Maximization Problems with Submodular Objective Functions

Moran Feldman

Maximization Problems with Submodular Objective Functions

Research Thesis

In Partial Fulfillment of the
Requirements for the
Degree of Doctor of Philosophy

Moran Feldman

Submitted to the Senate of
the Technion — Israel Institute of Technology
Tammuz 5773 Haifa June 2013

The Research Thesis was done under the Supervision of Prof. Joseph (Seffi) Naor at the Computer Science Department.

I would like to thank Seffi, my advisor, who wisely led me and graciously guided me throughout the years of my work. For pointing out many interesting new research directions. For working with me through the nights when a deadline approached. For introducing me to many other interesting researchers. And most of all for your great support and encouragement, especially when papers were rejected time after time. I highly appreciate the good fortune of having the chance to work with you.

The Generous Financial Help of the Google European Fellowship in Market Algorithms and the Technion is Gratefully Acknowledged.

Publication List:

1. Moran Feldman, Joseph(Seffi) Naor, Roy Schwartz and Justin Ward, "Improved approximations for k-exchange systems", In *Proceedings of the 19th Annual European Symposium on Algorithms, ESA11*, Saarbrücken, Germany, 2011.

2. Moran Feldman, Joseph(Seffi) Naor and Roy Schwartz, "A unified continuous greedy algorithm for submodular maximization.", In *Proceedings of the 52nd annual IEEE Symposium on Foundations of Computer Science, FOCS11*, Palm Springs, California, USA, 2011.

3. Niv Buchbinder, Moran Feldman, Joseph(Seffi) Naor and Roy Schwartz, "A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization.", In *Proceedings of the 53nd annual IEEE Symposium on Foundations of Computer Science, FOCS12*, New Brunswick, New Jersey, USA, 2012.

Contents

Figures

Tables

Abstract

The study of combinatorial problems with submodular objective functions has attracted much attention recently, and is motivated by the principle of economy of scale, prevalent in real world applications. Moreover, submodular functions are commonly used as utility functions in economics and algorithmic game theory. From a theoretical perspective, submodular functions and submodular optimization play a major role in combinatorics, graph theory and combinatorial optimization.

In this thesis, we consider a few constrained and unconstrained submodular maximization problems. These problems obey the following general structure. Given a submodular function f and (possibly) a set of constraints, find a feasible set S maximizing $f(S)$. The problems we consider in this thesis cannot be solved exactly in polynomial time due to hardness results which are based on information-theoretic arguments. Instead, we describe approximation algorithms for these problems, achieving the best possible approximation ratios for some of the problems.

Our approximation algorithms can be roughly partitioned based on the technique they use. The first approach is combinatorial in nature, and is mostly based on local search techniques and greedy rules. The second approach resembles a common paradigm for designing approximation algorithms and is composed of two steps. In the first step, a fractional solution is found for a relaxation of the problem. In the second step, the fractional solution is rounded to obtain an integral one while incurring only a small loss in the objective.

Notation and Abbreviations

$\mathcal{N}$ — Ground set
n — Number of elements in the ground set
A, B, S — Subsets of the ground set
u — An element of the ground set
$S + u$ — Synonym for the union $S \cup \{u\}$
$S - u$ — Synonym for the expression $S \setminus \{u\}$
f — Objective set function
f^- — The convex closure of f
f^+ — The concave closure of f
$\hat{f}$ — The Lovász extension of f
F — The multilinear extension of f
x, y — Vectors in the cube $[0,1]^{\mathcal{N}}$
$x \vee y$ — The coordinate-wise maximum of x and y (formally, $(x \vee y)_u = \max\{x_u, y_u\}$)
$x \wedge y$ — The coordinate-wise minimum of x and y (formally, $(x \wedge y)_u = \min\{x_u, y_u\}$)
$T_\lambda(x)$ — The set of all elements $u \in \mathcal{N}$ having $x_u \geq \lambda$
$\partial_u F(x)$ — The derivative of F, with respect to u, at x
1_S — The characteristic vector of a set S
M — Matroid
I — Collection of independent sets of a sets system (subset of $2^{\mathcal{N}}$)
$\mathcal{P}(M)$ — The matroid polytope corresponding to M
$\mathcal{B}(M)$ — The bases polytope corresponding to M

Chapter 1

Introduction

The study of combinatorial problems with submodular objective functions has attracted much attention recently, and is motivated by the principle of economy of scale, prevalent in real world applications. Moreover, submodular functions are commonly used as utility functions in economics and algorithmic game theory. From a theoretical perspective, submodular functions and submodular optimization play a major role in combinatorics, graph theory and combinatorial optimization.

A set function is a function that gives a numerical value to every subset of a given ground set. A submodular function is a set function with the following property: the marginal value of adding an element to a set is non-increasing as more elements are added to it. Several well known examples for submodular functions include cuts in directed and undirected graphs, rank functions in matroids, covering functions and cuts in hypergraphs [74]. In a submodular optimization problem the objective is to find a set that either maximizes or minimizes a submodular function subject to problem specific constraints on the allowed sets. Many optimization problems can be represented as constrained variants of submodular optimization. A partial list of classical well-studied problems captured by submodular optimization includes: Max Cut, Max k-Cover, Generalized Assignment, several variants of Max SAT and some welfare and scheduling problems. Hence, finding algorithms with good approximation ratios for submodular optimization problems will induce similar algorithms for the above problems and many others. Moreover, since many of the problems considered here are motivated by real life applications, it is important to find algorithms that guarantee both good approximations and are as simple and efficient as possible.

In this thesis, we consider a few constrained and unconstrained submodular maximization problems. These problems obey the following general structure. Given a submodular function f and (possibly) a set of constraints, find a feasible set S maximizing $f(S)$. The problems we consider in this thesis cannot be solved exactly in polynomial time due to hardness results which are based on information-theoretic arguments. Instead, we describe approximation algorithms for these problems, achieving the best possible approximation ratios for some of the problems.

The techniques used to compute approximate solutions to various submodular maximization problems can be partitioned into two main approaches. The first approach is combinatorial in nature, and is mostly based on local search techniques and greedy rules. This approach has been used as early as the late 70's for maximizing monotone submodular functions under the constraint that the solution should be an independent set of one of several specific matroids [22, 36, 46, 47, 50, 56, 71, 72]. Lately, this approach has been extended to include both non-monotone submodular objective functions [30, 37, 34, 81] and additional constraints sets [62] (*e.g.*, independent sets of matroids intersection). Sec-

tions 4 and 5 describe algorithms that can be classified under this approach. Section 4 addresses the basic problem of maximizing a submodular function under no constraints, as well as a few related problems. Section 5 considers the problem of maximizing an objective function subject to a k-exchange set system constraint.

The second approach for approximating submodular maximization problems resembles a common paradigm for designing approximation algorithms and is composed of two steps. In the first step, a fractional solution is found for a relaxation of the problem. In the second step, the fractional solution is rounded to obtain an integral one while incurring only a small loss in the objective. Despite the combinatorial association of submodular functions, this approach has been used to obtain many state-of-the-art tight approximation results [16, 18, 19, 20, 58, 61]. Most notable of these results is an *asymptotically tight* approximation for maximizing a monotone submodular function given a single matroid constraint [16, 71, 72]. Two issues arise when using this approach. First, since the objective function is not linear, it is not clear how to solve or even efficiently approximate a relaxation of the problem. Second, given a fractional solution, one needs a rounding procedure which outputs an integral solution without losing too much in the objective function. Sections 3 and 6 address these two issues, respectively.

Section 3 describes a solver for the multilinear relaxation, which is the standard relaxation for submodular maximization problems. For many problems this solver improves over the best approximation ratio achieved by any other known solver. Section 6 describes several extensions for the Contentions Resolution Scheme of [20]. This scheme describes a standard way for rounding fractional solutions of submodular maximization problems. Our extensions allow the scheme to address additional types of submodular maximization problems, and improve the approximation ratio achieved by the scheme for many other problems.

Chapter 2

Preliminaries

2.1 Submodular Functions and Matroids

For every set S and an element u, we denote the union $S \cup \{u\}$ by $S+u$, and the expression $S \setminus \{u\}$ by $S-u$. Throughout this work, we assume the existence of some ground set $\mathcal{N}$ whose size is denoted by n. A function $f : 2^{\mathcal{N}} \to \mathbb{R}$ assigning a number for every subset of $\mathcal{N}$ is called a *set function*. Following is a list of properties that a set function might obey.

- f is non-negative if $f(S) \geq 0$ for every set $S \subseteq \mathcal{N}$.
- f is normalized if $f(\varnothing) = 0$.
- f is monotone if $f(A) \leq f(B)$ for every two sets $A \subseteq B \subseteq \mathcal{N}$.
- f is submodular if $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$.

The first three properties are self explanatory, but submodularity is more involved. Submodular functions are discrete analogs of convex functions. For a submodular function, the marginal contribution of an element to a set decreases as the set gets larger. In fact, it can be easily proved that a function is submodular if and only if for every element $u \in \mathcal{N}$ and two sets $A \subseteq B \subseteq N - u$:

$$f(A+u) - f(A) \geq f(B+u) - f(B) \ .$$

A set system is a pair $(\mathcal{N}, \mathcal{I})$, where $\mathcal{N}$ is, as usual, a ground set, and $\mathcal{I} \subseteq 2^{\mathcal{N}}$ is a collection of subsets of $\mathcal{N}$. The collection $\mathcal{I}$ must obey the following two properties:

- Non-empty: $\mathcal{I} \neq \emptyset$.
- Monotone: If $A \subseteq B \in \mathcal{I}$, then $A \in \mathcal{I}$.

If $S \in \mathcal{I}$, we say that S is *independent*. A maximal (inclusion-wise) independent set is called *base*. There are many classes of set systems, the most important of which is the *matroids* class. Other classes of set systems are described in Section 5. A set system is a matroid (or belongs to the matroids class) if it obeys the following extra property:

- Matroid Exchange: If $A, B \in \mathcal{I}$ and $|A| < |B|$, then there exists an element $u \in B$ for which $A + u \in \mathcal{I}$.

