugging this equality into the previous inequality gives

$$f(A_i) - f(A_{i-1}) \geq \frac{f(OPT) - f(A_{i-1})}{k} \ .$$

The last inequality states that $f(A_i)$ improves over $f(A_{i-1})$ by at least $1/k$ of the gap between $f(A_{i-1})$ and OPT. Intuitively, it is clear that combining this inequality for all $1 \leq i \leq k$ should give a lower bound on $f(A_k)$ (and thus, also on the approximation ratio of the greedy algorithm). To derive this bound formally we rearrange the last inequality as follows.

$$f(OPT) - f(A_i) \leq (1 - 1/k) \cdot \big[f(OPT) - f(A_{i-1})\big] \ .$$

Combing the last inequality for every $1 \leq i \leq k$ gives

$$f(OPT) - f(A_k) \leq (1 - 1/k)^k \cdot \big[f(OPT) - f(A_0)\big]$$

Finally, we rearrange again and get:

$$f(A_k) \geq f(OPT) - (1 - 1/k)^k \cdot \big[f(OPT) - f(A_0)\big] \geq (1 - 1/e) \cdot f(OPT) \ ,$$

where the second inequality follows from the non-negativity of f and since $(1 - 1/k)^k \leq 1/e$. □

The greedy algorithm is often used in practice also when the submodular objective function f is not monotone. However, from a theoretical point of view the greedy algorithm has no constant approximation guarantee in this case. This is demonstrated, for example, by a non-negative submodular function $f \colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ defined as follows.

$$f(A) = \begin{cases} |A| & \text{if } u_n \notin A \ , \\ 2 & \text{otherwise} \ . \end{cases}$$

It can be verified that f is a non-negative (non-monotone) submodular function. The value of f for every set is $|A|$ unless the set contains u_n. If the set contains u_n then its value is 2, regardless of the presence of other elements in the set.

Assume the greedy algorithm is faced with the above function f and some value $2 \leq k < n$. In the first iteration of the greedy algorithm the element u_n has a marginal gain of 2, which is larger than the marginal gains of the other elements. Hence, the greedy algorithm takes the "bait" and adds u_n to its solution. This means that u_n remains in the solution of the greedy algorithm until the end, and thus, the value of this solution is 2 (notice that this is true even if one tries to improve the greedy algorithm by allowing it to stop as soon as no element has a positive marginal contribution). On the other hand, the optimal solution can take any k elements other than u_n and get a value of k. Thus, the approximation ratio of greedy is at most $2/k$.

The non-constant approximation ratio of the greedy algorithm for non-monotone functions motivated [3] to suggest a randomized version of the greedy algorithm called *Random Greedy*. Intuitively, Random Greedy tries to avoid "baits" by introducing randomization into the elements selection logic. A formal statement of Random Greedy appears as Algorithm 2.

Algorithm 2: Random Greedy(f, k)

Let $A_0 \leftarrow \varnothing$.
for $i = 1$ **to** k **do**
  Let $M_i = \arg\max_{B:|B|\leq k} \left\{\sum_{u\in B} f(u \mid A_{i-1})\right\}$.
  with *probability* $(1 - |M_i|/k)$ **do** $A_i \leftarrow A_{i-1}$.
  otherwise Let u_i be a uniformly random element of M_i, and set $A_i \leftarrow A_{i-1} + u_i$.
return A_k

Random Greedy finds in every iteration the set M of size at most k maximizing the total marginal contribution of the elements in the set with respect to the current solution. Then, the algorithm adds at most one element of M to its current solution. The algorithm picks the element to be added in such a way that every element of M has a probability of exactly $1/k$ to be added to the solution. Notice that this implies that, when $|M| < k$, there is a positive probability that the solution of the algorithm is unchanged by the iteration. This might be considered wasteful, however, avoiding this wastefulness does not improve the theoretical guarantee of the algorithm as far as we know.

Lemma 1.4. *For every* $0 \leq i \leq k$, $\mathbb{E}[f(A_i \cup OPT)] \geq (1 - 1/k)^i \cdot f(OPT)$.

Proof. Consider any $1 \leq j \leq k$, and let E_{j-1} be an arbitrary possible choice for the random decisions of Random Greedy during its first $j-1$ iterations. In the first part of this proof we implicitly condition on E_{j-1} all the expectations and random quantities (*i.e.*, we fix the random choices of the algorithm in the first $j-1$ iterations, and consider the distribution induced by its remaining random choices). Since A_{j-1} is now deterministic, we get

$$\begin{aligned}
\mathbb{E}[f(A_j \cup OPT)] - f(A_{j-1} \cup OPT) &= \frac{1}{k} \sum_{u \in M_j} f(u \mid A_{j-1} \cup OPT) \\
&\geq \frac{1}{k} \left[f(M_j \cup A_{j-1} \cup OPT) - f(A_{j-1} \cup OPT)\right] \\
&\geq -\frac{1}{k} f(A_{j-1} \cup OPT) \ ,
\end{aligned}$$

where the equality follows since each element of M_j is added to the solution with probability $1/k$, the first inequality follows by submodularity and the last inequality follows by the non-negativity of f. Rearraning the last inequality yields

$$\mathbb{E}[f(A_j \cup OPT)] \geq \left(1 - \frac{1}{k}\right) \cdot f(A_{j-1} \cup OPT) \ .$$

We now remove the implicit conditioning on E_{j-1}. Taking an expectation over all the possible ways to choose E_{j-1}, the last inequality yields

$$\mathbb{E}[f(A_j \cup OPT)] \geq \left(1 - \frac{1}{k}\right) \cdot \mathbb{E}[f(A_{j-1} \cup OPT)] \ .$$

Combining the above inequality for every $1 \leq j \leq i$ implies the lemma since $A_0 \cup OPT = \varnothing \cup OPT = OPT$. □

We are now ready to analyze the approximation ratio of Random Greedy.

Theorem 1.5 (Due to [3])**.** *Random greedy is a $1/e$-approximation algorithm for maximizing a non-negative submodular function subject to a cardinality constraint. If f is monotone the approximation ratio of random greedy is $1 - 1/e$.*

Proof. As in the proof of Lemma 1.4, consider any $1 \leq i \leq k$, and let E_{i-1} be an arbitrary possible choice for the random decisions of Random Greedy during its first $i-1$ iterations. The following inequality implicitly condition on E_{i-1} in the expectation and random quantities Observe that

$$\mathbb{E}[f(u_i \mid A_{i-1})] = \frac{\sum_{u \in M_i} f(u \mid A_{i-1})}{k} \geq \frac{\sum_{u \in OPT} f(u \mid A_{i-1})}{k} \geq \frac{f(A_{i-1} \cup OPT) - f(A_{i-1})}{k} \ ,$$

where the first inequality follows from the definition of M_i as the set that maximizes the total marginal contribution, and the second from the submodularity of f. We now remove the implicit conditioning on E_{i-1}. Taking an expectation over all the possible ways to choose E_{i-1}, the last inequality yields

$$\mathbb{E}[f(u_i \mid A_{i-1})] \geq \frac{\mathbb{E}[f(A_{i-1} \cup OPT)] - \mathbb{E}[f(A_{i-1})]}{k} \geq \frac{(1 - 1/k)^{i-1} \cdot f(OPT) - \mathbb{E}[f(A_{i-1})]}{k} \ ,$$

where the second inequality is due to Lemma 1.4. Note that if f is monotone then $f(A_{i-1} \cup OPT) \geq f(OPT)$, and thus, in this case we get the same bound as in Theorem 1.3. Hence, if we continued the analysis for monotone f we would have got the same bound of $1 - 1/e$ as in Theorem 1.3.

For non-monotone functions, we observe that $\mathbb{E}[f(u_i \mid A_{i-1})]$ is the expected increase in $f(A_i)$ compared to $f(A_{i-1})$, and thus, the last inequality gives a lower bound on the improvement in the expected value of the algorithm's solution in each iteration. This bound is given in terms of $f(OPT)$ and the expected value of the current solution. The reminder of the proof "translates" this lower bound into an explicit lower bound on the expected value of the algorithm's solution after any given number of iterations. More specifically, we prove by induction that $\mathbb{E}[f(A_i)] \geq \frac{i}{k} \cdot (1 - \frac{1}{k})^{i-1} \cdot f(OPT)$. For $i = 0$, this is true since $f(A_0) \geq 0 = \frac{0}{k} \cdot (1 - \frac{1}{k})^{-1} \cdot f(OPT)$.[1] Assume

[1] For $k = 1$ the term $(1 - 1/k)^{-1}$ is undefined. However, one can verify that Random Greedy is in fact optimal for $k = 1$.

now that the claim holds for every $0 \leq i' < i$, and let us prove it for $i > 0$.

$$\begin{aligned}
\mathbb{E}[f(A_i)] &= \mathbb{E}[f(A_{i-1})] + \mathbb{E}[f(u_i \mid A_{i-1})] \geq \mathbb{E}[f(A_{i-1})] + \frac{(1-1/k)^{i-1} \cdot f(OPT) - \mathbb{E}[f(A_{i-1})]}{k} \\
&= \left(1 - \frac{1}{k}\right) \cdot \mathbb{E}[f(A_{i-1})] + \frac{(1-1/k)^{i-1}}{k} \cdot f(OPT) \\
&\geq \left(1 - \frac{1}{k}\right) \cdot \left[\frac{i-1}{k} \cdot \left(1 - \frac{1}{k}\right)^{i-2} \cdot f(OPT)\right] + \frac{(1-1/k)^{i-1}}{k} \cdot f(OPT) \\
&= \frac{i}{k} \cdot \left(1 - \frac{1}{k}\right)^{i-1} \cdot f(OPT) \ .
\end{aligned}$$

Plugging $i = k$ into the above claim yields the theorem.

$$\mathbb{E}[f(A_k)] \geq \frac{k}{k} \cdot (1 - 1/k)^{k-1} \cdot f(OPT) \geq 1/e \cdot f(OPT) \ . \qquad \square$$

When the function f is monotone both greedy and random greedy achieve an approximation ratio of $1-1/e$. It can also be verified that both algorithms require $O(n)$ value oracle queries in each one of their k iterations, and thus, require $O(nk)$ queries in total. In the case of Random Greedy using $O(n)$ value oracle queries in each iteration is a bit wasteful. More specifically, Random Greedy queries the marginal contributions of all the elements in order to construct all of M, and then chooses a uniformly random element out of M (note that when f is monotone, we can assume M always contains exactly k elements). Instead, one can query the marginal contributions of a random subset of the elements, and use it to recover, with high enough probability, at least one element of M (*i.e.*, to find an element whose marginal contribution, with respect to the current solution, is among the top k marginal contributions). This is the idea behind Algorithm 3, which we term here Sample Greedy, originally suggested independently by [4] and [5].

Algorithm 3: Sample Greedy(f, k, ε)

```
Let A_0 ← ∅.
for i = 1 to k do
    Let B_i be a uniformly random subset of N of size ⌈(n·ln(1/ε))/k⌉.
    Let u_i be the element of B_i maximizing f(u_i | A_i).
    Let A_i ← A_{i-1} + u_i.
return A_k
```

Sample Greedy has a parameter $\varepsilon \in [e^{-k}, 1 - 1/e]$ controlling a tradeoff between the number of value oracle queries used by the algorithm and the quality of its output. This tradeoff is captured by the following theorem.

Theorem 1.6 (Due to [4, 5])**.** *Sample Greedy is a $(1 - 1/e - \varepsilon)$-approximation algorithm for maximizing a non-negative monotone submodular function subject to a cardinality constraint which uses $O(n \ln(1/\varepsilon))$ value oracle queries.*[2]

[2]Observe that the restriction of ε to the range $[e^{-k}, 1 - 1/e]$ is not an issue because Theorem 1.6 gives no approximation guarantee for $\varepsilon > 1 - e^{-1}$, and its guarantee for the number of value oracle queries is no better than that of Random Greedy for $\varepsilon < e^{-k}$.

The analysis of the number of value oracle queries used by Sample Greedy is quite straightforward, and thus, we concentrate on proving its approximation ratio. However, we would like to direct the reader's attention to the fact that the number of queries used by Sample Greedy is independent of k, which is quite surprising. The following lemma lower bounds the expected improvement in the value of the solution of Sample Greedy after every given iteration.

Lemma 1.7. *For every* $1 \leq i \leq k$, $\mathbb{E}[f(u_i \mid A_{i-1})] \geq (1-\varepsilon) \cdot \frac{f(OPT)-\mathbb{E}[f(A_{i-1})]}{k}$.

Proof. Let E_{i-1} be an event specifying the random decisions of Sample Greedy up to iteration $i-1$ (including). If the lemma holds conditioned on every given event E_{i-1}, then it holds also unconditionally. Hence, in the rest of the proof we fix an event E_{i-1} and prove the lemma conditioned on this event. For simplicity, all the probabilities, expectations and random quantities in the proof are implicitly conditioned on E_{i-1}. In particular, note that the set A_{i-1} is now deterministic since it is fully determined by E_{i-1}.