Matroids capture many natural collections of subsets such as: forests in graphs, independent sets in vector spaces and the sets of nodes that appear together in legal matchings of a given graph. [74, 59]

2.2 Problems Considered

All the problems considered in this thesis fall into the following pattern: given a non-negative (monotone) submodular function f and a collection $\mathcal{C} \subseteq 2^{\mathcal{N}}$ of subsets, find a set $S \in \mathcal{C}$ maximizing $f(\mathcal{C})$. One example for such a problem is `maximizing a non-negative monotone submodular function subject to a matroid constraint`. The objective in this problem is to find a set S maximizing a non-negative, monotone and submodular function f among the independent sets of a given matroid $M = (\mathcal{N}, \mathcal{I})$. More formally, in this problem the collection $\mathcal{C}$ of allowed sets coincides with the collection $\mathcal{I}$.

We look for algorithms that are polynomial in n, the size of $\mathcal{N}$. However, the explicit representation of both submodular functions and matroids might be exponential in the size of their ground set. The standard way to bypass this difficulty is to assume access to these objects via oracles. For a submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}$, given a set $S \subseteq \mathcal{N}$, the oracle returns the value of $f(S)$.[1] For a matroid $M = (\mathcal{N}, \mathcal{I})$, given a set $S \subseteq \mathcal{N}$, the oracle answers whether $S \in \mathcal{I}$. All the algorithms we describe access submodular functions and matroids via such oracles.

2.3 Extensions to the $[0,1]^{\mathcal{N}}$ Cube

The collection of sets $2^{\mathcal{N}}$ has a natural extension to the $[0,1]^{\mathcal{N}}$ cube, where each set S corresponds to its characteristic vector 1_S. For linear functions, the extension from $2^{\mathcal{N}}$ to the cube $[0,1]^{\mathcal{N}}$ is easily defined. However, finding the right extension for submodular functions is more difficult. Two natural extensions are the convex closure and the concave closure. The convex (concave) closure of a function $f : 2^{\mathcal{N}} \to \mathbb{R}$ is the point-wise largest convex (smallest concave) function from $[0,1]^{\mathcal{N}}$ to $\mathbb{R}$ that lower bounds (upper bounds) f. The convex (concave) closure of f is denoted by f^- (f^+).

Observation 2.3.1. *The convex and concave closures exist for every function $f : 2^{\mathcal{N}} \to \mathbb{R}$.*

Proof. We prove the observation for the convex closure. The proof for the concave closure is analogous. Fix a function $f : 2^{\mathcal{N}} \to \mathbb{R}$. Let $\mathcal{F}$ be the set of functions from $[0,1]^{\mathcal{N}}$ to $\mathbb{R}$ that are convex and lower bound f. Consider the function $f'(x) = \max\{g(x) | g \in \mathcal{F}\}$. Clearly f' is convex and lower bounds f since it is the maximum of a set of functions which are all convex and lower bound f. Moreover, f' is point-wise larger than every other function in $\mathcal{F}$. Thus, f' is the convex closure of f. □

The convex and concave closures were presented as extensions of f. However, this is not obvious from their definitions. To see that they are indeed extensions of f, we need the following lemma.

Lemma 2.3.2. *Fix a set function $f : 2^{\mathcal{N}} \to \mathbb{R}$. For every $x \in [0,1]^{\mathcal{N}}$, let $D_f^-(x)$ ($D_f^+(x)$) denote a distribution over $2^{\mathcal{N}}$ with marginals x minimizing $\mathbb{E}_{S \sim D_f^-(x)}[f(S)]$ (maximizing $\mathbb{E}_{S \sim D_f^+(x)}[f(S)]$), breaking ties arbitrary. Then:*

- $f^- = \mathbb{E}_{S \sim D_f^-(x)}[f(S)]$.
- $f^+ = \mathbb{E}_{S \sim D_f^+(x)}[f(S)]$.

[1] Such an oracle is called *value oracle*. Other, stronger, oracle types for submodular functions are also considered in the literature, but value oracles are probably the most widely used.

Proof. We prove the lemma for the convex closure. The proof for the concave closure is analogue. Consider the function $g = \mathbb{E}_{S\sim D_f^-(x)}[f(S)]$. Let us prove that g is convex. Consider any three points x, y, z so that $z = \lambda \cdot x + (1-\lambda)\cdot y$ (for some $0 \leq \lambda \leq 1$). Clearly,

$$\begin{aligned} g(z) &\leq \mathbb{E}_{S\sim[\lambda\cdot D_f^-(x)+(1-\lambda)\cdot D_f^-(y)]}[f(S)] \\ &= \lambda \cdot \mathbb{E}_{S\sim D_f^-(x)}[f(S)] + (1-\lambda)\cdot \mathbb{E}_{S\sim D_f^-(y)]}[f(S)] = g(x) + g(y) \ . \end{aligned}$$

The convexity of g, plus the fact that g agrees with f on every point $x \in \{0,1\}^{\mathcal{N}}$, proves that $g \leq f^-$. On the other hand, since f^- is convex, for every $x \in [0,1]^{\mathcal{N}}$:

$$f^-(x) \leq \mathbb{E}_{S\sim D_f^+(x)}[f^-(1_S)] \leq \mathbb{E}_{S\sim D_f^+(x)}[f(S)] = g(x) \ . \qquad \square$$

The last lemma implies that both f^- and f^+ agree with f on every point $x \in \{0,1\}^{\mathcal{N}}$, and therefore, they are indeed extensions of f. The following two sections consider uses of the convex and concave closures, as well as of two other extensions known as the Lovász extension and the multilinear extension.

2.3.1 The Convex Closure and the Lovász Extension

Convex extensions such as the convex closure are useful for minimization problems when they can be efficiently evaluated. However, both the definition and the characterization by Lemma 2.3.2 do not provide an efficient evaluation method for the convex closure. For this reason, we need to introduce the Lovász extension.

Given a vector $x \in [0,1]^{\mathcal{N}}$ and a scalar $\lambda \in [0,1]$, let $T_\lambda(x)$ be the set of elements in $\mathcal{N}$ whose coordinate in x is at least λ. The Lovász extension of a set function $f : 2^{\mathcal{N}} \to \mathbb{R}$ is defined as:

$$\hat{f}(x) = \int_0^1 f(T_\lambda(x))d\lambda \ .$$

This definition can also be interpreted in probabilistic terms as the expected value of f over the set $T_\lambda(x)$, where λ is uniformly selected from the range $[0,1]$. The Lovász extension has the following very useful property proved in [64].

Theorem 2.3.3. *A set function $f : 2^{\mathcal{N}} \to \mathbb{R}$ is submodular if and only if its Lovász extension is convex.*

Notice that the last theorem is especially interesting since we usually see submodular functions as a discrete variant of concave functions.

Corollary 2.3.4. *For a submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}$, $f^- = \hat{f}$.*

Proof. By definition $f^- \geq \hat{f}$ since $\hat{f}$ is convex. To see why $f^- \leq \hat{f}$, observe that $\hat{f}(x) = \mathbb{E}_{S\sim D}[f(S)]$ for some distribution D, and by Lemma 2.3.2 $f^- \leq \mathbb{E}_{S\sim D}[f(S)]$ for any distribution D. $\square$

The last corollary gives us a method for calculating the convex closure of submodular functions. This method is also one of the key ingredients in the first polynomial algorithm for unconstrained submodular minimization[2] given by Grötschel et al. [65]. This algorithm uses the ellipsoid method to minimize the Lovász extension of a submodular function f, and then rounds the resulting vector without loosing in the objective.

[2]The problem of finding a set minimizing a given submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$ over the entire collection $2^{\mathcal{N}}$.

2.3.2 The Concave closure and the Multilinear Extension

For maximization problems, we need concave extensions that can be efficiently evaluated. An obvious candidate for such an extension is the concave closure. Unfortunately, however, it is NP-hard to evaluate the concave closure (even for graph cut functions) [80]. Instead, we use the multilinear extension which is not concave, but has some concave-like properties.

Given a vector $x \in [0,1]^{\mathcal{N}}$, the random set $\mathtt{R}(x) \subseteq \mathcal{N}$ contains every element $u \in \mathcal{N}$ with probability x_u. Given a set function $f : 2^{\mathcal{N}} \to R$, its multilinear extension of f is denoted by F. For any vector $x \in [0,1]^{\mathcal{N}}$, the value of $F(x)$ is the expected value of f over the random subset $\mathtt{R}(x)$. Formally, for every $x \in [0,1]^{\mathcal{N}}$, $F(x) \triangleq \mathbb{E}[\mathtt{R}(x)] = \sum_{S \subseteq \mathcal{N}} f(S) \prod_{u \in S} x_u \prod_{u \notin S} (1 - x_u)$. The following theorem relates the multilinear and Lovász extensions.

Theorem 2.3.5 (Lemma A.4 in [81])**.** *Let $F(x)$ and $\hat{f}(x)$ be the multilinear and Lovász extensions, respectively, of a submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}$. Then, $F(x) \geq \hat{f}(x)$ for every $x \in [0,1]^{\mathcal{N}}$.*

In fact, in this thesis, our sole use of the Lovász extension is for constructing lower bounds on the multilinear extension via this theorem. At this point, we need some additional notation dealing with vectors of the cube $[0,1]^{\mathcal{N}}$ and the multilinear extension. For two vectors $x, y \in [0,1]^{\mathcal{N}}$, we use $x \vee y$ and $x \wedge y$ to denote the coordinate-wise maximum and minimum, respectively, of x and y (formally, $(x \vee y)_u = \max\{x_u, y_u\}$ and $(x \wedge y)_u = \min\{x_u, y_u\}$). We also make use of the notation $\partial_u F(x) = F(x \vee \mathbf{1}_u) - F(x \wedge \mathbf{1}_{\bar{u}})$, where $\mathbf{1}_u$ and $\mathbf{1}_{\bar{u}}$ are the characteristic vectors of the sets $\{u\}$ and $\mathcal{N} - u$, respectively. The multilinear nature of F yields the following useful observation, relating the terms just defined to each other.

Observation 2.3.6. *Let $F(x)$ be the multilinear extension of a submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}$. Then, for every $u \in \mathcal{N}$,*

$$\partial_u F(x) = \frac{F(x \vee \mathbf{1}_u) - F(x)}{1 - x_u} = \frac{F(x) - F(x \wedge \mathbf{1}_{\bar{u}})}{x_u} \ .$$

Since its introduction by [15], the multilinear extension has played a central part in the theory of submodular maximization (see [16, 61, 58, 81] for several examples). The usefulness of the multilinear extension stems from the many useful properties it has, among them constant first derivatives, non-increasing second derivatives and, for monotone functions, positive derivatives in any positive direction. The following lemma formalizes an additional basic property of the multilinear extension: within a small range, the multilinear extension can be treated as linear. The proof of this lemma is long and technical, and is, therefore, deferred to Appendix 2.A.

Lemma 2.3.7. *Consider two vectors $x, x' \in [0,1]^{\mathcal{N}}$ such that $|x_u - x'_u| \leq \delta$ for every $u \in \mathcal{N}$, and let F be the multilinear extension of a non-negative submodular function f. Then, $F(x') - F(x) \geq \sum_{u \in \mathcal{N}} (x'_u - x_u) \cdot \partial_u F(x) - O(n^3 \delta^2) \cdot \max_{u \in \mathcal{N}} f(\{u\})$.*

For many submodular functions we can evaluate F efficiently. However, we do not know how to do that for a general submodular function f, given only oracle access to f. Still, the following Chernoff like theorem allows us to approximate the value of F arbitrarily well using sampling.

Theorem 2.3.8 (Theorem A.1.16 in [4])**.** *Let X_i $(1 \leq i \leq k)$ be mutually independent with all $\mathbb{E}[X_i] = 0$ and all $|X_i| \leq 1$. Set $S = \sum_{i=1}^{k} X_i$, then $\Pr[|S| > a] \leq 2e^{-a^2 \cdot (2k)^{-1}}$.*

To simplify the exposition of our algorithms, we assume they have an oracle access to F. If this is not the case, every oracle access must be replaced with an approximation via sampling. This makes the approximation ratio deteriorate by a low order term only (see, *e.g.*, [16] for details).

The use of the multilinear extension in approximation algorithms follows the approach known from linear problems: find a good fractional point and round it. For linear problems the two steps are often separated:

- A good fractional point is found via a general solver such as an LP solver.
- The rounding is largely done via a tailored problem specific method.

The situation for submdoular problems is similar. A good fractional point can often be found via the Continuous Greedy algorithm discussed in Section 3. Rounding is performed either via one of a few general methods, or via a tailored problem specific method. Section 6 discusses contention resolution schemes, which is a very general rounding method. Another example for a rounding method is pipage rounding [2, 16], used for problems with a single matroid constraint. To understand the capabilities of pipage rounding, we need a few additional terms.

Matroid Polytopes are an extension of matroids to the cube $[0,1]^{\mathcal{N}}$. Given a matroid M, its matroid polytope is the convex-hall of all the independent sets of M, and is denoted by $\mathcal{P}(M)$. Similarly, the convex-hall of all the bases of the matroid is called the *Bases Polytope* of M, and is denoted by $\mathcal{B}(M)$. Both polytopes have separation oracles, so it is possible to optimize linear functions over them [59]. Given a matroid M and a point $x \in \mathcal{P}(M)$ within its matroid polytope, pipage rounding returns a random set S with the following properties:

- The set S is always independent in M.
- If x is in the bases polytope $\mathcal{B}(M)$, then S is also a base of M.
- Every element $u \in \mathcal{N}$ appears in S with probability x_u.
- For every submodular function f, the expected value of $f(S)$ is at least $F(x)$.

Consider, *e.g.*, the problem of `maximizing a non-negative monotone submodular function subject to a matroid constraint` defined above. An optimal algorithm for this problem works as following [16].

1. The input is a matroid M and a non-negative, monotone and submodular function f.
2. Use the continuous greedy algorithm to find a fractional solution $x \in \mathcal{P}(M)$ which is a $(1 - e^{-1} - o(1))$-approximation for this problem.
3. Use pipage rounding to round x into a set S with $\mathbb{E}[f(S)] = F(x)$.
4. Output the set S.

The above algorithm is a randomized $(1 - e^{-1} - o(1))$-approximation algorithm for the above problem, and it can be made deterministic via an appropriate derandomization.

2.A Proof of Lemma 2.3.7

Let us denote the term $\max_{u\in\mathcal{N}} f(\{u\})$ by w, and the term $\max_{S\subseteq\mathcal{N}}$ by W. From the submodularity of f, we get the following observation.

Observation 2.A.1. *For every set $S \subseteq \mathcal{N}$, $f(S) \leq n \cdot w$. Hence, $W \leq n \cdot w$.*

Let A be the set of elements u with $x'_u > x_u$, and let B be the set of elements with $x'_u < x_u$. Recall that $\mathtt{R}(x)$ is a random set containing every element $u \in \mathcal{N}$ with probability x_u. For the sake of the proof, we assume $\mathtt{R}(x')$ is formed from $\mathtt{R}(x)$ using the following process. Every element of $A \setminus \mathtt{R}(x)$ is added to a set D with probability of $1 - (1 - x'_u)/(1 - x_u)$, and every element of $B \cap \mathtt{R}(x)$ is added to D with probability $1 - x'_u/x_u$. Then, $\mathtt{R}(x')$ is chosen as $\mathtt{R}(x) \oplus D$. Observe that every element $u \in \mathcal{N}$ gets into D with probability $|x_u - x'_u| \leq \delta$, independently. We now bound the value of $F(x') - F(x) = \mathbb{E}[f(\mathtt{R}(x')) - f(\mathtt{R}(x))]$, given various constraints on D.

Lemma 2.A.2. $\sum_{u\in\mathcal{N}} \Pr[D = \{u\}]\cdot\mathbb{E}[F(x')-F(x)|D = \{u\}] \geq \sum_{u\in\mathcal{N}}(x'_u-x_u)\cdot\partial_u F(x)-O(n^3\delta^2)\cdot w$.

Proof. Let $\mathcal{N}^+$ be the set of elements from $\mathcal{N}$ which have $(x'_u - x_u)\cdot\partial_u F(x) \geq 0$. Observe that for every $u \in \mathcal{N}^+$:

$$\begin{aligned}
(x'_u - x_u)\cdot\partial_u F(z) &= |x'_u - x_u|\cdot\mathbb{E}[F(x') - F(x)|D = \{u\}] \\
&= \frac{\Pr[D = \{u\}]\cdot\mathbb{E}[F(x') - F(x)|D = \{u\}]}{\prod_{u'\in\mathcal{N}-u}\left(1 - |x'_{u'} - x_{u'}|\right)} \\
&\leq \frac{\Pr[D = \{u\}]\cdot\mathbb{E}[F(x') - F(x)|D = \{u\}]}{\prod_{u'\in\mathcal{N}-u}(1-\delta)} \\
&= (1-\delta)^{1-|\mathcal{N}|}\cdot\Pr[D = \{u\}]\cdot\mathbb{E}[F(x') - F(x)|D = \{u\}] \\
&< (1-\delta)^{-n}\cdot\Pr[D = \{u\}]\cdot\mathbb{E}[F(x') - F(x)|D = \{u\}] \ .
\end{aligned}$$

And the term, $(1-\delta)^n$ can be lower bounded as following.

$$(1-\delta)^n = (1-\delta)^{n\delta/\delta} \geq [e^{-1}(1-\delta)]^{n\delta} \geq e^{-2n\delta} \geq 1 - 2n\delta \ .$$

Also, for every $u \in \mathcal{N} \setminus \mathcal{N}^+$:

$$\begin{aligned}
(x'_u - x_u)\cdot\partial_u F(x) &= |x'_u - x_u|\cdot\mathbb{E}[F(x') - F(x)|D = \{u\}] \\
&= \frac{\Pr[D = \{u\}]\cdot\mathbb{E}[F(x') - F(x)|D = \{u\}]}{\prod_{u'\in\mathcal{N}-u}\left(1 - |x'_{u'} - x_{u'}|\right)} \\
&\leq \Pr[D = \{u\}]\cdot\mathbb{E}[F(x') - F(x)|D = \{u\}] \ .
\end{aligned}$$

Combining everything, and recalling that $f(S) \leq W \leq n \cdot w$ for every $S \subseteq \mathcal{N}$, we get:

$$\begin{aligned}
\sum_{u\in\mathcal{N}}(x'_u - x_u)\cdot\partial_u F(x) - 2n^3\delta^2 w &\leq \sum_{u\in\mathcal{N}}(x'_u - x_u)\cdot\partial_u F(x) - \delta n(2n\delta)W \\
&\leq \sum_{u\in\mathcal{N}}[(x'_u - x_u)\cdot\partial_u F(x) - 2n\delta\cdot|(x'_u - x_u)\cdot\partial_u F(x)|] \\
&\leq \sum_{u\in\mathcal{N}}\Pr[D = \{u\}]\cdot\mathbb{E}[F(x') - F(x)|D = \{u\}] \ . \qquad \square
\end{aligned}$$

Lemma 2.A.3. $\mathbb{E}[F(x') - F(x) \mid D = \varnothing] = 0$

Proof. $D = \varnothing$ implies $R(x') = R(x)$. □

Lemma 2.A.4. $\Pr[|D| \geq 2] \cdot \mathbb{E}[F(x') - F(x) \mid |D| \geq 2] \geq -O(n^3\delta^2) \cdot w$.