Let $v_1, v_2, \ldots, v_k$ be the k elements with the largest marginal contributions with respect to A_{i-1}, sorted in a non-increasing marginal contribution order. Additionally, let X_j be an indicator for the event that Sample Greedy selects v_j in its i-th iteration (*i.e.*, $u_i = v_j$). Using this notation and the linearity of the expectation, it is possible to lower bound $\mathbb{E}[f(u_i \mid A_{i-1})]$ as follows.

$$\mathbb{E}[f(u_i \mid A_{i-1})] \geq \mathbb{E}\left[\sum_{j=1}^{k} X_j \cdot f(v_j \mid A_{i-1})\right] = \sum_{j=1}^{k} \mathbb{E}[X_j] \cdot f(v_j \mid A_{i-1}) \ .$$

Recall that u_i is selected as the element with the highest marginal contribution in B_i; hence, when multiple elements of $v_1, v_2, \ldots, v_k$ belong to B_i, Sample Greedy selects the element with the lowest index among them. This implies that $\mathbb{E}[X_i] = \Pr[u_i = v_j]$ is a non-increasing function of j. Consider now the sum on the rightmost hand side of the above inequality. Every term of this sum is a multiplication of two non-increasing functions of j: $\mathbb{E}[X_j]$ and $f(v_j \mid A_{i-1})$. Thus, by Chebyshev's sum inequality, this rightmost hand side can be bounded by

$$\mathbb{E}[f(u_i \mid A_{i-1})] \geq \sum_{j=1}^{k} \mathbb{E}[X_j] \cdot \frac{\sum_{j=1}^{k} f(v_j \mid A_{i-1})}{k} \ . \tag{1.11}$$

Next, observe that,

$$\sum_{j=1}^{k} \mathbb{E}[X_j] = \Pr[\{v_1, v_2, \ldots, v_k\} \cap B_i \neq \varnothing] \geq 1 - \left(1 - \frac{k}{n}\right)^{\left\lceil \frac{n \cdot \ln(1/\varepsilon)}{k} \right\rceil} \geq 1 - \varepsilon \ .$$

The first equality follows since one of variables X_j is 1 if and only if $\{v_1, v_2, \ldots, v_k\} \cap B_i \neq \varnothing$, and the first inequality follows since the probability of choosing an element from $\{v_1, v_2, \ldots, v_k\}$ when the sample set B_i is chosen with repetitions is smaller than this probability when the set is chosen with no repetitions. Plugging the last inequality into Inequality (1.11) yields

$$\mathbb{E}[f(u_i \mid A_{i-1})] \geq (1-\varepsilon) \cdot \frac{\sum_{j=1}^{k} f(v_j \mid A_{i-1})}{k} \ .$$

The lemma now follows by observing that the definition of the v_j's and the submodularity and monotonicity of f imply together

$$\sum_{j=1}^{k} f(v_j \mid A_{i-1}) \geq \sum_{u \in OPT} f(u \mid A_{i-1}) \geq f(OPT \cup A_{i-1}) - f(A_{i-1}) \geq f(OPT) - f(A_{i-1}) \ . \quad \square$$

We are now ready to prove the approximation ratio of Sample Greedy.

Proof of the Approximation Ratio of Sample Greedy. Let us denote $\alpha = (1-\varepsilon)/k$. Then, by Lemma 1.7, for every $1 \leq i \leq k$,

$$\mathbb{E}[f(A_i) - f(A_{i-1})] = \mathbb{E}[f(u_i \mid A_{i-1})] \geq \alpha[f(OPT) - \mathbb{E}[f(A_{i-1})]] \ .$$

Rearranging gives

$$f(OPT) - \mathbb{E}[f(A_i)] \leq (1-\alpha) \cdot [f(OPT) - \mathbb{E}[f(A_{i-1})]] \ .$$

Combining the above inequality for every $1 \leq i \leq k$ yields

$$f(OPT) - \mathbb{E}[f(A_k)] \leq (1-\alpha)^k \cdot [f(OPT) - \mathbb{E}[f(A_0)]] \leq (1-\alpha)^k \cdot f(OPT) \ ,$$

where the last inequality follows from the non-negativity of f. Rearranging once more, we get

$$\begin{aligned}\mathbb{E}[f(A_k)] &\geq \left[1 - (1-\alpha)^k\right] \cdot f(OPT) \geq (1 - e^{-k\cdot\alpha}) \cdot f(OPT) \\ &= (1 - e^{\varepsilon - 1}) \cdot f(OPT) \geq (1 - 1/e - \varepsilon) \cdot f(OPT) \ ,\end{aligned}$$

where the last inequality holds for every $\varepsilon \in [0,1]$. □

1.2.2 Unconstrained Maximization

Perhaps the simplest variant of a submodular maximization problem is *unconstrained submodular maximization*, which is a problem asking to maximize a submodular function subject to no constraints on the feasible subset we may choose. When the function f is monotone the solution is trivial—simply take all elements to the solution. However, the problem is non-trivial when the function is non-monotone. We present in this section two algorithms for the problem. Amazingly, the first algorithm does not even evaluate the function. It simply takes each element to its solution with probability 1/2. We prove that this algorithm achieves an approximation ratio of 1/4. This result is due to [6], but the proof we present here for this result is different from the original proof of [6]. The advantage of the proof we present is that it provides us with some basic tools that are used later for analyzing the second algorithm. The second algorithm itself is a greedy algorithm which was shown to have an approximation ratio of 1/2 by [7]. A matching inapproximability result, showing that no polynomial time algorithm can achieve an approximation ratio of $1/2 + \varepsilon$ for any constant $\varepsilon > 0$, was proved by [6].

Let us begin with the analysis of the first algorithm. Recall that this algorithm simply takes every element to the solution with probability 1/2, independently. For the analysis's purpose we give a more verbose description of this algorithm as Algorithm 4. While reading Algorithm 4 it is important to recall that $u_1, u_2, \ldots, u_n$ is some (arbitrary) ordering of the elements of $\mathcal{N}$.

Notice that the algorithm maintains two random solutions. The values of these solutions after i iterations are denoted by X_i and Y_i. It can be easily seen that X_i and Y_i always agree on the first i elements, and thus, at the end of the execution $X_n = Y_n$. To analyze the performance of the algorithm we need some additional notation. Let $OPT_i \triangleq (OPT \cup X_i) \cap Y_i$. By the above discussion, we have $OPT_0 = OPT$ and $OPT_n = X_n = Y_n$. Similarly, let $\overline{OPT} = \mathcal{N} \setminus OPT$ and let $\overline{OPT}_i \triangleq (\overline{OPT} \cup X_i) \cap Y_i$, then $\overline{OPT}_0 = \overline{OPT}$ and $\overline{OPT}_n = X_n = Y_n$.

One can view $f(OPT_i)$, $f(\overline{OPT}_i)$, $f(X_i)$ and $f(Y_i)$ as functions of i. The following lemma takes this point of view and bounds the expected decrease of value in OPT_i and $\overline{OPT}_i$ when i increases by the total expected increase in the values of X_i and Y_i.

Algorithm 4: Random Sample(f)

Let $X_0 \leftarrow \varnothing, Y_0 \leftarrow \mathcal{N}$.
for $i = 1$ **to** n **do**
  with *probability* $1/2$ **do** $X_i \leftarrow X_{i-1} + u_i$, $Y_i \leftarrow Y_{i-1}$.
  otherwise $X_i \leftarrow X_{i-1}$, $Y_i \leftarrow Y_{i-1} - u_i$. `// Also done with probability 1/2.`
return X_n *(or equivalently* Y_n*).*

Lemma 1.8. *For every* $1 \leq i \leq n$,

$$\mathbb{E}[f(OPT_{i-1}) - f(OPT_i) + f(\overline{OPT}_{i-1}) - f(\overline{OPT}_i)] \leq \mathbb{E}[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})] \ , \tag{1.12}$$

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

Proof. Notice that it suffices to prove Inequality (1.12) conditioned on any event of the form $X_{i-1} = A_{i-1}$ for which the probability that $X_{i-1} = A_{i-1}$ is non-zero (notice that this can only happen when $A_{i-1} \subseteq \{u_1, u_2, \ldots, u_{i-1}\}$). Fix such an event. The rest of the proof implicitly assumes everything is conditioned on this event. Note that, due to the conditioning, the random variables Y_{i-1}, OPT_{i-1} and $\overline{OPT}_{i-1}$ become deterministic.

If $u_i \notin OPT$ (and, hence, $u_i \in \overline{OPT}$), then

$$\begin{aligned}
&\mathbb{E}[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})] \\
&\qquad = \frac{1}{2}\left[f(X_{i-1} + u_i) - f(X_{i-1}) + f(Y_{i-1} - u_i) - f(Y_{i-1})\right] \\
&\qquad \geq \frac{1}{2}\left[f(\overline{OPT}_{i-1}) - f(\overline{OPT}_{i-1} - u_i) + f(OPT_{i-1}) - f(OPT_{i-1} + u_i)\right] \\
&\qquad = \mathbb{E}[f(OPT_{i-1}) - f(OPT_i) + f(\overline{OPT}_{i-1}) - f(\overline{OPT}_i)] \ .
\end{aligned}$$

To see that the inequality follows by submodularity notice that $X_{i-1} \subseteq \left[(\overline{OPT} \cup X_{i-1}) \cap Y_{i-1}\right] - u_i = \overline{OPT}_{i-1} - u_i$ and $OPT_{i-1} = (OPT \cup X_{i-1}) \cap Y_{i-1} \subseteq Y_{i-1} - u_i$ (since $u_i \notin OPT$).

The proof that the lemma holds also when $u_i \in OPT$ is similar, and is, therefore, omited. □

The performance of the algorithm easily follows from Lemma 1.8.

Theorem 1.9 (Due to [6])**.** *Algorithm 4 is a randomized linear time* $1/4$*-approximation algorithm for the* unconstrained submodular maximization *problem. If the function* f *is symmetric, in addition to being submodular, then the algorithm is a* $1/2$*-approximation algorithm.*

Proof. Summing up the inequality in Lemma 1.8 for every $1 \leq i \leq n$ we get

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

The above sum is telescopic. Collapsing it yields

$$\mathbb{E}[f(OPT_0) - f(OPT_n) + f(\overline{OPT}_0) - f(\overline{OPT}_n)] \leq \mathbb{E}[f(X_n) - f(X_0) + f(Y_n) - f(Y_0)]$$

Using the non-negativity of f, and recalling the values of OPT_0, OPT_n, $\overline{OPT}_0$ and $\overline{OPT}_n$, we obtain

$$\mathbb{E}[f(X_n)] = \mathbb{E}[f(Y_n)] \geq \frac{1}{4}\left(f(OPT) + f(\overline{OPT}) + f(\varnothing) + f(\mathcal{N})\right) \geq \frac{1}{4} \cdot f(OPT) \ .$$

If f is also symmetric then $f(\overline{OPT}) = f(OPT)$, and the right hand side of the above inequality can be improved to $1/2 \cdot f(OPT)$. □

We next present a $1/2$-approximation algorithm for unconstrained submodular maximization. The algorithm is formally described in Algorithm 5.

Algorithm 5: Double Greedy(f)

Let $X_0 \leftarrow \varnothing, Y_0 \leftarrow \mathcal{N}$.
for $i = 1$ **to** n **do**
  Let $a_i \leftarrow \max\{f(X_{i-1} + u_i) - f(X_{i-1}), 0\}$.
  Let $b_i \leftarrow \max\{f(Y_{i-1} - u_i) - f(Y_{i-1}), 0\}$.
  with *probability* $a_i/(a_i + b_i)^*$ **do** $X_i \leftarrow X_{i-1} + u_i$, $Y_i \leftarrow Y_{i-1}$.
  otherwise $X_i \leftarrow X_{i-1}$, $Y_i \leftarrow Y_{i-1} - u_i$. `// Done with probability` $b_i/(a_i + b_i)$.
return X_n *(or equivalently* Y_n*).*

* *If* $a_i = b_i = 0$*, we assume* $a_i/(a_i + b_i) = 1$.

Like in the previous algorithm, X_i and Y_i are random variables denoting the two solutions maintained by the algorithm after i iterations. The behavior of these random variables is very similar to their behavior in Algorithm 4, except that now the decision whether to add the element u_i to X_{i-1} or to remove it from Y_{i-1} is done with probabilities that depend on the function f. The analysis of the algorithm requires the random variable OPT_i, which is defined exactly like in the analysis of Algorithm 4.

The following lemma plays the same role as Lemma 1.8.

Lemma 1.10. *For every* $1 \leq i \leq n$,

$$\mathbb{E}[f(OPT_{i-1}) - f(OPT_i)] \leq \frac{1}{2} \cdot \mathbb{E}\left[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})\right] \ , \tag{1.13}$$

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

Proof. Again, notice that it suffices to prove Inequality (1.13) conditioned on any event of the form $X_{i-1} = A_{i-1}$ for which the probability that $X_{i-1} = A_{i-1}$ is non-zero (recall that this can only happen when $A_{i-1} \subseteq \{u_1, u_2, \ldots, u_{i-1}\}$). Fix such an event. The rest of the proof implicitly assumes everything is conditioned on this event. Notice that, as before, due to the conditioning, the random variables Y_{i-1} and OPT_{i-1} become deterministic once we fix the event. Moreover, the values a_i, b_i also become deterministic when we do it. We claim that

$$\begin{aligned} &\mathbb{E}\left[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})\right] \\ &= \frac{a_i}{a_i + b_i}\left(f(X_{i-1} + u_i) - f(X_{i-1})\right) + \frac{b_i}{a_i + b_i}\left(f(Y_{i-1} - u_i) - f(Y_{i-1})\right) = \frac{a_i^2 + b_i^2}{a_i + b_i} \ . \end{aligned} \tag{1.14}$$

The first equality follows by definition. To see why the second equality holds observe that, by submodularity, $f(X_{i-1} + u_i) - f(X_{i-1}) + f(Y_{i-1} - u_i) - f(Y_{i-1}) \geq 0$ since $X_{i-1} \subseteq Y_{i-1} - u_i$. This means that $f(X_{i-1} + u_i) - f(X_{i-1}) < 0$ implies $a_i/(a_i + b_i) = 0$ and $f(Y_{i-1} - u_i) - f(Y_{i-1}) < 0$ implies $b_i/(a_i + b_i) = 0$.