Proof. Let us bound the probability that $|D| \geq 2$. Since every element gets into D with probability at most δ:

$$\begin{aligned}\Pr[|D| \geq 2] &\leq 1 - (1-\delta)^n - n\delta \cdot (1-\delta)^{n-1} \leq 1 - (1+n\delta) \cdot (1-\delta)^n \\ &\leq 1 - (1+n\delta) \cdot e^{-n\delta} \cdot (1-n\delta^2) \leq 1 - (1+n\delta)(1-n\delta)(1-n\delta^2) \leq 2n^2\delta^2 \ .\end{aligned}$$

Therefore,

$$\Pr[|D| \geq 2] \cdot \mathbb{E}[F(x') - F(x) \mid |D| \geq 2] \geq \Pr[|D| \geq 2] \cdot (-W) \geq -2n^3\delta^2 \cdot w \ . \qquad \square$$

Lemma 2.3.7 now follows immediately from the above lemmata and the law of total probability.

Chapter 3

Continuous Greedy Algorithms

Consider the basic problem of maximizing a non-negative submodular function $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}^+$ over a ground set $\mathcal{N}$ under the constraint that the solution must belong to a set system $(\mathcal{N}, \mathcal{I})$.[1] This basic (constrained) submodular maximization problem generalizes, *e.g.*, the two well studied problems `Max-Cut` and `Max-`k`-Cover` [38, 45, 48, 51, 53, 55, 70, 69, 78].

The techniques used to compute approximate solutions to various (constrained) submodular maximization problems can be partitioned into two main approaches. The first approach is combinatorial in nature, and is mostly based on local search techniques and greedy rules. This approach has been used as early as the late 70's for maximizing monotone submodular functions under the constraint that the solution should be an independent set of one of several specific matroids [22, 36, 46, 47, 50, 56, 71, 72]. Lately, this approach has been extended to include both non-monotone submodular objective functions [30, 37, 34, 81] and additional constraint sets $\mathcal{I}$ [62] (*e.g.*, independent sets of matroids intersection). Though for some problems this approach yields the current state of the art solutions [62], or even tight results [77], these solutions are usually tailored for the specific structure of the problem at hand, making extensions quite difficult.

The second approach for approximating (constrained) submodular maximization problems overcomes the above obstacle. This approach resembles a common paradigm for designing approximation algorithms and is composed of two steps. In the first step, a fractional solution is found for a relaxation of the problem. In the second step, the fractional solution is rounded to obtain an integral one while incurring only a small loss in the objective. This approach has been used to obtain improved approximations to various problems [16, 18, 19, 20, 58, 61]. Most notable of these results is an *asymptotically tight* approximation for maximizing a monotone submodular function given a single matroid constraint [16, 71, 72]. Two issues arise when using this approach. First, since the objective function is not linear, it is not clear how to formulate a relaxation which can be solved or even approximated efficiently. Second, given a fractional solution, one needs a rounding procedure which outputs an integral solution without losing too much in the objective function.

Let us elaborate on the first issue, namely how to find good fractional solutions to (constrained) submodular maximization problems. The standard relaxation for such a problem has a variable for every element of the ground set $\mathcal{N}$ taking values from the range $[0, 1]$. As with linear programming relaxations, the collection $\mathcal{I}$ is replaced by a set of linear inequality constraints on the variables which define a down-monotone polytope[2] $\mathcal{P}$. Unlike

[1] Note that many natural collections of subsets form set systems, *e.g.*, collections induced by matroid and knapsack contraints.

[2] A polytope $\mathcal{P} \subseteq [0, 1]^{\mathcal{N}}$ is *down-monotone* if $x \in \mathcal{P}$ and $0 \leq y \leq x$ imply $y \in \mathcal{P}$.

the linear case, the formulation of an objective function for the relaxation is not obvious. A good objective function is a continuous extension of the given integral objective f which allows for efficient computation of a good fractional solution. The extension commonly used to overcome this difficulty, in the context of (constrained) submodular maximization problems, is the multilinear extension F. Such relaxations are very common, since first introduced by [15] (see [16, 61, 58, 81] for several additional examples).

Even though the objective function defined by the multilinear extension is neither convex nor concave, it is still possible to efficiently compute an approximate feasible fractional solution for the relaxation, assuming its feasibility polytope $\mathcal{P}$ is down monotone and solvable[3]. The first method proposed for computing such a solution is the *continuous greedy* algorithm [16]. It is simple and quick, and its analysis is rather short and intuitive. However, it is only known to work for the multilinear extensions of *monotone* submodular functions f. For non-monotone functions f and specific polytopes, other methods are known for solving the multilinear extension, *e.g.*, for a constant number of knapsack constraints [61] and for a single matroid [37, 81]. These methods use extensions of the local search approach, as opposed to the simple continuous greedy method, making the analysis quite involved. Recently, three algorithms for the non-monotone case and general down-monotone solvable polytopes were suggested by [20]. Similarly to [61, 37], these three algorithms are also based on extensions of the local search approach. The best of the three (with respect to its approximation guarantee) uses a simulated annealing technique [37]. Therefore, these algorithms, and especially the best of the three, have quite a complex analysis.

In this section we present the measured continuous greedy algorithm which finds approximate fractional solutions for both the non-monotone and monotone cases, and improves on the approximation ratio for many applications. For general non-monotone submodular objective functions, our algorithm achieves an approximation ratio of about $1/e$. For monotone submodular objective functions, our algorithm achieves an approximation ratio that depends on the density of the polytope defined by the problem at hand, which is always at least as good as the $1-1/e$ approximation guaranteed by the "traditional" continuous greedy. Some notable immediate applications are an improved $1/e$-approximation for maximizing a non-monotone submodular function subject to a matroid or $O(1)$-knapsack constraints, and information-theoretic tight approximations for `Submodular Max-SAT` and `Submodular Welfare` with k players, for *any* number of players k.

3.1 Continuous Greedy

Let us present the continuous greedy algorithm of [16], and analyze it. The measured continuous greedy is presented in Section 3.2. The continuous greedy algorithm has a parameter T called *stopping time*. The stopping time controls a tradeoff between two important properties of the fractional solution found by the algorithm. The first property is the value of the solution: a larger stopping time implies a better fractional solution. The second property is how much slack does the fractional solution has: a smaller stopping time implies more slack (refer to Section 6.1 for uses of the second property).[4]

Remark: The way δ is defined in Algorithm 1 implies that δ^{-1} has two properties: it

[3] A polytope $\mathcal{P}$ is *solvable* if linear functions can be maximized over it in polynomial time. Using the ellipsoid algorithm, one can prove $\mathcal{P}$ is solvable by describing a polynomial-time algorithm that given x determines whether $x \in \mathcal{P}$.

[4] The concept of stopping time did not exist in the original presentation of [16]. It was only introduced with the measured continuous greedy in [34].

Algorithm 1: Continuous Greedy$(f, \mathcal{P}, T)$

```
// Initialization
Set: δ ← T(⌈n^5 T⌉)^{-1}.
Initialize: t ← 0, y(0) ← 1_∅.
// Main loop
while t < T do
    foreach u ∈ 𝒩 do
        w_u(t) ← F(y(t) ∨ 1_u) − F(y(t)).
    Let I(t) ∈ 𝒫 be a vector maximizing I(t) · w(t).
    foreach u ∈ 𝒩 do
        y_u(t + δ) ← y_u(t) + δ I_u(t).
    t ← t + δ.
Return y(T).
```

is at least n^5, and it is dividable by T^{-1}. The last property guarantees that after $T\delta^{-1}$ iterations, t will be exactly T.

Theorem 3.1.1. *For any monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, down-monotone solvable polytope $\mathcal{P} \subseteq [0,1]^{\mathcal{N}}$ and stopping time $T \in [0,1]$, the* continuous greedy *algorithm finds a point $x \in [0,1]^{\mathcal{N}}$ such that $F(x) \geq [1 - e^{-T} - o(1)] \cdot f(OPT)$ and $x/T \in \mathcal{P}$.*

Notice that for $T = 1$ the algorithm outputs a point x such that $x \in \mathcal{P}$ and $F(x) \geq [1 - e^{-1} - o(1)] \cdot f(OPT)$. In the rest of this section we prove Theorem 3.1.1.

Lemma 3.1.2. *For every $T \geq 0$, the continuous greedy algorithm produces a solution x such that $x/T \in \mathcal{P}$.*

Proof. Notice that $x = \delta \cdot \sum_{i=0}^{T/\delta - 1} I(i \cdot \delta)$. x/δ is the sum of T/δ points in $\mathcal{P}$, and therefore, $x/T = (x/\delta)/(T/\delta) \in \mathcal{P}$. □

The following lemma together with Lemma 2.3.7 gives a lower bound on the improvement achieved by the algorithm in each iteration. This lower bound is stated explicitly in Corollary 3.1.4.

Lemma 3.1.3. *For every time $0 \leq t < T$, $\sum_{u \in \mathcal{N}} I_u(t) \cdot \partial_u F(y(t)) \geq f(OPT) - F(y(t))$.*

Proof. Recall that $\mathtt{R}(x)$ is a random set containing every element $u \in \mathcal{N}$ with probability x_u, and that $F(x) = \mathbb{E}[\mathtt{R}(x)]$. Let us calculate the weight of OPT according to weight function $w(t)$.

$$\begin{aligned} w(t) \cdot \mathbf{1}_{OPT} &= \sum_{u \in OPT} w_u(t) = \sum_{u \in OPT} [F(y(t) \vee \mathbf{1}_u) - F(y(t))] \\ &= \mathbb{E}\left[\sum_{u \in OPT} f(\mathtt{R}(y(t)) + u) - f(\mathtt{R}(y(t)))\right] \\ &\geq \mathbb{E}\left[f(\mathtt{R}(y(t)) \cup OPT) - f(\mathtt{R}(y(t)))\right] = F(y(t) \vee \mathbf{1}_{OPT}) - F(y(t)) \ , \end{aligned}$$

where the inequality follows from submodularity. Since $\mathbf{1}_{OPT} \in \mathcal{P}$, we get:

$$w(t) \cdot I(t) \geq F(y(t) \vee \mathbf{1}_{OPT}) - F(y(t)) \ .$$

Hence,

$$\begin{aligned}\sum_{u\in\mathcal{N}} I_u(t)\cdot\partial_u F(y(t)) &= \sum_{e\in\mathcal{N}} I_u(t)\cdot[F(y(t)\vee 1_u)-F(y(t)\wedge 1_{\bar{u}})]\\ &\geq \sum_{e\in\mathcal{N}} I_u(t)\cdot[F(y(t)\vee 1_u)-F(y(t))] = I(t)\cdot w(t)\\ &\geq F(y(t)\vee 1_{OPT})-F(y(t)) \geq f(OPT)-F(y(t))\ .\end{aligned}$$ □