Let us now analyze the expected change in OPT_i. If $u_i \notin OPT$, then $OPT_{i-1} - u_i = OPT_{i-1}$. Thus,

$$\begin{aligned}\mathbb{E}[f(OPT_{i-1}) - f(OPT_i)] &= \frac{a_i}{a_i + b_i} \cdot [f(OPT_{i-1}) - f(OPT_{i-1} + u_i)] \\ &\leq \frac{a_i}{a_i + b_i} \cdot [f(Y_{i-1} - u_i) - f(Y_{i-1})] \leq \frac{a_i b_i}{a_i + b_i} \ ,\end{aligned}$$

where the first inequality follows by the submodularity of f as $OPT_{i-1} = (OPT \cup X_{i-1}) \cap Y_{i-1} \subseteq Y_{i-1} - u_i$ (since $u_i \notin OPT$). On the other hand, if $u_i \in OPT$ then $OPT_{i-1} + u_i = OPT_{i-1}$. Thus,

$$\begin{aligned}\mathbb{E}[f(OPT_{i-1}) - f(OPT_i)] &= \frac{b_i}{a_i + b_i} \cdot [f(OPT_{i-1}) - f(OPT_{i-1} - u_i)] \\ &\leq \frac{b_i}{a_i + b_i} \cdot [f(X_{i-1} + u_i) - f(X_{i-1})] \leq \frac{a_i b_i}{a_i + b_i} \ ,\end{aligned}$$

where the first inequality follows by again the submodularity of f as $X_{i-1} \subseteq (OPT \cup X_{i-1}) \cap Y_{i-1} - u_i = OPT_{i-1} - u_i$.

The lemma follows by combining the last two inequalities with Inequality (1.14) since, for every $a, b \in \mathbb{R}$, $2ab \leq a^2 + b^2$. □

Using Lemma 1.10 the next theorem follows easily.

Theorem 1.11 (Due to [7])**.** *Algorithm 5 is a randomized linear time* 1/2*-approximation algorithm for the* unconstrained submodular maximization *problem.*

Proof. Summing up Lemma 1.10 for every $1 \leq i \leq n$ gives

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

The above sum is telescopic. Collapsing it, we get

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

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

1.2.3 The Submodular Welfare Problem

In this section we design a simple 1/2-approximation algorithm for the `Submodular Welfare` problem. The algorithm is very natural. It begins with the empty assignment. Then, in each iteration it assigns a yet unassigned item to the buyer for whom the item has the largest marginal contribution. A formal statement of this algorithm is given as Algorithm 6.

Theorem 1.12 (Due to [8])**.** *Algorithm 6 is a* 1/2*-approximation algorithm for the* `Submodular Welfare` *problem.*

Algorithm 6: The Greedy Algorithm for Submodular Welfare($\{f_i\}_{i=1}^k$)

$A_i \leftarrow \varnothing$, for each $i = 1, \ldots, k$.
for $i = 1$ **to** n **do**
  Let j be the player maximizing $f_j(u_i \mid A_j)$.
  Let $A_j \leftarrow A_j + u_i$
return $\{A_j\}_{j=1}^k$

Proof. Let A_j^i be the set of items allocated by the algorithm to buyer j out of items $u_1, \ldots, u_i$. Let OPT_j^i be the set of items allocated by the optimal solution to buyer j out of items $u_{i+1}, u_{i+2}, \ldots, u_n$. For every $1 \leq i \leq k$ we claim that the following inequality holds.

$$\sum_{j=1}^{k} \left[f_j(A_j^i) - f_j(A_j^{i-1}) \right] \geq \sum_{j=1}^{k} \left[f_j(A_j^{i-1} \cup OPT_j^{i-1}) - f_j(A_j^i \cup OPT_j^i) \right] \ . \tag{1.15}$$

Let us explain why this is the case. Assume that the algorithm allocated item i to buyer j, while OPT allocated item i to buyer j^*. Note that this means that on the left hand side of the last inequality only the j-th term may be none zero, while on the right hand side only terms j and j^* may be non-zero. Moreover, the monotonicity of f_j guarantees that the j-th term on the right hand side is non-positive. Thus, it is enough to prove the following.

$$\begin{aligned} f_j(A_j^i) - f_j(A_j^{i-1}) &= f_j(u_i \mid A_j^{i-1}) \geq f_{j^*}(u_i \mid A_{j^*}^{i-1}) \\ &\geq f_{j^*}(u_i \mid A_{j^*}^{i-1} \cup OPT_{j^*}^i) \geq f_{j^*}(A_{j^*}^{i-1} \cup OPT_{j^*}^{i-1}) - f_{j^*}(A_{j^*}^i \cup OPT_{j^*}^i), \end{aligned}$$

where the first inequality follows since j maximizes $f_j(u_i \mid A_j)$, and the second inequality follows by submodularity. Finally the last inequality holds as an equality when $j \neq j^*$ because in this case $A_{j^*}^{i-1} = A_{j^*}^i$ and $OPT_{j^*}^{i-1} = OPT_{j^*}^i + u_i$. When $j = j^*$ the last inequality still holds because its left hand side is non-negative by the monotonicity, and its right hand side is 0.

Summing up Inequality (1.15) for $1 \leq i \leq n$, and using the non-negativity of each f_j, we get:

$$\begin{aligned} \sum_{j=1}^{k} f_j(A_j) &\geq \sum_{j=1}^{k} [f_j(A_j^n) - f_j(A_j^0)] \\ &\geq \sum_{j=1}^{k} f_j(OPT_j^0 \cup A_j^0) - \sum_{j=1}^{k} f_j(OPT_j^n \cup A_j^n) = \sum_{j=1}^{k} f_j(OPT_j) - \sum_{j=1}^{k} f_j(A_j) \ , \end{aligned}$$

which concludes the proof. □

The following observation shows that the above analysis of Algorithm 6 cannot be improved.

Observation 1.13. *The approximation ratio of Algorithm 6 is exactly* 1/2 *even when there are only two buyers.*

Proof. Consider a ground set with only two elements, $\mathcal{N} = \{u_1, u_2\}$. The utility of buyer 1 is equal to 1 if he is assigned at least one of the elements, and 0 if he is assigned no elements. The utility of buyer 2 is 1 if he is assigned u_1, and 0 otherwise. Formally, the utility functions of both players are:

$$f_1(A) = \min\{|A|, 1\} \quad \forall\ A \subseteq \mathcal{N} \qquad \text{and} \qquad f_2(A) = |A \cap \{u_1\}| \quad \forall\ A \subseteq \mathcal{N} \ .$$

One can observe that, given the empty assignment, the marginal contribution of adding u_1 to buyer 1 is 1, and no other assignment of a single element to any user has a larger marginal contribution. Hence, Algorithm 6 might assign u_1 to buyer 1 in its first iteration;[3] in which case, regardless of the assignment of u_2, buyer 1 ends up with a utility of 1 and buyer 2 ends up with no utility, leading to a total utility of 1. On the other hand, the optimal assignment assigns u_1 to buyer 2 and u_2 to buyer 1, which leads to a total utility of 2. □

1.2.4 Notes and Open Questions

Cardinality constraint: The greedy algorithm (Algorithm 1) and its analysis are due to [1]. Random Greedy (Algorithm 2) and its analysis are due to [3]. Finally, the faster Sample Greedy (Algorithm 3) is due to [5] and, independently, [4].

Unlike in the case of the greedy algorithm, the approximation ratio of Random Greedy is not optimal. A much more involved algorithm by [3] achieves an improved approximation ratio of $1/e + 0.004$ for the same problem. On the other hand, a hardness result by [9] shows that no algorithm can achieve an approximation ratio better than 0.491 for the problem using only a polynomial number of value oracle queries. Arguably, one of the the most important open questions related to the results discussed in this chapter is closing this gap. The unnatural state of the art approximation ratio of $1/e + 0.004$ due to [3] can probably be improved. On the other hand, it is unclear whether the corresponding inapproximability result of 0.491 of [9] is tight or not.

An additional interesting question is whether one can get deterministic algorithms achieving the same bounds. Buchbinder and Feldman [10] designed a deterministic variant of Random Greedy achieving the same approximation ratio of $1/e$. A different technique was used by [11] to get a *deterministic* $(1 - 1/e - \varepsilon)$-approximation algorithm for maximizing a non-negative monotone submodular function subject to a cardinality constraint which uses only $O((n/\varepsilon)\log(n/\varepsilon))$ value oracle queries. Later, Buchbinder et al. [4] developed two incomparable randomized $(1/e - \varepsilon)$-approximation algorithms for objective functions that are not necessarily monotone. One algorithm is based on the technique of [11] and uses $O(k\sqrt{(n/\varepsilon)\ln(k/\varepsilon)} + (n/\varepsilon)\ln(k/\varepsilon))$ value oracle queries, while the other is a slight variation of Sample Greedy requiring $O(\frac{n}{\varepsilon^2}\ln(1/\varepsilon))$ value oracle queries.

The cardinality constraint is also related to the *strict cardinality* constraint which allows only sets of size *exactly* k. For monotone objective functions the two constraints are equivalent since one can increase to k the size of any set whose size is smaller than k by adding to it arbitrary elements, and this never decreases the value of the set. For non-monotone objectives, however, the strict cardinality constraint seems to be more difficult. The 0.491 inapproximability result of [9] applies also to maximizing a non-negative submodular function subject to a strict cardinality constraint, but the best approximation ratio known for this problem is only $1/e - o(1)$ [12].

Unconstrained: The random sample algorithm (Algorithm 4) is due to [6]. The double greedy algorithm (Algorithm 5) is due to [7]. It is known that no polynomial time algorithm for the unconstrained submodular maximization can have an approximation ratio of $1/2+\varepsilon$ for any constant $\varepsilon > 0$ [6], and a deterministic algorithm achieving the optimal approximation ratio of $1/2$ for the problem was presented in [10].

Algorithm 4 is non-adaptive in the sense that the list of the sets on which it queries the value oracle is chosen before the first query is made. Feige et al. [6] showed that the approximation ratio of $1/4$ achieved by this algorithm is optimal for non-adaptive algorithms that are also required to

[3]The exact behavior of the algorithm in this case depends on the tie breaking rules used to implement it. If one wants to avoid this, f_1 should be replaced with $f_1(A) = \min\{|A|, 1\} + \varepsilon \cdot |A \cap \{u_1\}|$ for an arbitrary small constant $\varepsilon > 0$.

return one of the sets whose value they queried. If the algorithm is allowed to return a different set, then an improved $1/3$-approximation non-adaptive algorithm exists [6].

`Submodular Welfare` The greedy algorithm presented for `Submodular Welfare` (Algorithm 6) is due to [8]. In fact, [8] designed a greedy algorithm for the much more general problem of maximizing a monotone submodular function with matroid constraints. The best inapproximability result for `SW` shows that no polynomial time algorithm can achieve an approximation ratio of $1-(1-1/k)^k+\varepsilon$ for any constant $\varepsilon > 0$ [13], and thus, for general k, no polynomial time algorithm can achieve an approximation ratio of $1 - 1/e + \varepsilon$. The large gap between this inapproximability result and the approximation ratio of the greedy algorithm motivated Călinescu et al. [14] to develop the Continuous Greedy algorithm, which yields $(1-1/e)$-approximation for `SW`. We discuss this in more details in Sections 1.3.2 and 1.3.3.

1.3 Continuous Methods

Many results for submodular maximization problems are based on a continuous relaxation of these problems knowns as the *multilinear* relaxation. This section surveys a few of the most important results based on this relaxation, and is organized as follows. Sections 1.3.1 gives definitions and preliminary results necessary for the other sections. Sections 1.3.2 and 1.3.3 describe algorithms for obtaining approximately optimal fractional solutions for the multilinear relaxation. Rounding of these fractional solutions is discussed both in Sections 1.3.2 and 1.3.4. Finally, Section 1.3.5 describes a method for obtaining inapproximability results for submodular maximization problems which is based on similar techniques.

1.3.1 Definitions and Preliminaries

The sets of $2^{\mathcal{N}}$ (or $\{0,1\}^{\mathcal{N}}$) can be naturally identified with points of the $[0,1]^{\mathcal{N}}$ cube by identifying each set $A \subseteq \mathcal{N}$ with its characteristic vector (which we denote here by $\mathbf{1}_A$). For linear functions this gives a natural way to extend any function from $\{0,1\}^{\mathcal{N}}$ to the cube $[0,1]^{\mathcal{N}}$. However, finding the right extension for submodular functions is more difficult. One useful extension of submodular functions is the Lovász extension defined as follows.

Definition 1.4. *(The Lovász extension) Given a vector $x \in [0,1]^{\mathcal{N}}$ and a scalar $\lambda \in [0,1]$, let $T_\lambda(x) = \{u \in \mathcal{N} \mid x_u \geq \lambda\}$ be the set of elements in $\mathcal{N}$ whose coordinate in x is at least λ. Then,*

$$\hat{f}(x) = \int_{\lambda=0}^{1} f(T_\lambda(x))d\lambda \ .$$

Definition 1.4 can also be interpreted in probabilistic terms as the expected value of f over the set $T_\lambda(x)$, where λ is selected uniformly at random from the range $[0,1]$.

The following lemma proves an important property of the Lovász extension. One way to prove this lemma is via the equality, which follows from the work of [15], between the Lovász extension of a submodular function and another extension known as *the convex closure*. However, the proof we give here is based on a proof originally given by [3].

Lemma 1.14. *Let $f\colon 2^{\mathcal{N}} \to \mathbb{R}$ be a submodular function, and let $\hat{f}$ be its Lovász extension. For every $x \in [0,1]^{\mathcal{N}}$, let D_x denote an arbitrary distribution over $\{0,1\}^{\mathcal{N}}$ such that $\Pr[u \in D_x] = x_u$ for every $u \in \mathcal{N}$ (i.e., its marginals agree with x). Then, $\hat{f}(x) \leq \mathbb{E}_{A\sim D_x}[f(A)]$.*

Proof. Let us denote by x_i the coordinate of x corresponding to element u_i. Assume without loss of generality that $x_1 \geq x_2 \geq \ldots \geq x_n$ is a non-increasing series (this can always be guaranteed by choosing the right order for the elements), and let $B_i = \{u_1, u_2, \ldots, u_i\}$. Let A be a random set distributed according to the distribution D_x, and let us denote by Y_i an indicator for the event that $u_i \in A$. Then,

$$\begin{aligned}\mathbb{E}_{A\sim D_x}[f(A)] &= \mathbb{E}\left[f(\varnothing) + \sum_{i=1}^{n} Y_i \cdot f(u_i \mid A \cap B_{i-1})\right] \geq \mathbb{E}\left[f(\varnothing) + \sum_{i=1}^{n} Y_i \cdot f(u_i \mid B_{i-1})\right] \\ &= f(\varnothing) + \sum_{i=1}^{n} \mathbb{E}[Y_i] \cdot f(u_i \mid B_{i-1}) = f(\varnothing) + \sum_{i=1}^{n} x_i \cdot f(u_i \mid B_{i-1}) \\ &= (1-x_1)\cdot f(\varnothing) + \sum_{i=1}^{n-1}(x_i - x_{i+1}) \cdot f(B_i) + x_n \cdot f(\mathcal{N}) = \hat{f}(A) \ ,\end{aligned}$$

where the inequality follows from submodularity and the last equality is true since our assumption that $x_1 \geq x_2 \geq \cdots \geq x_n$ implies that $T_\lambda(x) = B_i$ for every $\lambda \in (x_{i+1}, x_i]$ (except for the cases $i = 0$ and $i = n$ for which the correct ranges are $(x_1, 1]$ and $[0, x_n]$, respectively). □

Remark: Recall that $\hat{f}(x)$ was defined as the expected value of f over the distribution of the set $T_\lambda(x)$ when λ is chosen uniformly at random from the range $[0, 1]$. One can observe that the marginals of this distribution agree with x. Hence, Lemma 1.14 can be interpreted as claiming that, among all the distributions D_x whose marginals agree with x, the distribution defining $\hat{f}(x)$ is the one with the minimum expected value.

Another important extension of submodular functions is the multilinear extension.

Definition 1.5. *(The multilinear extension) Given a vector $x \in [0,1]^{\mathcal{N}}$, let $\mathtt{R}(x)$ denote a random subset of $\mathcal{N}$ containing every element $u \in \mathcal{N}$* **independently** *with probability x_u. Then,*

$$F(x) = \mathbb{E}[\mathtt{R}(x)] = \sum_{A \subseteq \mathcal{N}} \left(f(A) \cdot \Pr[\mathtt{R}(x) = A]\right) = \sum_{A \subseteq \mathcal{N}} \left(f(A) \cdot \prod_{u \in A} x_u \cdot \prod_{u \notin A} (1 - x_u)\right) \ .$$

The explicit formula for the multilinear extension makes it clear that, as suggested by its name, the multilinear extension is a multilinear function[4]. The multilinear extension is central to all the results presented in the remaining sections of this chapter. By definition $F(x)$ is the expected value of f over the distribution of the random set $\mathtt{R}(x)$. Since the marginals of this distribution agree with x, we get by Lemma 1.14 the following immediate corollary.

Corollary 1.15. *Let $f\colon 2^{\mathcal{N}} \to \mathbb{R}$ be a submodular function, and let $\hat{f}$ and F be its Lovász and multilinear extensions, respectively. Then, for every $x \in [0,1]^{\mathcal{N}}$, $F(x) \geq \hat{f}(x)$.*

We conclude this section with an additional notation which is used heavily in the next sections. For two vectors $x, y \in [0,1]^{\mathcal{N}}$, we use $x \vee y$ to denote the coordinate-wise maximum of x and y (formally, $(x \vee y)_u = \max\{x_u, y_u\}$ for every $u \in \mathcal{N}$).

1.3.2 Continuous Greedy

Recall that the problems we are interested in fall into the framework of maximizing a given non-negative submodular function $f\colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ subject to a (possibly trivial) constraint. Denoting by

[4] A multilinear function is a polynomial function in which the degree of every variable is at most 1.

$\mathcal{I}$ the collection of subsets of $\mathcal{N}$ obeying the constraint, this class of problems can be formulated as follows.

$$\begin{array}{ll} \max & f(A) \\ \text{s.t.} & A \in \mathcal{I} \subseteq \{0,1\}^{\mathcal{N}} \end{array} \tag{1.16}$$

An important relaxation of the general problem (1.16), called the multilinear relaxation, is obtained by replacing f with its multilinear extension F and $\mathcal{I}$ with a convex body P which is contained in the cube $[0,1]^{\mathcal{N}}$ on the one hand, and contains the characteristic vectors of the sets in $\mathcal{I}$ on the other hand. Formally, the relaxation we get is

$$\begin{array}{ll} \max & F(x) \\ \text{s.t.} & x \in P \subseteq [0,1]^{\mathcal{N}} \end{array} \tag{1.17}$$

Remark: Usually the natural relaxation P of the constraint is a polytope. However, this is not necessary for the results we present, and thus, we allow P to be an arbitrary convex body.

Given a relaxation in the form of (1.17), one can attempt to approximate the original (integral) problem by first finding an approximate solution for the relaxation, and then rounding this solution without loosing too much in the objective. Note that this approach is similar to the standard approach for linear optimization problems in which the algorithm first solves a linear programming (LP) relaxation of the problem, and then rounds the fractional solution obtained.

The Continuous Greedy algorithm is analogous to the LP solver in the above approach. In other words, given a relaxation in the form of (1.17), Continuous Greedy finds a fractional solution whose value approximates the value of the optimal integral solution. This fractional solution can then be rounded by various methods discussed later in this section and in Section 1.3.4. It is interesting to note that unlike LP solvers, the output of Continuous Greedy only approximates the value of the optimal integral solution. This is unavoidable, as is shown later in this section. Continuous Greedy assumes that P is *solvable* meaning that one can optimize linear functions over it.[5]

We are now ready to describe Continuous Greedy. Continuous Greedy maintains a solution that evolves during the time interval $[0,1]$. The solution starts at time 0 as the empty solution, *i.e.*, $\mathbf{1}_{\varnothing}$. At every time point $t \in [0,1)$ the algorithm calculates a weight vector $w(t) \in \mathbb{R}^n$ whose value for each element $u \in N$ is $w_u(t) = F(x \vee \mathbf{1}_{\{u\}}) - F(x)$. In other words, $w_u(t)$ is the gain that the algorithm can get by increasing the coordinate of u in its solution to be 1. The algorithm then finds a vector $x(t)$ in P maximizing the linear objective function $w(t) \cdot x(t)$ defined by these weights. Intuitively, $x(t)$ tells Continuous Greedy which coordinates should be increased at the current time. The algorithm uses this information by adding an infinitesimal fraction of $x(t)$ to its current solution. A formal statement of Continuous Greedy is given as Algorithm 7.

Algorithm 7: Continuous Greedy(f, P)

```
Let y(0) ← 1_∅.
foreach t ∈ [0,1) do
    For each u ∈ N, let w_u(t) ← F(y(t) ∨ 1_{u}) − F(y(t)).
    x(t) ← arg max_{x∈P}{w(t) · x}.
    Increase y(t) at a rate of dy(t)/dt = x(t).
return y(1)
```

[5]Continuous Greedy can also be used when one can only approximately maximize linear functions over P, but this translates into a poorer guarantee on its output.

Naturally, the above continuous formulation of Continuous Greedy cannot be implemented in polynomial time. However, there is a discretized version of Continuous Greedy that has the same guarantee (up to low order terms that can be removed by an appropriate preprocessing step) and can be implemented efficiently. The discretization involves two main steps. First, instead of having a continuous time, the algorithm advances time in small steps. Second, instead of calculating $w_u(t)$ by evaluating the multilinear extension F exactly (which might require exponential time), the algorithm approximates $w_u(t) = F(y(t) \vee \mathbf{1}_{\{u\}}) - F(y(t)) = \mathbb{E}[f(\mathtt{R}(y(t) \vee \mathbf{1}_{\{u\}})) - f(\mathtt{R}(y(t)))]$ by averaging a polynomial number of samples from the distribution of the random quantity $f(\mathtt{R}(y(t) \vee \mathbf{1}_{\{u\}})) - f(\mathtt{R}(y(t)))$.

The continuous version of Continuous Greedy is cleaner, and thus, we give here only its analysis and leave out the technical details of how to apply this analysis to the discretized version of the algorithm. Unfortunately, the analysis of the continuous version is not completely formal since it involves integration of various functions, and we implicitly assume that these functions are integrable. The reader should keep in mind, however, that this issue goes away when the analysis is applied to the discretized version since the above integrations are then replaced by finite sums. Călinescu et al. [14] proved the following guarantee for Continuous Greedy.

Theorem 1.16 (Due to [14])**.** *Let f be a non-negative monotone submodular function, and let P is a solvable convex body. Then, Continuous Greedy outputs a vector $y(1) \in P$ such that $F(y(1)) \geq (1 - 1/e) \cdot f(OPT)$, where OPT is a solution in $\mathcal{I}$ maximizing $f(OPT)$.*

Proof. Recall that $x(t)$ is a vector inside P for every time $t \in [0, 1)$. Hence, $y(1) = \int_{t=0}^{1} x(t)dt$ is a convex combination of vectors in P, and thus, belongs to P by the convexity of P. Additionally, by the chain rule,

$$\frac{dF(y(t))}{dt} = \sum_{u \in \mathcal{N}} \left(\frac{dy_u(t)}{dt} \cdot \left. \frac{\partial F(y)}{\partial y_u} \right|_{y=y(t)} \right) = \sum_{u \in \mathcal{N}} \left(x_u(t) \cdot \left. \frac{\partial F(y)}{\partial y_u} \right|_{y=y(t)} \right) \ .$$

Recall, now, that F is multilinear. Thus, its partial derivative with respect to a single coordinate is equal to the difference between the value of the function for two different values of this coordiante over the difference between the values. Plugging this observation into the previous inequality yields (for $t < 1$)

$$\begin{aligned} \frac{dF(y(t))}{dt} &= \sum_{u \in \mathcal{N}} \left(x_u(t) \cdot \left. \frac{\partial F(y)}{\partial y_u} \right|_{y=y(t)} \right) \\ &= \sum_{u \in \mathcal{N}} \left(x_u(t) \cdot \frac{F(y(t) \vee \mathbf{1}_{\{u\}}) - F(y(t))}{1 - y_u(t)} \right) \geq \sum_{u \in \mathcal{N}} x_u(t) \cdot w_u(t) = x(t) \cdot w(t) \ . \end{aligned}$$

Next, note that one possible candidate to be $x(t)$ is $\mathbf{1}_{OPT}$ since P contains the characteristic vectors of all the feasible sets. Hence, $x(t) \cdot w(t) \geq \mathbf{1}_{OPT} \cdot w(t)$. Plugging this into the previous inequality gives

$$\begin{aligned} \frac{dF(y(t))}{dt} &\geq x(t) \cdot w(t) \geq \mathbf{1}_{OPT} \cdot w(t) = \sum_{u \in OPT} [F(y(t) \vee \mathbf{1}_{\{u\}}) - F(y(t))] \qquad (1.18) \\ &= \sum_{u \in OPT} \mathbb{E}[f(\mathtt{R}(y(t) \vee \mathbf{1}_{\{u\}})) - f(\mathtt{R}(y(t)))] \geq \mathbb{E}[f(\mathtt{R}(y(t) \vee \mathbf{1}_{OPT})) - f(\mathtt{R}(y(t)))] \\ &= F(y(t) \vee \mathbf{1}_{OPT}) - F(y(t)) \geq f(OPT) - F(y(t)) \ , \end{aligned}$$

where the penultimate inequality holds by the submodularity of f and the last inequality holds by its monotonicity.

Inequality 1.18 is a differential inequality with respect to the function $F(y(t))$. Solving this inequality for the initial condition $F(y(0)) \geq 0$ gives

$$F(y(t)) \geq (1 - e^{-t}) \cdot f(OPT) \ .$$

The theorem now follows by plugging $t = 1$ into the last inequality. □

Rounding the output of Continuous greedy: We consider next the problem of rounding the output of Continuous Greedy. In particular, as an illustrative example, let us consider rounding for Submodular Welfare (SW). Recall that we have already shown (in Section 1.1.2) an embedding for SW into the general problem (1.16). The natural multilinear relaxation for the embedded form of SW is as follows.

$$\begin{array}{lll} \max & F(x) = \mathbb{E}[\sum_{i=1}^{k} f_i(\{u \in \mathcal{N}^{\mathtt{SW}} : (u,i) \in \mathtt{R}(x)\})] & \\ \text{s.t.} & \sum_{i=1}^{k} x_{(u,i)} \leq 1 & \forall\ u \in \mathcal{N}^{\mathtt{SW}} \\ & x_{(u,i)} \geq 0 & \forall\ u \in \mathcal{N}^{\mathtt{SW}}, 1 \leq i \leq k \end{array}$$

Notice that every fractional solution for this relaxation splits each item fractionally among the k buyers, so that the total fractions of all the buyers add up to at most 1. This means that one can round such a solution x by assigning each item $u \in \mathcal{N}^{\mathtt{SW}}$, independently, to at most one user where the probability that u is assigned to user $1 \leq i \leq k$ is exactly $x_{(u,i)}$. Let us denote the random output of this rounding process by $\mathtt{R}_d(x)$ (the subscript d standards for "dependent" because, unlike in $\mathtt{R}(x)$, here there is a dependency between the rounded values of all the coordinates corresponding to the same item u). The following lemma shows that this rounding does not loose in expectation.