Corollary 3.1.4. *For every time $0 \leq t < T$, $F(y(T+\delta)) - F(y(T)) \geq \delta\cdot[f(OPT) - F(y(t))] - O(n^3\delta^2)\cdot f(OPT)$.*

At this point we have a lower bound on the improvement achieved in each iteration in terms of $f(OPT)$ and $F(y(t))$. In order to complete the analysis of the algorithm, we need to derive from it a bound on the value of $F(y(t))$ for every time t. Let $g(t)$ be defined as following. $g(0) = 0$ and $g(t+\delta) = g(t) + \delta[f(OPT) - g(t)]$. The next lemma shows that a lower bound on $g(t)$ also gives a lower bound on $F(y(t))$

Lemma 3.1.5. *For every $0 \leq t \leq T$, $g(t) \leq F(y(t)) + O(n^3\delta)\cdot tf(OPT)$.*

Proof. Let c be the constant hiding behind the big O notation in Corollary 3.1.4. We prove by induction on t that $g(t) \leq F(y(t)) + cn^3\delta tf(OPT)$. For $t = 0$, $g(0) = 0 \leq F(y(0))$. Assume now that the claim holds for some t, and let us prove it for $t+\delta$. Using Corollary 3.1.4, we get:

$$\begin{aligned}g(t+\delta) &= g(t) + \delta[f(OPT) - g(t)] = (1-\delta)g(t) + \delta f(OPT)\\ &\leq (1-\delta)[F(y(t)) + cn^3\delta tf(OPT)] + \delta f(OPT)\\ &= F(y(t)) + \delta[f(OPT) - F(y(t))] + c(1-\delta)n^3\delta tf(OPT)\\ &\leq F(y(t+\delta)) + cn^3\delta^2 f(OPT) + c(1-\delta)n^3\delta tf(OPT)\\ &\leq F(y(t+\delta)) + cn^3\delta(t+\delta)f(OPT)\ .\end{aligned}$$ □

The function g is given by a recursive formula, thus, evaluating it is not immediate. Instead, we show that the function $h(t) = (1-e^{-t})\cdot f(OPT)$ lower bounds g for every value of t.

Lemma 3.1.6. *For every time $0 \leq t \leq T$, $g(t) \geq h(t)$.*

Proof. The proof is by induction on t. For $t = 0$, $g(0) = 0 = (1-e^{-0})\cdot f(OPT) = h(0)$. Assume now that the lemma holds for some t, and let us prove it holds for $t+\delta$.

$$\begin{aligned}h(t+\delta) &= h(t) + \int_t^{t+\delta} h'(\tau)d\tau = h(t) + f(OPT)\cdot\int_t^{t+\delta} e^{-\tau}d\tau \leq h(t) + f(OPT)\cdot\delta e^{-t}\\ &= (1-\delta)h(t) + \delta\cdot f(OPT) \leq (1-\delta)g(t) + \delta\cdot f(OPT) = g(t+\delta)\ .\end{aligned}$$ □

We can now use the last result to lower bound the quality of the algorithm's output.

Corollary 3.1.7. $F(y(T)) \geq [1 - e^{-T} - o(1)]\cdot f(OPT)$.

Proof. By Lemmata 3.1.5 and 3.1.6, $F(y(T)) \geq g(T) - O(n^3\delta)\cdot T\cdot f(OPT) \geq h(T) - O(n^3\delta)\cdot f(OPT) = [1-e^{-T}-O(n^3\delta)]\cdot f(OPT)$. Recall that $\delta \leq n^{-5}$, hence, $O(n^3\delta) = o(1)$, and the proof is complete. □

Theorem 3.1.1 now follows immediately from Lemma 3.1.2 and Corollary 3.1.7.

3.2 Measured Continuous Greedy

The measured continues greedy is based on a simple but crucially useful insight on which we now elaborate. The continuous greedy algorithm of [16] (presented above) starts with an empty solution and at each step moves by a small δ in the direction of a feasible point $x \in \mathcal{P}$. Let y be the current position of the algorithm. Then x is chosen greedily (hence the name "continuous greedy") by solving $x = \text{argmax}\{w(y) \cdot x \mid x \in \mathcal{P}\}$ where the weight vector $w(y) \in \mathbb{R}^{\mathcal{N}}$ is defined by $w(y)_u \triangleq F(y \vee 1_u) - F(y)$, for every $u \in \mathcal{N}$. Thus, x is chosen according to the *residual increase* of each element u, *i.e.*, $F(y \vee 1_u) - F(y)$. However, one would intuitively expect that the step should be chosen according to the *gradient* of $F(y)$. Observe that the residual increase is equal to $\partial_u F(y) \cdot (1 - y_u)$. The measured continues greedy compensates for the difference between the residual increase of elements at point y, and $\partial_u F(y)$, by *distorting* the direction x. Each coordinate of x_u is decreased by a multiplicative factor of $1 - y_u$. Hence, both the weight w_u and the step are multiplied by the same factor. The name of the algorithm is derived from this decrease.

The following two theorems quantify the guaranteed performance of the measured continuous greedy algorithm for non-monotone and monotone submodular functions. We denote by OPT the optimal integral solution. Note that the first bullet of Theorem 3.2.2, $x/T \in \mathcal{P}$, repeats, in fact, the guarantee of the continuous greedy algorithm. However, the second bullet of this theorem enables us to obtain improved approximation guarantees for several well studied problems. This property states that in some settings one can use stopping times larger than 1. The maximal stopping time that can be used depends on the *density* of the underlying polytope. Consider a down-monotone polytope $\mathcal{P} \subseteq [0,1]^{\mathcal{N}}$ defined by positivity constraints ($x \geq 0$) and additional m inequality constraints. Let $\sum_{u \in \mathcal{N}} a_{i,u} x_u \leq b_i$ denote the i^{th} inequality constraint. The density of $\mathcal{P}$ is defined by: $d(\mathcal{P}) = \min_{1 \leq i \leq m} \frac{b_i}{\sum_{u \in \mathcal{N}} a_{i,u}}$.[5] Since $\mathcal{P}$ is a down monotone polytope within the hypercube $[0,1]^{\mathcal{N}}$, one can assume all coefficients $a_{i,u}$ and b_i are non-negative, and $0 < d(\mathcal{P}) \leq 1$. See Appendix 3.A for details.

Theorem 3.2.1. *For any given non-negative submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, down-monotone solvable polytope $\mathcal{P} \subseteq [0,1]^{\mathcal{N}}$ and stopping time $T \geq 0$, the* measured continuous greedy *algorithm finds a point $x \in [0,1]^{\mathcal{N}}$ such that $F(x) \geq [Te^{-T} - o(1)] \cdot f(OPT)$ and $x/T \in \mathcal{P}$.*

Theorem 3.2.2. *For any normalized monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, down-monotone solvable polytope $\mathcal{P} \subseteq [0,1]^{\mathcal{N}}$ and stopping time $T \geq 0$, the* measured continuous greedy *algorithm finds a point $x \in [0,1]^{\mathcal{N}}$ such that $F(x) \geq [1 - e^{-T} - o(1)] \cdot f(OPT)$. Additionally,*

- *$x/T \in \mathcal{P}$.*
- *Let $T_{\mathcal{P}} = -\ln(1 - d(\mathcal{P}) + n\delta)/d(\mathcal{P})$. Then, $T \leq T_{\mathcal{P}}$ implies $x \in \mathcal{P}$.*

For monotone submodular objectives, the dependance of the approximation ratio on the stopping time T is identical for both the measured continues greedy and the continuous greedy algorithm of [16]. This is somewhat counter intuitive, since the measured continuous greedy makes a "smaller" step in each iteration (recall that the movement in direction u is reduced by a multiplicative factor of $(1 - y_u)$). This seems to suggest that the traditional continuous greedy algorithm is a bit wasteful. The smaller steps of the measured algorithm prevent this waste, keep its fractional solution within the polytope for a longer period of time, and thus, allow the use of larger stopping times in some settings.

[5] Notice that the density resembles the width parameter used by [6].

Theorem 3.2.2 gives an approximation ratio of $1 - e^{-T_{\mathcal{P}}} \approx 1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})}$. In some cases one can get a cleaner approximation ratio of exactly $1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})}$ by guessing the most valuable single element of OPT (the technique of guessing the most valuable single element of OPT is not new, and can be found, *e.g.*, in [16]). The following theorem examplifies that. A *binary* polytope $\mathcal{P}$ is a polytope defined by constraints with only $\{0, 1\}$ coefficients.

Theorem 3.2.3. *Given a binary down-monotone solvable polytope $\mathcal{P}$ with a bounded $T_{\mathcal{P}}$ and a normalized monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, there is a polynomial time algorithm outputting a point $x \in \mathcal{P}$ with $F(x) \geq [1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})}] \cdot f(OPT)$.*

The measured continuous greedy algorithm is depicted as Algorithm 2. Notice its similarity to the traditional continuous greedy (Algorithm 1). The sole change in Algorithm 2 is the distortion of the direction y, which appears in line 8 of the algorithm as the multiplication of $I_e(t)$ with $1 - y_e(t)$.

Algorithm 2: Measured Continuous Greedy$(f, \mathcal{P}, T)$

```
   // Initialization
1  Set: n ← |N|, δ ← T(⌈n^5 T⌉)^{-1}.
2  Initialize: t ← 0, y(0) ← 1_∅.
   // Main loop
3  while t < T do
4      foreach e ∈ N do
5          w_e(t) ← F(y(t) ∨ 1_e) − F(y(t)).
6      Let I(t) ∈ P be a vector maximizing I(t) · w(t).
7      foreach e ∈ N do
8          y_e(t + δ) ← y_e(t) + δ I_e(t) · (1 − y_e(t)).
9      t ← t + δ.
10 Return y(T).
```

3.2.1 Analysis for Non-Monotone f

In this section we analyze the measured continuous greedy algorithm for general non-negative submodular functions, and prove Theorem 3.2.1. We first prove that the algorithm always remains within the cube $[0, 1]^{\mathcal{N}}$, regardless of the stopping time. Without this observation, the algorithm is not well-defined for $T > 1$.

Observation 3.2.4. *For every value of t, $y(t) \in [0, 1]^{\mathcal{N}}$.*

Proof. We prove the observation by induction on t. Clearly the observation holds for $y(0) = 1_{\varnothing}$. Assume the observation holds for some t, then, for every $u \in \mathcal{N}$, $y_u(t + \delta) \leq y_u(t) + I(t) \cdot (1 - y_u(t)) \leq 1$. □

Next, we prove a counterpart of Lemma 3.1.2.

Lemma 3.2.5. *For every $T \geq 0$, the measured continuous greedy algorithm produces a solution x such that $x/T \in \mathcal{P}$.*

Proof. Notice that x is coordinate-wise upper bounded by $x' = \delta \cdot \sum_{i=0}^{T/\delta - 1} I(i \cdot \delta)$. Since $\mathcal{P}$ is a down-monotone polytope, it is enough to show that $x'/T \in \mathcal{P}$. x'/δ is the sum of T/δ points in $\mathcal{P}$, and therefore, $x'/T = (x'/\delta)/(T/\delta) \in \mathcal{P}$. □

The following lemma gives together with Lemma 2.3.7 a lower bound on the improvement achieved by the algorithm in each iteration. This lower bound is stated explicitly in Corollary 3.2.7.

Lemma 3.2.6. *For every time* $0 \leq t < T$, $\sum_{u \in \mathcal{N}} (1 - y_u(t)) \cdot I_u(t) \cdot \partial_u F(y(t)) \geq F(y(t) \vee 1_{OPT}) - F(y(t))$.

Proof. Recall that $\mathtt{R}(x)$ is a random set containing every element $u \in \mathcal{N}$ with probability x_u, and that $F(x) = \mathbb{E}[\mathtt{R}(x)]$. Let us calculate the weight of OPT according to weight function $w(t)$.

$$\begin{aligned} w(t) \cdot 1_{OPT} &= \sum_{u \in OPT} w_u(t) = \sum_{u \in OPT} [F(y(t) \vee 1_u) - F(y(t))] \\ &= \mathbb{E}\left[\sum_{u \in OPT} f(\mathtt{R}(y(t)) + u) - f(\mathtt{R}(y(t))) \right] \\ &\geq \mathbb{E}\left[f(\mathtt{R}(y(t)) \cup OPT) - f(\mathtt{R}(y(t))) \right] = F(y(t) \vee 1_{OPT}) - F(y(t)) \ . \end{aligned}$$

Where the inequality follows from submodularity. Since $1_{OPT} \in \mathcal{P}$, we get:

$$w(t) \cdot I(t) \geq F(y(t) \vee 1_{OPT}) - F(y(t)) \ .$$

Hence,

$$\begin{aligned} \sum_{u \in \mathcal{N}} (1 - y_u(t)) \cdot I_u(t) \cdot \partial_u F(y(t)) &= \sum_{u \in \mathcal{N}} (1 - y_u(t)) \cdot I_u(t) \cdot [F(y(t) \vee 1_u) - F(y(t) \wedge 1_{\bar{u}})] \\ &= \sum_{u \in \mathcal{N}} I_u(t) \cdot [F(y(t) \vee 1_u) - F(y(t))] = I(t) \cdot w(t) \\ &\geq F(y(t) \vee 1_{OPT}) - F(y(t)) \ . \end{aligned}$$

□

Corollary 3.2.7. *For every time* $0 \leq t < T$, $F(y(T+\delta)) - F(y(T)) \geq \delta \cdot [F(y(t) \vee 1_{OPT}) - F(y(t))] - O(n^3\delta^2) \cdot f(OPT)$.

The lower bound given by the last corollary is in terms of $F(y(t) \vee 1_{OPT})$. To make this lower bound useful, we need to lower bound the term $F(y(t) \vee 1_{OPT})$. This is done by the following two lemmata and corollary.

Lemma 3.2.8. *Consider a vector* $x \in [0,1]^{\mathcal{N}}$. *Assuming* $x_u \leq a$ *for every* $u \in \mathcal{N}$, *then for every set* $S \subseteq \mathcal{N}$, $F(x \vee 1_S) \geq (1 - a) f(S)$.

Proof. Notice that if $\lambda > a$, then $T_\lambda(x) = \varnothing$. By Theorem 2.3.5, we have:

$$\begin{aligned} F(x \vee 1_S) \geq \hat{f}(x \vee 1_S) &= \int_0^1 f(T_\lambda(x \vee 1_S)) d\lambda = \int_0^1 f(T_\lambda(x) \cup S) d\lambda \\ &\geq \int_a^1 f(T_\lambda(x) \cup S) d\lambda = \int_a^1 f(S) d\lambda = (1 - a) \cdot f(S) \ . \end{aligned}$$

□

Lemma 3.2.9. *For every time* $0 \leq t \leq T$ *and element* $u \in \mathcal{N}$, $y_u(t) \leq 1 - (1 - \delta)^{t/\delta} \leq 1 - e^{-t} + O(\delta)$.

Proof. We prove the first inequality by induction on t. For $t = 0$, the inequality holds because $y_u(0) = 0 = 1 - (1-\delta)^{0/\delta}$. Assume the inequality holds for some t, and let us prove it for $t + \delta$.

$$\begin{aligned} y_u(t+\delta) &= y_u(t) + \delta I_u(t)(1 - y_u(t)) = y_u(t)(1 - \delta I_u(t)) + \delta I_u(t) \\ &\leq (1 - (1-\delta)^{t/\delta})(1 - \delta I_u(t)) + \delta I_u(t) = 1 - (1-\delta)^{t/\delta} + \delta I_u(t)(1-\delta)^{t/\delta} \\ &\leq 1 - (1-\delta)^{t/\delta} + \delta(1-\delta)^{t/\delta} = 1 - (1-\delta)^{(t+\delta)/\delta} \ . \end{aligned}$$

We complete the proof by deriving the second inequality: $1-(1-\delta)^{t/\delta} \leq 1-[e^{-1}(1-\delta)]^t = 1 - e^{-t}(1-\delta)^t \leq 1 - e^{-t}(1 - T\delta) = 1 - e^{-t} + O(\delta)$, where the last inequality holds since $t \in [0, T]$. □

Corollary 3.2.10. *For every time* $0 \leq t < T$, $F(y(T+\delta)) - F(y(T)) \geq \delta \cdot [(e^{-t} - O(\delta)) \cdot f(OPT) - F(y(t))] - O(n^3\delta^2) \cdot f(OPT) = \delta \cdot [e^{-t} \cdot f(OPT) - F(y(t))] - O(n^3\delta^2) \cdot f(OPT)$.

Proof. By Lemma 3.2.9, every coordinate in $y(t)$ is at most $1 - e^{-t} + O(\delta)$. Therefore, by Lemma 3.2.8, $F(y(t) \vee 1_{OPT})] \geq [e^{-t} - O(\delta)] \cdot f(OPT)$. Plugging this into Corollary 3.2.7 completes the proof. □

At this point we have a lower bound on the improvement achieved in each iteration in terms of t, $f(OPT)$ and $F(y(t))$. In order to complete the analysis of the algorithm, we need to derive from it a bound on the value of $F(y(t))$ for every time t. Let $g(t)$ be defined as following. $g(0) = 0$ and $g(t+\delta) = g(t) + \delta[e^{-t}f(OPT) - g(t)]$. The next lemma shows that a lower bound on $g(t)$ also gives a lower bound on $F(y(t))$

Lemma 3.2.11. *For every* $0 \leq t \leq T$, $g(t) \leq F(y(t)) + O(n^3\delta) \cdot tf(OPT)$.

Proof. Let c be the constant hiding behind the big O notation in Corollary 3.2.10. We prove by induction on t that $g(t) \leq F(y(t)) + cn^3\delta t f(OPT)$. For $t = 0$, $g(0) = 0 \leq F(y(0))$. Assume now that the claim holds for some t, and let us prove it for $t + \delta$. Using Corollary 3.2.10, we get:

$$\begin{aligned} g(t+\delta) &= g(t) + \delta[e^{-t}f(OPT) - g(t)] = (1-\delta)g(t) + \delta e^{-t}f(OPT) \\ &\leq (1-\delta)[F(y(t)) + cn^3\delta t f(OPT)] + \delta e^{-t}f(OPT) \\ &= F(y(t)) + \delta[e^{-t}f(OPT) - F(y(t))] + c(1-\delta)n^3\delta t f(OPT) \\ &\leq F(y(t+\delta)) + cn^3\delta^2 f(OPT) + c(1-\delta)n^3\delta t f(OPT) \\ &\leq F(y(t+\delta)) + cn^3\delta(t+\delta)f(OPT) \ . \end{aligned}$$

□

The function g is given by a recursive formula, thus, evaluating it is not immediate. Instead, we show that the function $h(t) = te^{-t} \cdot f(OPT)$ lower bounds g within the range $[0, 1]$.

Lemma 3.2.12. *For every* $0 \leq t \leq T$, $g(t) \geq h(t)$.

Proof. The proof is by induction on t. For $t = 0$, $g(0) = 0 = 0 \cdot e^{-0} \cdot f(OPT) = h(0)$. Assume now that the lemma holds for some t, and let us prove it holds for $t + \delta$.

$$\begin{aligned} h(t+\delta) &= h(t) + \int_t^{t+\delta} h'(\tau)d\tau = h(t) + f(OPT) \cdot \int_t^{t+\delta} e^{-\tau}(1-\tau)d\tau \\ &\leq h(t) + f(OPT) \cdot \delta e^{-t}(1-t) = (1-\delta)h(t) + \delta e^{-t} \cdot f(OPT) \\ &\leq (1-\delta)g(t) + \delta e^{-t} \cdot f(OPT) = g(t) + \delta \cdot [e^{-t} \cdot f(OPT) - g(t)] = g(t+\delta) \ . \end{aligned}$$

□

We can now use the last result to lower bound the quality of the algorithm's output.