Lemma 1.17. *For every vector $x \in [0,1]^{\mathcal{N}}$ obeying $\sum_{i=1}^{k} x_{(u,i)} \leq 1$ for every item $u \in \mathcal{N}^{\mathtt{SW}}$, $F(x) = \mathbb{E}[\sum_{i=1}^{k} f_i(\{u \in \mathcal{N}^{\mathtt{SW}} : (u,i) \in \mathtt{R}_d(x)\})]$.*

Proof. For every given buyer $1 \leq i \leq k$, every item $u \in \mathcal{N}^{\mathtt{SW}}$ is assigned to i with probability $x_{(u,i)}$ both by $\mathtt{R}_d(x)$ and by $\mathtt{R}(x)$. Moreover, in both cases this assignment is independent of the assignment of the other items of $\mathcal{N}^{\mathtt{SW}}$. Hence, for every buyer i, $\{u \in \mathcal{N}^{\mathtt{SW}} : (u,i) \in \mathtt{R}_d(x)\}$ has the same distribution as $\{u \in \mathcal{N}^{\mathtt{SW}} : (u,i) \in \mathtt{R}(x)\}$. The observation now follows by the linearity of the expectation since:

$$\begin{aligned} F(x) &= \mathbb{E}\left[\sum_{i=1}^{k} f_i(\{u \in \mathcal{N}^{\mathtt{SW}} : (u,i) \in \mathtt{R}(x)\})\right] = \sum_{i=1}^{k} \mathbb{E}[f_i(\{u \in \mathcal{N}^{\mathtt{SW}} : (u,i) \in \mathtt{R}(x)\})] \\ &= \sum_{i=1}^{k} \mathbb{E}[f_i(\{u \in \mathcal{N}^{\mathtt{SW}} : (u,i) \in \mathtt{R}_d(x)\})] = \mathbb{E}\left[\sum_{i=1}^{k} f_i(\{u \in \mathcal{N}^{\mathtt{SW}} : (u,i) \in \mathtt{R}_d(x)\})\right] \ . \end{aligned}$$

□

In conclusion, one can get a $(1 - 1/e)$-approximation algorithm for SW, which improves on the greedy approach described in Section 1.2.3, using the following two steps. First, use Continuous Greedy to get a fractional solution y for the multilinear relaxation of SW such that $F(y)$ is at least $(1 - 1/e)$ times the value of the best assignment. Then, round y randomly and output $\mathtt{R}_d(y)$. By the last lemma, the expected value of this random assignment is equal to $F(y)$; and thus, the approximation ratio of this algorithm is indeed $(1 - 1/e)$.

Remark: The approximation ratio of the above algorithm matches the inapproximability result of [13] for SW with general k values. Thus, the guarantee of Continuous Greedy cannot be improved in general since any such improvement (except for a low order terms improvement) will violate the result of [13].

1.3.3 Measured Continuous greedy

This section presents a variant of continuous greedy, called Measured Continuous Greedy and originally suggested by [16], that has several advantages over the original algorithm. The first advantage is the ability of Measured Continuous Greedy to achieve an improved approximation in some cases (although, as discussed above, the approximation ratio of Continuous Greedy cannot be improved for the general case). Intuitively, the improved approximation of Measured Continuous Greedy is based on the following observation. The analysis of Continuous Greedy shows that the solution that it maintains at time t has value of at least $(1-e^{-t})\cdot f(OPT)$. This guarantee improves as t grows, and thus, it is best to let the algorithm run for as long as possible. Unfortunately, Continuous Greedy cannot run after time $t = 1$ since this might result in an infeasible solution. Measured Continuous Greedy tries to avoid this problem by increasing the coordinates of its solution slower, which can sometimes allow it to reach larger values of t. Algorithm 8 is a formal statement of Measured Continuous Greedy. The parameter T is the amount of time we let the algorithm run.

Algorithm 8: Measured Continuous Greedy(f, P, T)

Let $y(0) \leftarrow \mathbf{1}_{\varnothing}$.
foreach $t \in [0, T)$ **do**
  For each $u \in \mathcal{N}$, let $w_u(t) \leftarrow F(y(t) \vee \mathbf{1}_{\{u\}}) - F(y(t))$.
  $x(t) \leftarrow \arg\max_{x \in P}\{w(t) \cdot x\}$.
  Increase $y(t)$ at a rate of $\frac{dy(t)}{dt} = (1 - y(t)) \cdot x(t)$.
return $y(T)$

It is not obvious from the description of Measured Continuous Greedy that $y(t)$ is always within $[0, 1]^{\mathcal{N}}$. The following lemma proves a stronger claim which we use later on.

Lemma 1.18. *For every time $t \in [0, T]$ and element $u \in \mathcal{N}$, $y_u(t) \leq 1 - e^{-t} < 1$.*

Proof. Since $x(t)$ is always in $P \subseteq [0, 1]^{\mathcal{N}}$, $y_u(t)$ obeys the differential inequality

$$\frac{dy_u(t)}{dt} = (1 - y_u(t)) \cdot x_u(t) \leq 1 - y_u(t) \ .$$

Using the initial condition $y_u(0) = 0$, the solution for this differential inequality is $y_u(t) \leq 1 - e^{-t}$. □

Let us now analyze the approximation guarantee of Measured Continuous Greedy.

Lemma 1.19. *Let f be a non-negative monotone submodular function, and let P be a solvable convex body. Measured Continuous Greedy outputs a vector $y(T)$ such that $F(y(T)) \geq (1 - e^{-T}) \cdot f(OPT)$, where OPT is a solution in $\mathcal{I}$ maximizing $f(OPT)$.*

Proof. By the chain rule,

$$\frac{dF(y(t))}{dt} = \sum_{u \in \mathcal{N}} \left(\frac{dy_u(t)}{dt} \cdot \frac{\partial F(y)}{\partial y_u}\Big|_{y=y(t)} \right) = \sum_{u \in \mathcal{N}} \left((1 - y_u(t)) \cdot x_u(t) \cdot \frac{\partial F(y)}{\partial y_u}\Big|_{y=y(t)} \right) \ .$$

As F is multilinear, its partial derivative with respect to a single coordinate is the difference between the value of the function for two different values of this coordinate over the difference

between the values. Plugging this observation into the previous inequality yields

$$\begin{aligned}\frac{dF(y(t))}{dt} &= \sum_{u\in\mathcal{N}}\left((1-y_u(t))\cdot x_u(t)\cdot \left.\frac{\partial F(y)}{\partial y_u}\right|_{y=y(t)}\right)\\ &= \sum_{u\in\mathcal{N}}\left((1-y_u(t))\cdot x_u(t)\cdot \frac{F(y(t)\vee \mathbf{1}_{\{u\}})-F(y(t))}{1-y_u(t)}\right)\\ &= \sum_{u\in\mathcal{N}} x_u(t)\cdot w_u(t) = x(t)\cdot w(t)\ .\end{aligned}$$

The rest of the proof is completely analogous to the corresponding part in the proof of Theorem 1.16, and is, thus, omitted. □

The values of T for which the output of Measured Continuous Greedy is feasible depend on the properties of the convex body P. In fact the output of Measured Continuous Greedy might not be feasible even for $T = 1$. We demonstrate this issue by studying the values of T for which Measured Continuous Greedy outputs a feasible solution when given the multilinear relaxation of SW. As a consequence, we get an improved approximation algorithm for SW.

Lemma 1.20. *Given the multilinear relaxation of SW and $T = -k\ln(1-1/k)$, Measured Continuous Greedy outputs a feasible solution of value at least $[1-(1-1/k)^k]\cdot f(OPT)$.*

Proof. Fix an arbitrary item $u \in \mathcal{N}^{\mathtt{SW}}$. We need to prove that $\sum_{i=1}^{k} y_{(u,i)}(T) \le 1$. For every $1 \le i \le k$ and time $t \in [0,T)$ we have

$$\frac{dy_{(u,i)}(t)}{dt} = (1-y_{(u,i)}(t))\cdot x_{(u,i)}(t)\ .$$

Solving this differential equation for $y_{(u,i)}(t)$ and plugging the initial condition $y_{(u,i)}(0) = 0$ yield

$$y_{(u,i)}(t) = 1 - e^{-\int_{\tau=0}^{t} x_{(u,i)}(\tau)d\tau}\ . \tag{1.19}$$

Notice that the sum $\sum_{i=1}^{k}\int_0^T x_{(u,i)}(\tau)d\tau$ is at most T since $\sum_{i=1}^{k} x_{(u,i)}(t) \le 1$ for every time $t \in [0,T)$. Additionally, the function $1-e^{-x}$ is concave. Combining these observations with Equation (1.19) gives

$$\begin{aligned}\sum_{i=1}^{k} y_{(u,i)}(T) &= \sum_{i=1}^{k}[1-e^{-\int_{\tau=0}^{T} x_{(u,i)}(\tau)d\tau}] \le k[1-e^{-k^{-1}\cdot\sum_{i=1}^{k}\int_{\tau=0}^{T} x_{(u,i)}(\tau)d\tau}]\\ &\le k[1-e^{-k^{-1}\cdot T}] = k[1-e^{\ln(1-1/k)}] = 1\ ,\end{aligned}$$

where the penultimate equality follows by plugging in the value of T. It remains to analyze the quality of the solution $y(T)$ outputted by Measured Continuous Greedy. By Lemma 1.19,

$$F(y(T)) \ge (1-e^{-T})\cdot f(OPT) = [1-e^{k\ln(1-1/k)}]\cdot f(OPT) = [1-(1-1/k)^k]\cdot f(OPT)\ .\ \square$$

Lemma 1.17 shows that the fractional solution obtained by Lemma 1.20 can be efficiently rounded without any loss in the objective, yielding a $(1-(1-1/k)^k)$-approximation algorithm for SW. The approximation ratio of this algorithm matches (up to low order terms) the inapproximability result of [13] for any given value of k. For general values of k this result does not improve over the $(1-1/e)$-approximation algorithm presented in Section 1.3.2, however, for small values

of k the improvement is significant. For example, for $k = 2$ the improved approximation ratio is $1 - (1 - 1/2)^2 = 0.75$, whereas the algorithm presented previously achieves, even for this case, an approximation ratio of only $(1 - 1/e) \cong 0.632$.

An additional advantage of Measured Continuous Greedy is its ability to handle well non-monotone functions. A convex body $P \subseteq [0,1]^{\mathcal{N}}$ is called *down-closed* if decreasing the coordinates of a vector inside it (while keeping them non-negative) can never take the vector outside of P. Formally, P is down-closed if, for every two vectors $x, y \in [0,1]^{\mathcal{N}}$, $x \leq y$ and $y \in P$ imply $x \in P$. We prove the following result for down-closed convex bodies.

Theorem 1.21 (Due to [16])**.** *Let f be a non-negative submodular function, and let $\mathcal{P}$ be a solvable down-closed convex body. Then, for $T = 1$ Measured Continuous Greedy outputs a vector $y(1) \in \mathcal{P}$ such that $F(y(1)) \geq 1/e \cdot f(OPT)$, where OPT is a solution in $\mathcal{I}$ maximizing $f(OPT)$.*

Proof. Recall that $x(t)$ is a vector inside P for every time $t \in [0,1]$, and thus, since P is down-closed, $(1-y(t)) \cdot x(t)$ also belongs to P. Hence, $y(1) = \int_{t=0}^{1} (1-y(t) \cdot x(t))dt$ is a convex combination of vectors in P, and therefore, belongs to P by the convexity of P.

Let us now analyze the value of $F(y(1))$. Repeating the proof of Lemma 1.19 we get

$$\frac{dF(y(t))}{dt} = x(t) \cdot w(t) \ .$$

As before, one possible candidate to be $x(t)$ is $\mathbf{1}_{OPT}$ since P contains the characteristic vectors of all the feasible sets. Hence, $x(t) \cdot w(t) \geq \mathbf{1}_{OPT} \cdot w(t)$. Plugging this into the previous inequality gives

$$\begin{aligned}\frac{dF(y(t))}{dt} = x(t) \cdot w(t) \geq \mathbf{1}_{OPT} \cdot w(t) &= \sum_{u \in OPT} [F(y(t) \vee \mathbf{1}_{\{u\}}) - F(y(t))] \\ &\geq F(y(t) \vee \mathbf{1}_{OPT}) - F(y(t)) \geq \hat{f}(y(t) \vee \mathbf{1}_{OPT}) - F(y(t)) \ ,\end{aligned}$$

where $\hat{f}$ is the Lovász extension of f, the penultimate inequality holds by the submodularity of f and the last inequality holds by Corollary 1.15. To lower bound the term $\hat{f}(y(t) \vee \mathbf{1}_{OPT})$ we recall that by Lemma 1.18 every coordinate of $y(t)$ is upper bounded by $1 - e^{-t}$. Hence,

$$\hat{f}(y(t) \vee \mathbf{1}_{OPT}) = \int_{\lambda=0}^{1} f(T_\lambda(y(t) \vee \mathbf{1}_{OPT}))d\lambda \geq \int_{\lambda=1-e^{-t}}^{1} f(T_\lambda(y(t) \vee \mathbf{1}_{OPT}))d\lambda = e^{-t} \cdot f(OPT) \ .$$

Plugging the last inequality into the previous one gives

$$\frac{dF(y(t))}{dt} \geq e^{-t} \cdot f(OPT) - F(y(t)) \ ,$$

which is a differential inequality with respect to the function $F(y(t))$. Solving this inequality for the initial condition $F(y(0)) \geq 0$ gives

$$F(y(t)) \geq te^{-t} \cdot f(OPT) \ .$$

Finally, the theorem follows by plugging $t = T = 1$ into the last inequality. □

Theorem 1.21 sets T to 1. Like in the monotone case, it is often possible to get a feasible output also for other values of T. However, this does not seem to be beneficial for non-monotone functions as the formula te^{-t} used by the proof of Theorem 1.21 to determine the approximation ratio of Measured Continuous Greedy attains its maximal value at $t = 1$.