Corollary 3.2.13. $F(y(T)) \geq [Te^{-T} - o(1)] \cdot f(OPT)$.

Proof. By Lemmata 3.2.11 and 3.2.12, $F(y(T)) \geq g(T) - O(n^3\delta) \cdot Tf(OPT) \geq h(T) - O(n^3\delta) \cdot f(OPT) = [Te^{-T} - O(n^3\delta)] \cdot f(OPT)$. Recall that $\delta \leq n^{-5}$, hence, $O(n^3\delta) = o(1)$, and the proof is complete. □

Theorem 3.2.1 now follows immediately from Lemma 3.2.5 and Corollary 3.2.13.

3.2.2 Analysis for Monotone f

In this section we analyze the measured continuous greedy algorithm for normalized monotone submodular functions, and prove Theorems 3.2.2. Observe that all claims of Section 3.2.1 apply here too because a normalized monotone submodular function is a special case of a non-negative submodular function.

Theorem 3.2.2 has three parts. The first part we prove is $F(y(T)) \geq [(1 - e^T) - o(1)] \cdot f(OPT)$. By combining Lemma 3.2.6 with monotonicity, we get the following corollary.

Corollary 3.2.14. *For every time* $0 \leq t < T$, $F(y(T + \delta)) - F(y(T)) = \delta \cdot [f(OPT) - F(y(t))] - O(n^3\delta^2) \cdot f(OPT)$.

Corollary 3.2.14 is identical to Corollary 3.1.4 from the proof of Theorem 3.1.1. Notice that Theorem 3.1.1 also guarantees $F(y(T)) \geq [(1-e^T)-o(1)] \cdot f(OPT)$, and this guarantee follows solely from Corollary 3.1.4. Hence, we can follow the proof of Theorem 3.1.1, and prove that $F(y(T)) \geq [(1 - e^T) - o(1)] \cdot f(OPT)$ holds also for Algorithm 2. The proof of the first part of the theorem is now complete.

The second part of the theorem states that $x/T \in \mathcal{P}$, which was already proved by Lemma 3.2.5. Hence, we are left to prove the third part of the theorem, which states that if $\mathcal{P}$ is a packing constraint and $T \leq T_{\mathcal{P}}$, then $y(T) \in \mathcal{P}$. Consider some general constraint $\sum_{u \in \mathcal{N}} a_u x_u \leq b$ of $\mathcal{P}$. We assume $a_u > 0$ for some $u \in \mathcal{N}$, otherwise, the constraint holds always and can be ignored. Let $I_u^t = \delta \cdot \sum_{i=0}^{t/\delta - 1} I_u(\delta \cdot i)$, *i.e.*, I_u^t is the scaled sum of I_u over all times up to time t. The following two lemmata prove some properties of I_u^t.

Lemma 3.2.15. $\sum_{u \in \mathcal{N}} a_u \cdot I_u^T \leq Tb$.

Proof. For every time $t \in [0, T)$, $I(t)$ is a feasible solution, and therefore, $\sum_{u \in \mathcal{N}} a_i \cdot I_u(t) \leq b$. Summing over all times in this range, we get:

$$\sum_{i=0}^{T/\delta - 1} \sum_{u \in \mathcal{N}} a_u \cdot I_u(\delta \cdot i) \leq \sum_{i=0}^{T/\delta - 1} b \ .$$

Since there are T/δ different times in the above range, the right hand side of the last expression is Tb/δ. The lemma now follows by switching the order of summation in the left hand side, and plugging in the definition of I_u^t. □

Lemma 3.2.16. *For every* $0 \leq t \leq T$, $y_u(t) \leq 1 - e^{-I_u^t} + O(\delta) \cdot t$.

Proof. We prove by induction on t that $y_e(t) \leq 1 - e^{-I_e^t} + 0.5\delta t$. For $t = 0$ the lemma holds since $y_e(t) = 0 = 1 - e^{-0} + 0 \cdot \delta$. Assume that the lemma holds for some time t, and let us prove it for time $t + \delta$.

$$\begin{aligned} y_u(t + \delta) &= y_u(t) + \delta I_u(t) \cdot (1 - y_u(t)) \leq (1 - e^{-I_u^t} + 0.5t\delta)(1 - \delta I_u(t)) + \delta I_u(t) \\ &\leq 1 - e^{-I_u^t} \cdot (1 - \delta I_u(t)) + 0.5t\delta \leq 1 - e^{-I_u^t} \cdot [e^{-\delta I_u(t)} - 0.5(\delta I_u(t))^2] + 0.5t\delta \\ &\leq 1 - e^{-I_u^t - \delta I_u(t)} + 0.5\delta^2 + 0.5t\delta = 1 - e^{-I_u^{t+\delta}} + 0.5(t + \delta)\delta \ . \end{aligned}$$ □

The following lemma is a mathematical observation needed to combine the last two lemmata.

Lemma 3.2.17. *Let $c_1, c_2 > 0$, and let z_1, z_2 be two variables whose values obey $c_1 z_1 + c_2 z_2 = s$ for some constant s. Then, the expression $c_1(1-e^{-z_1})+c_2(1-e^{-z_2})$ is maximized when $z_1 = z_2$.*

Proof. The value of z_2 is given, in terms of z_1, by $z_2 = (s - c_1 z_1)/c_2$. Hence, we can derive the expression $c_1(1 - e^{-z_1}) + c_2(1 - e^{-z_2})$ by z_1 as following.

$$\begin{aligned}\frac{d[c_1(1 - e^{-z_1}) + c_2(1 - e^{z_2})]}{dz_1} &= \frac{d[c_1(1 - e^{-z_1}) + c_2(1 - e^{(c_1 z_1 - s)/c_2})]}{dz_1} \\ &= c_1 e^{-z_1}[1 - e^{z_1 + (c_1 z_1 - s)/c_2}] \ .\end{aligned}$$

Observe that the first part of the derivative is always positive. The second part is a decreasing function of z_1, and therefore, the original function has a global maximum when the right hand side equals 0, which happens when:

$$e^{z_1 + (c_1 z_1 - s)/c_2} = 1 \Leftrightarrow z_1 + (c_1 z_1 - s)/c_2 = 0 \Leftrightarrow z_1 = z_2 \ . \qquad \square$$

The following lemma upper bounds, at time T, the left hand side of our general constraint $\sum_{u \in \mathcal{N}} a_u \cdot x_u \leq b$.

Lemma 3.2.18. *Let $\mathcal{N}' \subseteq \mathcal{N}$ be the set of elements with a strictly positive a_u. Then, $\sum_{u \in \mathcal{N}'} a_u \cdot y_u(T) \leq \frac{b}{d(\mathcal{P})} \cdot (1 - e^{-Td(\mathcal{P})} + O(\delta) \cdot T)$.*

Proof. By Lemma 3.2.16:

$$\sum_{u \in \mathcal{N}'} a_u \cdot y_u(T) \leq \sum_{u \in \mathcal{N}'} a_u \cdot (1 - e^{-I_u^T} + O(\delta) \cdot T) \leq \sum_{u \in \mathcal{N}'} a_u \cdot (1 - e^{-I_u^T}) + O(\delta) \cdot Tb/d(\mathcal{P}) \ .$$

The second term of the right hand side is independent of the values taken by the I_u^T's, therefore, we can upper bound the entire right hand side by assigning to the I_u^t's values maximizing the first term. Let us determine these values.

- We can assume the summand is an increasing function of I_u^T, the sum $\sum_{u \in \mathcal{N}'} a_u \cdot I_u^T$ has its maximal value, which is Tb by Lemma 3.2.15.
- By Lemma 3.2.17, the maximum is attained when I_u^T is identical for all elements $u \in \mathcal{N}'$.

It can be easily seen that the sole solution satisfying these conditions is $I_u^T = Tb/\sum_{u \in \mathcal{N}'} a_u$. Plugging this solution to into the previous bound on $\sum_{u \in \mathcal{N}'} a_u \cdot y_u(T)$, we get:

$$\begin{aligned}\sum_{u \in \mathcal{N}'} a_u \cdot y_u(T) &\leq \sum_{u \in \mathcal{N}'} a_u \cdot (1 - e^{-Tb/\sum_{u \in \mathcal{N}'} a_u}) + O(\delta) \cdot Tb/d(\mathcal{P}) \\ &= (1 - e^{-Tb/\sum_{u \in \mathcal{N}'} a_u}) \cdot \sum_{u \in \mathcal{N}'} a_u + O(\delta) \cdot Tb/d(\mathcal{P}) \ .\end{aligned}$$

Let us denote by Σ the sum $\sum_{u \in \mathcal{N}'} a_u$. The first term of the last expression can now be rewritten as $\Sigma(1 - e^{-Tb/\Sigma})$, and its derivative by Σ is:

$$\begin{aligned}\frac{d[\Sigma(1 - e^{-Tb/\Sigma})]}{d\Sigma} &= (1 - e^{-Tb/\Sigma}) - \Sigma \cdot \frac{Tb}{\Sigma^2} e^{-Tb/\Sigma} = 1 - (1 + Tb/\Sigma) \cdot e^{-Tb/\Sigma} \\ &\geq 1 - e^{Tb/\Sigma} \cdot e^{-Tb/\Sigma} = 0 \ .\end{aligned}$$

Hence, increasing the value of Σ only worsens the bound we have on $\sum_{u\in\mathcal{N}'} a_u \cdot y_u(T)$. Therefore, we can plug $\Sigma = b/d(\mathcal{P})$, which is an upper bound on Σ, and get:

$$\sum_{u\in\mathcal{N}'} a_u \cdot y_u(T) \leq (1-e^{-Tb/(b/d(\mathcal{P}))})\cdot b/d(\mathcal{P}) + O(\delta)\cdot Tb/d(\mathcal{P}) = \frac{b}{d(\mathcal{P})}\cdot(1-e^{-Td(\mathcal{P})}+O(\delta)\cdot T) \ .$$

□

As long as the upper bound proved in the last lemma is at most b, the constraint $\sum_{u\in\mathcal{N}} a_u x_u \leq b$ is not violated. The next corollary shows that if $T \leq T_{\mathcal{P}}$, then this is the case.