Implementation of Measured Continuous Greedy requires discretization, which reduces the approximation ratio by a low order term. Like in the case of Continuous Greedy, this loss in the approximation ratio can be fixed by an appropriate preprocessing for monotone objective functions. However, this is not known to be the case for non-monotone functions. Hence, for polynomial time implementations of Measured Continuous Greedy, the guarantee of Theorem 1.21 should be reduced from $1/e \cdot f(OPT)$ to $(1/e - o(1)) \cdot f(OPT)$, where $o(1)$ is a term which diminishes when $n = |\mathcal{N}|$ increases.

1.3.4 Contention Resolution Schemes

Contention resolution schemes, originally proposed by [17], are a framework for rounding fractional solutions under submodular and linear objective functions. As such, contention resolution schemes play important role in many submodular maximization algorithms that are based on the standard algorithmic scheme of first finding an approximately optimal fractional solution and then rounding it. Moreover, contention resolution schemes found uses also outside the world of submodular maximization (see, *e.g.*, [18, 19, 20]).

We begin the presentation of the contention resolution schemes framework with some definitions.

Definition 1.6. *(Contention Resolution Scheme) A* contention resolution scheme *(CRS) for a given down-closed body*[6] $P \subseteq [0,1]^{\mathcal{N}}$ *is a (possibly random) function* $\pi_x : 2^{\mathcal{N}} \to 2^{\mathcal{N}}$*, where* $x \in P$ *and* $A \subseteq \mathcal{N}$ *that satisfies that for every* $x \in P$ *and* $A \subseteq \mathcal{N}$*:*

1. $\pi_x(A) \subseteq A$.
2. *The characteristic vector of* $\pi_x(A)$ *always belongs to* P.

Intuitively, we think of $\pi_x(\mathtt{R}(x))$ as a random rounding for the fractional point $x \in P$. Notice that this is indeed a rounding in the sense that it gets as input a possibly fractional point $x \in P$ and outputs a set corresponding to an integral point in P. Given this point of view, we can think of the parameter x of the CRS as informing the CRS about the distribution of its other parameter, and allowing it to choose a response for every given input set which yields a good (in some sense) rounding overall. To get some guarantee on the quality of the rounding induced by a CRS, we first need to define additional properties of CRSs.

Definition 1.7. *Let* $P \subseteq [0,1]^{\mathcal{N}}$ *be a down-closed body and let* π *be a CRS for* P.

- π *is* c-balanced *(for* $c \in [0,1]$*) if* $\Pr[u \in \pi_x(\mathtt{R}(x))] \geq c \cdot x_u$ *for every element* $u \in \mathcal{N}$ *and vector* $x \in P$.
- π *is* monotone *if* $\Pr[u \in \pi_x(A)] \geq \Pr[u \in \pi_x(B)]$ *whenever* $x \in P$ *and* $A \subseteq B \subseteq \mathcal{N}$.

It is immediate that a c-balanced CRSs can be used for rounding solutions under a linear function objective as follows. Let $f \colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ be a non-negative linear function, and let F be its the natural extension. Then, $\mathbb{E}[f(\pi_x(\mathtt{R}(x)))] \geq c \cdot F(x)$ for every $x \in P$. We are interested in proving a similar claim for a non-negative submodular function f and its multilinear extension F. Unfortunately, requiring the CRS to be c-balanced is not enough even when f is a monotone function. This is demonstrated by the example described in Figure 1.1. Notice that in this example π is $(1/2)$-balanced (at least for the x given in this figure) since $\Pr[u \in \pi_x(\mathtt{R}(x))] = \Pr[v \in \pi_x(\mathtt{R}(x))] = 1/4 = x_u/2 = x_v/2$. However, $\mathbb{E}[f(\pi_x(\mathtt{R}(x)))] = 1/4 \not\geq F(x)/2 = 3/8$.

[6] It is usually assumed that P is a down-closed polytope, however, this assumption is not essential for the results we present in this section.

Instance			CRS	
$\mathcal{N}$	$= \{u, v\}$		$\pi_x(\varnothing)$	$= \varnothing$
$f(A)$	$= \min\{\lvert A\rvert, 1\}$	$\forall\, A \subseteq \mathcal{N}$	$\pi_x(\{u\})$	$= \varnothing$
P	$= [0,1]^{\mathcal{N}}$		$\pi_x(\{v\})$	$= \varnothing$
x	$= (1/2, 1/2)$		$\pi_x(\{u, v\})$	$= \{u, v\}$

Figure 1.1: An example of a c-balanced CRS for which $\mathbb{E}[f(\pi_x(\mathtt{R}(x)))] \geq c \cdot F(x)$ is not true.

The problem, intuitively, is that the CRS π described by Figure 1.1 is an unnatural choice for a rounding scheme. We expect a rounding scheme to be more liberal in removing elements when there are more elements in its input set (and thus, the set is more likely to violate down-closed constraints). However, the behavior of π is quite the opposite. Namely, π keeps elements only when its input set contains both u and v. This intuitive idea is formalized by the monotonicity property in Definition 1.7 (which is violated in this example). Assuming both properties of Definition 1.7 (c-balance and monotonicity) allow us to obtain the desired result for submodular functions.

Theorem 1.22 (Due to [17])**.** *Let $f\colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ be a non-negative monotone submodular function. Let π be a monotone c-balanced CRS for a down-closed body P. Then, $\mathbb{E}[f(\pi_x(\mathtt{R}(x)))] \geq c \cdot F(x)$ for every $x \in P$.*

Proof. Let $A \sim R(x)$ be a random set distributed like $\mathtt{R}(x)$, and for every $1 \leq i \leq n$ let X_i be an indicator for the event that $u_i \in \pi_x(A)$, and let $B_i = \{u_1, \ldots, u_i\}$. Using this notation we get for every $1 \leq i \leq n$:

$$\begin{aligned}
\mathbb{E}_{A\sim R(x)}[f(\pi_x(A) \cap B_i) - f(\pi_x(A) \cap B_{i-1})] &= \mathbb{E}[X_i \cdot f(u_i \mid \pi_x(A) \cap B_{i-1})] \\
&\geq \mathbb{E}[X_i \cdot f(u_i \mid A \cap B_{i-1})] = \mathbb{E}[\Pr[u_i \in \pi_x(A)] \cdot f(u_i \mid A \cap B_{i-1})] \\
&\geq \mathbb{E}[\Pr[u_i \in \pi_x(A)]] \cdot \mathbb{E}[f(u_i \mid A \cap B_{i-1})] \\
&\geq c x_{u_i} \cdot \mathbb{E}[f(u_i \mid A \cap B_{i-1})] \ .
\end{aligned}$$

The first inequality follows by submodularity since $\pi_x(A) \subseteq A$. The second inequality follows by the FKG inequality since $\Pr[u_i \in \pi_x(A)]$ is a non-increasing function of A due to the monotonicity of π and $f(u_i \mid A \cap B_{i-1})$ is also a non-increasing function of A due to the submodularity of f. The final inequality follows since π is c-balanced. Summing up the above inequality for all $1 \leq i \leq n$ we get

$$\begin{aligned}
\mathbb{E}_{A\sim R(x)}[f(\pi_x(A))] = \mathbb{E}[f(\pi_x(A) \cap B_n)] &= f(\varnothing) + \sum_{i=1}^{n} \mathbb{E}_{A\sim R(x)}[f(\pi_x(A) \cap B_i) - f(\pi_x(A) \cap B_{i-1})] \\
&\geq f(\varnothing) + \sum_{i=1}^{n} c x_{u_i} \cdot \mathbb{E}[f(u_i \mid A \cap B_{i-1})] \geq c \cdot F(x) \ ,
\end{aligned}$$

where the last inequality holds since the non-negativity of f guarantees that $f(\varnothing) \geq c \cdot f(\varnothing)$. □

There is an extension of Theorem 1.22 for non-monotone submodular functions. The only difference between this extension and the original theorem is that it requires the output of the CRS to go through an additional post-processing step which might remove from it additional elements.

Theorem 1.23 (Due to [17]). *Let $f\colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ be a non-negative submodular function. Let π be a monotone c-balanced CRS for a down-closed body P. Then, there exists a CRS π' which can be efficiently computed (given that π can) such that $\mathbb{E}[f(\pi'_x(\mathtt{R}(x)))] \geq c \cdot F(x)$ for every $x \in P$.*

We do not prove Theorem 1.23 in this survey. However, we would like to point out that π' is very simple. It first apply π, and then it simply scans the elements left in some predetermined order, and removes every element whose marginal contribution to the subset of the elements scanned so far is negative.

Contention Resolution Scheme for matchings: At this point we would like to give an example of a CRS and analyze its properties. The down-closed body of our example CRS is the natural relaxation of matching. More formally, given a graph $G = (V, E)$, we define the polytope P to be

$$(P) \quad \begin{array}{ll} \sum_{u \in \delta(a)} x_u \leq 1 & \forall\, a \in V \\ x_u \geq 0 & \forall\, u \in E \end{array} ,$$

where $\delta(a)$ stands for the set of edges emanating from node a. Notice that P is indeed a down-closed body. The CRS we define for P is given by Algorithm 9. The technique used by this CRS was first given by [16] (see also [21] for more details). It is important to keep in mind while reading Algorithm 9 and its analysis that the ground set here is the set of edges E, and thus, we use the term "edge" to denote the elements of this ground set.

Algorithm 9: CRS for Matching(x, A)

1 Let B be a subset of A containing every edge u of A with probability $(1 - e^{-x_u})/x_u$, independently.
2 Let $C \leftarrow \varnothing$.
3 For each $u \in B$: Add u to C only if no other edge of B shares a node with u.
4 **return** C.

Note, for example, that if two edges in B share a node, then both of them are not added to C, although it is possible that one of them could be added without violating the matching constraint. This unnatural behavior is necessary to guarantee the monotonicity of the CRS (see Lemma 1.24). One can verify that Algorithm 9 is indeed a CRS for P, *i.e.*, it always outputs a subset of the edges in its input set, and the edges in the output subset form a legal matching. The next lemma analyzes the properties of this CRS.

Lemma 1.24. *Algorithm 9 is a monotone e^{-2}-balanced CRS.*

Proof. Observe that every edge of A is copied to B with a probability independent of the membership of other edges in A. Hence, given that an edge u belongs to A, adding other edges to A can only increase the probability that some edge adjacent to u ends up in B. Since u belongs to output set C if and only if it is in B and no other edge of B shares a node with it, we get that adding other edges to A can only decrease the probability of u to belong to the output set. Thus, by definition, the CRS defined by Algorithm 9 is monotone.

It remains to analyze the balance of this CRS. Consider an arbitrary edge $u \in E$. We assume in the rest of the proof that $A \sim \mathtt{R}(x)$, and our objective is to show that $\Pr[u \in C] \geq e^{-2} \cdot x_u$. The probability that u ends up in B is

$$\Pr[u \in B] = \Pr[u \in A] \cdot \frac{1 - e^{-x_u}}{x_u} = 1 - e^{-x_u} \ .$$

Polytope	Balance	Reference
Matching Polytope	(b, e^{-2b})-balanced	[16]
Matroid Polytope	$(b, (1-e^{-b})/b)$-balanced	[17]
The natural relaxation of a knapsack constraint	$(1-\varepsilon, 1-\varepsilon)$-balanced[7]	[17]
The natural relaxation of "Independent set in an interval graph"	(b, e^{-b})-balanced	[21]

Table 1.1: The parameters of a few known CRSs. All the CRSs in this table are monotone and can be calculated efficiently.

Let a and b denote the two end nodes of u. Since the probability of every edge to get to B is independent, the probability that no other edge hitting a is in B is

$$\prod_{v \in \delta(a)-u} \Pr[v \notin B] = \prod_{v \in \delta(a)-u} (1 - \Pr[v \in B]) = \prod_{v \in \delta(a)-u} e^{-x_v} = e^{-\sum_{v \in \delta(a)-u} x_v} \geq e^{x_u - 1} \ ,$$

where the inequality holds since x belongs to the polytope P. Similarly, we get that with probability at least e^{x_u-1} no edge hitting b belongs to B. As these events are independent (no edge other than u hits both a and b), we get

$$\begin{aligned} \Pr[u \in C] &= \Pr[e \in B] \cdot \left(\prod_{v \in \delta(a)-u} \Pr[v \notin B] \right) \cdot \left(\prod_{v \in \delta(b)-u} \Pr[v \notin B] \right) \\ &\geq (1 - e^{-x_u}) \cdot (e^{x_u - 1})^2 \geq (e^{x_u} - 1) \cdot e^{-2} \geq x_u \cdot e^{-2} \ . \end{aligned}$$

□

In the last proof we used the fact that $\sum_{u \in \delta(a)} x_u \leq 1$ for every vector $x \in P$ and node $a \in V$. Observe that we would have gotten a better balance guarantee if we had $\sum_{u \in \delta(a)} x_u \leq b$ for some $b \in (0,1)$. In other words, the balance of the CRS improves when $x \in bP$. This common phenomenon motivates the following definition.

Definition 1.8. *((b,c)-balanced CRS) A CRS π for a down-closed body $P \subseteq [0,1]^{\mathcal{N}}$ is* (b,c)-balanced *(for $b, c \in [0,1]$) if $\Pr[u \in \pi_x(\mathtt{R}(x))] \geq c \cdot x_u$ for every element $u \in \mathcal{N}$ and vector $x \in bP$.*

A slight modification of the proof of Lemma 1.24 shows that the CRS given as Algorithm 9 is in fact (b, e^{-2b})-balanced for every $b \in [0,1]$. The parameters of a few other known CRSs are given by Table 1.1. One can derive from these CRSs new CRSs for more involved constraints using the following important lemma.

Lemma 1.25. *(Combining CRSs) Let π^1 be a monotone (b, c_1)-balanced CRS for a down-closed body P_1, and let π^2 be a monotone (b, c_2)-balanced CRS for a down-closed body P_2. Then, there is a monotone $(b, c_1 c_2)$-balanced CRS π for $P_1 \cap P_2$. Moreover, π can be computed efficiently whenever π^1 and π^2 can be computed efficiently.*

Proof. For every vector $x \in P_1 \cap P_2$, we define $\pi_x(A) = \pi^1_x(A) \cap \pi^2_x(A)$. One can verify that π is indeed a CRS whenever π^1 and π^2 are CRSs. To see that π is monotone, consider two sets

[7] ε can be any constant greater than 0. The use of this CRS requires a pre-processing step whose time complexity depends on ε.

$A \subseteq B \subseteq \mathcal{N}$, an element $u \in A$ and some vector $x \in P_1 \cap P_2$. Since $\pi^1_x(A)$ and $\pi^2_x(A)$ are independent when A is deterministic, the monotonicity of π^1 and π^2 implies

$$\begin{aligned}\Pr[u \in \pi_x(A)] &= \Pr[u \in \pi^1_x(A) \cap \pi^2_x(A)] = \Pr[u \in \pi^1_x(A)] \cdot \Pr[u \in \pi^2_x(A)] \\ &\geq \Pr[u \in \pi^1_x(B)] \cdot \Pr[u \in \pi^2_x(B)] = \Pr[u \in \pi^1_x(B) \cap \pi^2_x(B)] = \Pr[u \in \pi_x(B)] \ .\end{aligned}$$

It remains to analyze the balance of π. Let A be a set distributed like $\mathtt{R}(x)$. By the monotonicity of π^1 and π^2, the probabilities of the two events $u \in \pi^1_x(A)$ and $u \in \pi^2_x(A)$ are both non-increasing in A. Hence, by the FKG inequality, for every vector $x \in bP$ and element u such that $x_u > 0$,

$$\begin{aligned}\Pr[u \in \pi_x(\mathtt{R}(x))] &= x_u \cdot \Pr[u \in \pi_x(A) \mid u \in A] = x_u \cdot \Pr[u \in \pi^1_x(A) \wedge u \in \pi^2_x(A) \mid u \in A] \\ &\geq x_u \cdot \Pr[u \in \pi^1_x(A) \mid u \in A] \cdot \Pr[u \in \pi^2_x(A) \mid u \in A] \\ &= \frac{1}{x_u} \cdot \Pr[u \in \pi^1_x(\mathtt{R}(x))] \cdot \Pr[u \in \pi^2_x(\mathtt{R}(x))] \geq x_u \cdot c_1 c_2 \ .\end{aligned}$$

To complete the proof that π is (b, c_1c_2)-balanced we observe that the inequality $\Pr[u \in \pi_x(\mathtt{R}(x))] \geq x_u \cdot c_1c_2$ is trivial for an element u such that $x_u = 0$. □

Using Lemma 1.25 it is easy to derive CRSs for relaxations of very involved constraints. For example, plugging CRSs from Table 1.1 into Lemma 1.25 can yield a monotone $(b, e^{-4b} - \varepsilon)$-balanced CRS for a relaxation of a constraint allowing a set of edges only if it is a legal matching in two different graphs (over the same set of edges) and also obeys some knapsack constraint.

Finally, once we have a monotone (b, c)-balanced CRS for a solvable down-closed convex body P which is a relaxation of some constraint, it is possible to get an approximation algorithm for this constraint as summarized by the following theorem.

Theorem 1.26. *Let P be a solvable down-closed convex body, and let π be a monotone (b, c)-balanced CRS for it. Then,*

- *There is a cb-approximation algorithm for maximizing a linear function over the integral points of P.*
- *There is a (cbe^{-b})-approximation algorithm for maximizing a submodular function over the integral points of P.*
- *There is a $c(1 - e^{-b})$-approximation algorithm for maximizing a* **monotone** *submodular function over the integral points of P.*

Proof. Consider first a linear objective function f. Since we assumed P is solvable, it is possible to find a fractional solution $x \in P$ maximizing f. Since $bx \in bP$, we can feed bx into the CRS to get an integral solution whose value is, in expectation, at least $c \cdot f(bx) = bc \cdot f(x)$. Hence, the approximation ratio of the algorithm is bc.

For a submodular objective function it is possible to find an approximately optimal fractional solution $x \in P$ using Measured Continuous Greedy. We can then use bx as our vector in bP, and the approximation ratio of this vector is be^{-1} for non-monotone functions and $b(1 - e^{-1})$ for monotone functions. However, there is a better way. If one stops Measured Continuous Greedy at time $T = b$, then its output is a vector which already belongs to bP. The approximation ratio of this vector is be^{-b} for non-monotone functions and $1 - e^{-b}$ for monotone functions. Feeding this vector into the monotone (b, c)-balanced CRS we have yields an approximation algorithm whose approximation ratio is $c \cdot be^{-b}$ for non-monotone functions and $c(1 - e^{-b})$ for monotone functions. □

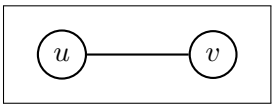

Figure 1.2: A graph inducing a simple unconstrained submodular maximization instance.

1.3.5 Inapproximability Results

In this section we describe the symmetry gap technique (originally proposed by [22]) which can be used to prove many of the state of the art inapproximability results in the field of submodular maximization. We begin by studying a very simple instance of unconstrained submodular maximization induced by the cut function of the graph depicted in Figure 1.2. Formally, our instance contains a ground set of two elements $\mathcal{N} = \{u, v\}$. The function defined over this ground set is

$$f(A) = \begin{cases} 1 & \text{if } |A| = 1 \ , \\ 0 & \text{otherwise} \ . \end{cases}$$

It is easy to verify that this function is indeed non-negative and submodular. One can also observe that u and v play completely symmetric roles in this instance. Let us assume, for the moment, that an algorithm for unconstrained submodular maximization must output a vector $x \in [0,1]^{\mathcal{N}}$ which is symmetric with respect to every symmetry that its input instance have. In particular, for the instance we consider, it must be that $x_u = x_v = y$ for some value $y \in [0,1]$. The value of any such symmetric vector, with respect to the multilinear extension F of f, is

$$F(x) = F(y \cdot \mathbf{1}_{\mathcal{N}}) = 2y(1-y) \leq \frac{1}{2} \ .$$

On the other hand, there is an integral solution for this instance (for example, the vector $(1,0)$) whose value is 1. Hence, there is a multiplicative gap of $(1/2) : 1 = 1/2$ between the best symmetric and non-symmetric (feasible) fractional solutions of this instance. This gap is called the *symmetry gap*.

On the face of it, the symmetry gap does not seem to be related to inapproximability. After all, real algorithms output integral solutions, and these solutions are allowed to be asymmetric. However, it turns out that there is a way to force polynomial time algorithms to output a close to symmetric fractional solution. As a first step in that direction, let us "blow" each element of the above instance into t elements. This yields a new ground set that we denote by $\mathcal{N}_t = \{u_i, v_i\}_{i=1}^{t}$. The objective function that we associate with this ground set is

$$f_t(A) = F(t^{-1} \cdot |A \cap \{u_i\}_{i=1}^{t}|, t^{-1} \cdot |A \cap \{v_i\}_{i=1}^{t}|) \ .$$

Informally, every set $A \subseteq \mathcal{N}_t$ is interpreted as a fractional solution over the ground set $\mathcal{N}$, where the fraction of u is proportional to the number of $\{u_i\}_{i=1}^{t}$ elements in A, and the the fraction of v is proportional to the number of $\{v_i\}_{i=1}^{t}$ elements in A. It can be verified that the resulting objective function f_t is non-negative and submodular whenever the objective function f of the original instance has these properties (this follows, for example, from the more general results of [22]).

Assume that the elements of $\mathcal{N}_t$ are assigned names, uniformly at random, from the list $1, 2, \ldots, 2t$, and the algorithm can access the elements only via these names (*i.e.*, the algorithm has no access to the original names of the elements). The randomness of the names assignment means that

the algorithm does not know which of the elements descend from u and which descend from v. If the algorithm fails to find out more information about the origin of the different elements, then its output set will necessarily be chosen independently of this information. Thus, assuming t is large, the output set is likely to contain a similar number of elements descending from u and v. In other words, if the algorithm cannot find information about the origin of the different elements, then its output is close to being a symmetric fractional solution for the original (non-blown) instance; and thus, its approximation ratio cannot be significantly better than the symmetry gap.

The only way in which the algorithm can gather information is through the value oracle. To query this oracle the algorithm constructs a set A and forwards it to the oracle. Note that when the number of element in A originating from u and v is equal, then the response of the oracle is

$$F(t^{-1} \cdot |A|/2, t^{-1} \cdot |A|/2) = 2 \cdot \frac{|A|}{2t}\left(1 - \frac{|A|}{2t}\right) = \frac{|A|}{t}\left(1 - \frac{|A|}{2t}\right) \ ,$$

which is a function of the size of A alone. Hence, queries about sets with equal number of elements originating from u and v does not give the algorithm any information about the origin of the different elements. Thus, getting information about the origin of the elements requires the algorithm to construct a set which is unbalanced in terms of the number elements in it originating from u and v.

By a slight modification of f_t, one can further guarantee that getting information about the origin of the elements requires the algorithm to construct a set which is *significantly* unbalanced in terms of the number elements in it originating from u and v. However, such a set cannot be constructed without prior knowledge about the origin of elements because any set constructed without such knowledge is likely, for large values of t, to contain a similar number of elements descending from u and v.

This concludes our intuitive explanation about how to convert the symmetry gap of unconstrained submodular maximization into an inapproximability result. The next theorem, due to [6], gives a formal proof, based on this idea, of this inapproximability.

Theorem 1.27 (Due to [6])**.** *For every constant $\varepsilon > 0$, there is no polynomial time $(1/2 + \varepsilon)$-approximation algorithm for unconstrained submodular maximization.*

Proof. For simplicity, we prove the equivalent claim that there is no polynomial time $(1/2 + 3\varepsilon)$-approximation algorithm for unconstrained submodular maximization. This claim is meaningless for $\varepsilon > 1/6$, thus, we assume $\varepsilon \in (0, 1/6]$. Additionally, we also assume $1/\varepsilon$ is integral (otherwise, one can replace ε with some value from the range $[\varepsilon/2, \varepsilon]$ having this property).

Let t be an arbitrary large positive integer dividable by $1/\varepsilon$ (meaning that εt is integral), and let $\mathcal{N}_t = \{1, 2, \ldots, 2t\}$. Let $U \subseteq \mathcal{N}_t$ be an arbitrary subset of size t. Intuitively, the elements of U are the elements originating from u in the above general description of the proof. For every set $A \subseteq \mathcal{N}_t$ let $u_A = |A \cap U|$ and $v_A = |A \setminus U|$. Using this notation we define a function $f_t \colon 2^{\mathcal{N}_t} \to \mathbb{R}_{\geq 0}$ as follows.

$$f_t(A) = \begin{cases} \frac{u_A + v_A}{t}\left(1 - \frac{u_A + v_A}{2t}\right) = \frac{|A|}{t}\left(1 - \frac{|A|}{2t}\right) & \text{when } |u_A - v_A| \leq \varepsilon t \ , \\ \frac{u_A}{t}\left(1 - \frac{v_A}{t}\right) + \frac{v_A}{t}\left(1 - \frac{u_A}{t}\right) + \frac{\varepsilon^2}{2} - \frac{\varepsilon |u_A - v_A|}{t} & \text{when } |u_A - v_A| \geq \varepsilon t \ . \end{cases}$$

The definition of this function in the case $|u_A - v_A| \geq \varepsilon t$ is identical to the definition of the function f_t from the intuitive description above up to the additional terms involving ε. These additional terms change the maximum of the function very little, but allow for the two cases of the definition to combine into one submodular function. For ease of the reading, we omit the technical verification of the following claim.

Claim. *f_t is a non-negative submodular function. Its maximum value $1+\varepsilon^2/2-\varepsilon \in (1-\varepsilon,1)$ is attained both when $u_A = t$ and $v_A = 0$ and when $u_A = 0$ and $v_A = t$.*

The definition of f_t depends on the set U which we assumed so far to be fixed. From now on we assume it is a uniformly random subset of $\mathcal{N}_t$ of size t. Hence, f_t is now a sample out of a distribution of non-negative submodular functions. All the functions in the support of this distribution share the same maximum value. We would like to show that, given access to a function f_t sampled from this distribution, no polynomial time algorithm can output, in expectation, a solution of value at least $1/2+\varepsilon$ times this optimal value (where the expectation is over the randomness of U and the random coins of the algorithm). By Yao's principal, it is enough to prove this for deterministic algorithms (if there is a single distribution over instances which is hard for any given polynomial time deterministic algorithm, then Yao's principal guarantees that this distribution is hard also for polynomial time randomized algorithms); and thus, we restrict ourselves to deterministic algorithms in the rest of the proof.

Consider an arbitrary polynomial time deterministic algorithm ALG for unconstrained submodular maximization, and consider an execution of this algorithm on the instance induced by the following non-negative submodular function.

$$g_t(A) = \frac{|A|}{t}\left(1-\frac{|A|}{2t}\right) \ .$$

Let $Q_1, Q_2, \ldots, Q_r$ be the list of sets that ALG passes as queries to its value oracle given this instance. Since ALG is polynomial, r (the number of queries) is bounded by a polynomial function of t. Additionally, we can assume, without loss of generality, that the output set of ALG is one of the sets in this list. For every set Q_i, $u_{Q_i} = |Q_i \cap U|$ has a hypergeometric distribution, and thus, by bounds given in [23] (which are based on results of [24, 25]) we get

$$\begin{aligned}\Pr[|u_{Q_i} - v_{Q_i}| \geq \varepsilon t] &= \Pr[|2|Q_i \cap U| - |Q_i|| \geq \varepsilon t] = \Pr\left[||Q_i \cap U| - \mathbb{E}[|Q_i \cap U|]| \geq |Q_i| \cdot \frac{\varepsilon t}{2|Q_i|}\right] \\ &\leq 2e^{-2\cdot\left(\frac{\varepsilon t}{2|Q_i|}\right)^2 \cdot |Q_i|} = 2e^{-\frac{\varepsilon^2 t^2}{2|Q_i|}} \leq 2e^{-\varepsilon^2 t/4} \ ,\end{aligned}$$

where the last inequality holds since $|Q_i| \leq |\mathcal{N}_t| = 2t$. Notice that the event $|u_{Q_i} - v_{Q_i}| \leq \varepsilon t$ implies $f_t(Q_i) = g_t(Q_i)$. Hence, what we proved is in fact that $f_t(Q_i) = g_t(Q_i)$ with probability at least $1 - 2e^{-\varepsilon^2 t/4}$ for every given set Q_i. By the union bound, we get that, with probability at least $1 - 2re^{-\varepsilon^2 t/4}$, $f_t(Q_i) = g_t(Q_i)$ for every $1 \leq i \leq r$. Let us denote by $\mathcal{E}$ the last event (*i.e.*, the event that $f_t(Q_i) = g_t(Q_i)$ for every $1 \leq i \leq r$).

The main observation of the proof is that when $\mathcal{E}$ happens ALG has an identical execution given either f_t or g_t. Thus, $\mathcal{E}$ guarantees that given both inputs ALG outputs the same output set, and this set has the same value under both functions because we assumed that it is one of the sets in the list $Q_1, Q_2, \ldots, Q_r$. Since the maximum value of g_t is $1/2$, this implies that ALG outputs a set of value at most $1/2$ whenever $\mathcal{E}$ occurs. We can now upper bound the expected value of the output of ALG given f_t (where the expectation is over the randomness of U) by

$$\frac{1}{2} \cdot \Pr[\mathcal{E}] + \max_{A \subseteq \mathcal{N}_t} f_t(A) \cdot (1 - \Pr[\mathcal{E}]) \leq \frac{1}{2} \cdot 1 + 1 \cdot (1 - \Pr[\mathcal{E}]) \leq \frac{1}{2} + 2re^{-\varepsilon^2 t/4} \leq \frac{1}{2} + \varepsilon \ ,$$

where the last inequality holds for large enough t since r is polynomial in t. On the other hand, by the claim the maximum value of f_t is at least $1-\varepsilon$, and thus, the approximation ratio of ALG is worse than

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

This completes the proof of the theorem by the above discussion. □

In general, the work of [22] shows that the symmetry gap can be converted into an inapproximability result in many cases. We do not state the results of [22] explicitly here, however, intuitively, the conversion of the symmetry gap into an inapproximability result can be done whenever the following two conditions hold.

- The symmetries of the objective function that one uses in order to calculate the symmetry gap cannot be broken by the algorithm using some bypass. In particular, the constraint of the problem should obey these symmetries as well.
- As seen in the proof of Theorem 1.27, the hardness result applies in fact to a distribution over blown up instances. Hence, it should be possible to somehow present blown up instances as new instances of the original problem for which the inapproximability result should apply.

To better understand these conditions, we now give an informal proof based on the symmetry gap technique for another inapproximability result—the problem of maximizing a monotone submodular function subject to a cardinality constraint. Consider an instance of this problem in which one is allowed to pick up to one element out of some ground set $\mathcal{N}$ of size n. The objective function of the instance is the monotone submodular function $f(A) = \min\{|A|, 1\}$. Clearly all the elements of the ground set are symmetric with respect to both this objective function and the constraint. Hence, any symmetric fractional solution must assign an identical value to each one of the elements. To keep this fractional solution feasible, this identical value must be at most $1/n$. Thus, the best feasible symmetric fractional solution is $n^{-1} \cdot \mathbf{1}_{\mathcal{N}}$, and its value with respect to the multilinear extension F of f is

$$F(n^{-1} \cdot \mathbf{1}_{\mathcal{N}}) = 1 - (1 - 1/n)^n \ .$$

One the other hand, any set containing a single element of $\mathcal{N}$ is feasible and has a value of 1. Hence, the symmetry gap of this instance of maximizing a monotone submodular function subject to a cardinality constraint is $[1 - (1 - 1/n)^n]/1 = 1 - (1 - 1/n)^n$.

When blowing up the above instance by a factor of t one gets a new instance with a ground set of size nt in which a feasible solution contains at most t elements. Clearly, this blown up instance is a valid instance of maximizing a monotone submodular function subject to a cardinality constraint, and thus, the two intuitive conditions given above hold. This means that the symmetry gap we have shown translates into an inapproximability result for this problem. In conclusion, we got that no polynomial time algorithm can achieve an approximation ratio of $1 - (1 - 1/n)^n + \varepsilon$ for any constant $\varepsilon > 0$ and integer n. As $(1 - 1/n)^n$ approaches $1/e$ when n grows, this implies that no polynomial time algorithm can achieve an approximation ratio of $1 - 1/e + \varepsilon$ for any constant $\varepsilon > 0$. This inapproximability result is optimal (as shown in Section 1.2.1). It was first proved by [2] using a different technique.

It is interesting to note that the inapproximability result proved by Theorem 1.27, like all the other inapproximability results discussed in this chapter, is unconditional. In particular, this inapproximability result does not relay on complexity assumptions such as $P \neq NP$. Instead, its proof argues that a polynomial number of value oracle queries does not suffice in order to distinguish between two functions f_t and g_t having very different maximum values—which is an information theoretic argument. The use of this information theoretic argument means that our assumption that the objective function can be accessed only via a value oracle is essential for the proof. More specifically, the proof does not hold if the objective function is specified to the algorithm using some succinct representation. Dobzinski and Vondrák [26] showed that inapproximability results based

on the symmetry gap, such as the inapproximability result proved by Theorem 1.27, can usually be modified to allow some succinct representation of the input functions. This modification, however, comes at the cost of loosing the unconditionality of the result and relaying on the widely believed complexity assumption $RP \neq NP$.

1.3.6 Notes and Open Questions

Measured Continuous Greedy: The $1/e$-approximation guarantee of Measured Continuous Greedy for non-monotone functions on down-closed convex body P is not known to be the best possible. For this problem it is only known that no polynomial time algorithm can output a fractional solution of value at least $0.478 \cdot f(OPT)$ [9]. Closing the gap between the guarantee of Measured Continuous Greedy and this impossibility result is one of the most important open problems related to the topics presented in this section.

Rounding Techniques: The fractional solutions produced by Continuous Greedy and Measured Continuous Greedy are of little use without rounding algorithms that convert their fractional solutions into integral solutions. In Section 1.3.4 we presented a powerful technique which yields rounding algorithms for many types of constraints. We would like to mention a few rounding algorithms which are based on other techniques.

The problem of maximizing a submodular function subject to a matroid constraint[8] was one of the first problems studied in the context of submodular functions maximization [8]. Matroid constraint generalizes, for example, both the cardinality constraint and the constraints in SW. Two rounding algorithms called Pipage Rounding and Swap Rounding were suggested by [14] and [27], respectively, for the multilinear relaxation of this problem in which the convex body P is the matroid polytope. Both rounding techniques loose nothing in the objective, and thus, yield an optimal $(1 - 1/e)$-approximation algorithm for maximizing a non-negative monotone submodular function subject to a matroid constraint as well as a $(1/e - o(1))$-approximation for the non-monotone counterpart of this problem. Swap Rounding, suggested later, is faster than Pipage Rounding and also enjoys additional concentration bounds which make it more suitable for some applications.

Kulik et al. [28] designed a rounding algorithm for the natural multilinear relaxation of the problem of maximizing a non-negative monotone submodular function subject to a *constant* number of knapsack constraints. This rounding algorithm looses only a factor of $1 - \varepsilon$ (for any fixed $\varepsilon > 0$), and thus, combined with Continuous Greedy, implies $(1 - 1/e - \varepsilon)$-approximation for this problem. Notice that this result is optimal up to the term ε since the problem generalizes maximization of a submodular function subject to a cardinality constraint. In a later work Kulik et al. [29] extended their rounding algorithm also to non-monotone functions, which, combined with Measured Continuous Greedy, yields a $(1/e - \varepsilon)$-approximation algorithm for the non-monotone counterpart of the above problem.

Inapproximability: Despite its origin in the field of submodular maximization, the symmetry gap technique found some applications also in the study of submodular minimization problems [30].

[8] Unfortunately, the definition of matroids is outside the scope of this survey; however, we mention a few results concerning matroid constraints for the sake of more advanced readers.

References

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

[2] G. L. Nemhauser and L. A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of Operations Research*, 3(3):177–188, 1978.

[3] Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *SODA*, pages 1433–1452, 2014.

[4] Niv Buchbinder, Moran Feldman, and Roy Schwartz. Comparing apples and oranges: Query tradeoff in submodular maximization. In *SODA*, pages 1149–1168, 2015.

[5] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *AAAI*, pages 1812–1818, 2015.

[6] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM J. Comput.*, 40(4):1133–1153, 2011.

[7] Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *SIAM J. Comput.*, 44(5):1384–1402, 2015.

[8] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions – II. *Mathematical Programming Study*, 8:73–87, 1978.

[9] Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *SODA*, pages 1098–1116, 2011.

[10] Niv Buchbinder and Moran Feldman. Deterministic algorithms for submodular maximization problems. In *SODA*, pages 392–403, 2016.

[11] Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *SODA*, pages 1497–1514, 2014.

[12] Moran Feldman. Maximizing symmetric submodular functions. In *ESA*, pages 521–532, 2015.

[13] Vahab S. Mirrokni, Michael Schapira, and Jan Vondrák. Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions. In *EC 2008*, pages 70–77, 2008.

[14] Gruia Călinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.*, 40(6):1740–1766, 2011.

[15] László Lovász. Submodular functions and convexity. In A. Bachem, M. Grötschel, and B. Korte, editors, *Mathematical Programming: the State of the Art*, pages 235–257. Springer, 1983.

[16] Moran Feldman, Joseph Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *FOCS*, pages 570–579, 2011.

[17] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM J. Comput.*, 43(6):1831–1879, 2014.

[18] Anupam Gupta and Viswanath Nagarajan. A stochastic probing problem with applications. In *IPCO*, pages 205–216, 2013.

[19] Marek Adamczyk. Non-negative submodular stochastic probing via stochastic contention resolution schemes. *CoRR*, abs/1508.07771, 2015.

[20] Moran Feldman, Ola Svensson, and Rico Zenklusen. Online contention resolution schemes. In *SODA*, pages 1014–1033, 2016.

[21] Moran Feldman. *Maximization Problems with Submodular Objective Functions*. PhD thesis, Computer Science Department, Technion - Israel Institute of Technology, 2013.

[22] Jan Vondrák. Symmetry and approximability of submodular maximization problems. *SIAM J. Comput.*, 42(1):265–304, 2013.

[23] Matthew Skala. Hypergeometric tail inequalities: ending the insanity. *CoRR*, abs/1311.5939, 2013.

[24] Vaclav Chvátal. The tail of the hypergeometric distribution. *Discrete Mathematics*, 25(3):285–287, 1979.

[25] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[26] Shahar Dobzinski and Jan Vondrák. From query complexity to computational complexity. In *STOC*, pages 1107–1116, 2012.

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

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

[29] Ariel Kulik, Hadas Shachnai, and Tami Tamir. Approximations for monotone and nonmonotone submodular maximization with knapsack constraints. *Math. Oper. Res.*, 38(4):729–739, 2013.

[30] Alina Ene, Jan Vondrák, and Yi Wu. Local distribution and the symmetry gap: Approximability of multiway partitioning problems. In *SODA*, pages 306–325, 2013.

Index