Corollary 3.2.19. *For $T \leq T_{\mathcal{P}}$, the solution $y(T)$ respects the constraint $\sum_{u\in\mathcal{N}} a_u x_u \leq b$. Moreover, since $\sum_{u\in\mathcal{N}} a_u x_u \leq b$ is an arbitrary constraint of $\mathcal{P}$, $y(T) \in \mathcal{P}$.*

Proof. By Lemma 3.2.18,

$$\begin{aligned}\sum_{u\in\mathcal{N}} a_u \cdot y_u(T) &= \sum_{u\in\mathcal{N}'} a_u \cdot y_u(T) \leq \sum_{u\in\mathcal{N}'} a_u \cdot y_u(T_{\mathcal{P}}) \leq \frac{b}{d(\mathcal{P})}\cdot(1-e^{-T_{\mathcal{P}}d(\mathcal{P})}+O(\delta)\cdot T_{\mathcal{P}}) \\ &= \frac{b}{d(\mathcal{P})}\cdot(1-e^{\ln(1-d(\mathcal{P})+n\delta)}+O(\delta)\cdot T_{\mathcal{P}}) \\ &= \frac{b}{d(\mathcal{P})}\cdot(d(\mathcal{P})-n\delta+O(\delta)\cdot T_{\mathcal{P}}) \leq b \ .\end{aligned}$$

□

This completes the proof of the third (and last) part of Theorem 3.2.2.

3.2.3 Analysis for Monotone f and Binary $\mathcal{P}$

In this section we prove Theorem 3.2.3. Notice that all claims of Section 3.2.2 still hold because the setting considered here is a special case of the setting considered in Section 3.2.2. Theorem 3.2.3 guarantees that given a binary down-monotone solvable polytope $\mathcal{P}$ with a bounded $T_{\mathcal{P}}$ and a normalized monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, there exists a polynomial time algorithm outputting a point $x \in \mathcal{P}$ with $F(x) \geq [1-(1-d(\mathcal{P}))^{1/d(\mathcal{P})}]\cdot f(OPT)$.

A naive attempt to prove Theorem 3.2.3 is to use the measured continuous greedy with stopping time $T = T_{\mathcal{P}}$. Corollary 3.2.19 guarantee that the output x of the measured continuous greedy is a feasible solution. The next lemma lower bounds $F(x)$.

Lemma 3.2.20. *Let x be the output of the measured continues greedy, assuming $T = T_{\mathcal{P}}$. Then, $F(x) \geq [1-(1-d(\mathcal{P}))^{1/d(\mathcal{P})} - O(n^{-2})]\cdot f(OPT) = [1-(1-d(\mathcal{P}))^{1/d(\mathcal{P})} - o(1)]\cdot f(OPT)$.*

Proof. First, observe that $d(\mathcal{P})$ must be at least $1/n$ in a binary down-monotone polytope. Using this observation, let us derive $1-(1-d(\mathcal{P})+c)^{1/d(\mathcal{P})}$ by c, for $c \leq n^{-1}$.

$$\frac{d[1-(1-d(\mathcal{P})+c)^{1/d(\mathcal{P})}]}{dc} = -\frac{(1-d(\mathcal{P})+c)^{1/d(\mathcal{P})-1}}{d(\mathcal{P})} \geq -\frac{1}{d(\mathcal{P})} \geq -n \ .$$

Hence, for $c \leq 1/n$, $1-(1-d(\mathcal{P})+c)^{1/d(\mathcal{P})} \geq 1-(1-d(\mathcal{P}))^{1/d(\mathcal{P})} - cn$. The discussion in Section 3.2.2 proves that Corollary 3.1.7 (including its proof) applies to the measured

continuous greedy. By the proof of Corollary 3.1.7:

$$\begin{aligned}
F(y(x)) &\geq [1 - e^{-T_{\mathcal{P}}} - O(n^3\delta) \cdot T_{\mathcal{P}}] \cdot f(OPT) \\
&= [1 - (1 - d(\mathcal{P}) + n\delta)^{1/d(\mathcal{P})} - O(n^3\delta) \cdot T_{\mathcal{P}}] \cdot f(OPT) \\
&\geq [1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})} - n^2\delta - O(n^3\delta) \cdot T_{\mathcal{P}}] \cdot f(OPT) \\
&= [1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})} - O(n^3\delta) \cdot T_{\mathcal{P}}] \cdot f(OPT) \\
&= [1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})} - O(n^{-2})] \cdot f(OPT) \ ,
\end{aligned}$$

where the last equality follows since $\delta \leq n^{-5}$. □

Unfortunately, Lemma 3.2.20 is not enough for proving Theorem 3.2.3 because of the $o(1)$ term in its guarantee. To solve that problem, consider Algorithm 3, which guesses one element of the optimal solution before applying the measured continuous greedy.

Algorithm 3: Measured Continuous Greedy with Enumeration$(f, \mathcal{P})$

`// Guess`

If n is small enough, guess the optimal solution and terminate.

Otherwise, guess an element $u^* \in OPT$ such that $f(u^*) \geq f(OPT)/n$.

`// Definitions`

Let $\mathcal{P}'$ be the polytope formed from $\mathcal{P}$ by setting x_{u^*} to be identically 1.

Let $\mathcal{N}_0$ be the set of elements of $\mathcal{N}$ whose corresponding variables are identically 0 in $\mathcal{P}'$.

Set $\mathcal{N}' \leftarrow \mathcal{N} - \mathcal{N}_0 - \{e^*\}$.

Define $f'(A) = f(A + u^*) - f(\{u^*\})$.

`// Solve`

Use the measured continuous greedy algorithm with ground set $\mathcal{N}'$, submodular function f', polytope $\mathcal{P}'$ and stopping time $T_{\mathcal{P}}$.

Let $y(T_{\mathcal{P}})$ be the output of the measured continuous greedy algorithm.

Output $y(T_{\mathcal{P}}) \vee \mathbf{1}_{u^*}$.

If n is small enough, then clearly the Algorithm 3 is optimal. Hence, we can assume from now on that n is large. Let us make sure that there is an element u^* for the algorithm to guess.

Lemma 3.2.21. *There is an element $u^* \in OPT$ satisfying the requirement of the algorithm, i.e., $f(u^*) \geq f(OPT)/n$.*

Proof. Assume for the sake of contradiction that every element $u \in OPT$ has $f(u) < f(OPT)/n$. Then, we get:

$$f(OPT) \leq \sum_{u \in OPT} f(u) < \sum_{u \in OPT} \frac{f(OPT)}{n} = \frac{|OPT| \cdot f(OPT)}{n} \leq f(OPT) \ ,$$

which is, of course, a contradiction. □

Algorithm 3 applies the measured continuous greedy algorithm with the objective function f'. The following lemma shows that f' has the properties required by Theorem 3.2.2.

Lemma 3.2.22. *f' is a normalized monotone submodular function over the ground set $\mathcal{N}'$.*

Proof. f is submodular because for every two subsets $A, B \in \mathcal{N}'$:

$$\begin{aligned} f'(A) + f'(B) &= f(A + u^*) - f(\{u^*\}) + f(B + u^*) - f(\{u^*\}) \\ &\geq f(A \cup B + u^*) + f((A + u^*) \cap (B + u^*)) - 2f(\{u^*\}) \\ &= f(A \cup B + u^*) - f(\{u^*\}) + f((A \cap B) + u^*) - f(\{u^*\}) \\ &= f'(A \cup B) + f'(A \cap B) \ . \end{aligned}$$

f is monotone because for every $A \subseteq B \subseteq \mathcal{N}'$:

$$f'(A) = f(A + u^*) - f(\{u^*\}) \leq f(B + u^*) - f(\{u^*\}) = f'(B) \ .$$

And finally, f' is normalized because:

$$f'(\varnothing) = f(\varnothing + u^*) - f(\{u^*\}) = 0 \ . \qquad \square$$

In order for us to use the full power of Theorem 3.2.2, we need to demonstrate that $\mathcal{P}'$ is a packing polytope and bound its density.

Lemma 3.2.23. *The polytope $\mathcal{P}'$ can be represented as a packing polytope over $\mathcal{N}'$ with $d(\mathcal{P}') \geq d(\mathcal{P})$.*

Proof. Consider a general constraint $\sum_{u \in \mathcal{N}} a_u x_u \leq b$ of $\mathcal{P}$. The corresponding constraint in $\mathcal{P}'$ is $\sum_{u \in \mathcal{N}'} a_u x_u \leq b - a_{u^*}$. If all the coefficients a_u on the left side of this constraint are 0, the constraint always holds, and therefore, can be removed. Hence, we can assume this is not the case, and there exists $u' \in \mathcal{N}'$ such that $a_{u'} = 1$.

Clearly $a_{u^*} \leq b$ because otherwise u^* could not be a member of a feasible solution, contradicting our assumption that $u^* \in OPT$. Therefore, the free coefficient of the constraint is either 0 or 1. If $b - a_{e^*} = 0$, then this constraint implies $x_{u'} = 0$, contradicting the fact $u' \in \mathcal{N}'$. Hence, $b - a_{u^*} = 1$, which implies $a_{u^*} = 0$. Thus,

$$\frac{b - a_{u^*}}{\sum_{u \in \mathcal{N}'} a_u} \geq \frac{b}{\sum_{u \in \mathcal{N}} a_u} \ .$$

The last inequality holds for a general constraint of $\mathcal{P}'$, and therefore, $d(\mathcal{P}') \geq d(\mathcal{P})$. $\square$

Corollary 3.2.24. *Let x be the output of Algorithm 3. Then, $x \in \mathcal{P}$.*

Proof. Lemma 3.2.23 imply $T_{\mathcal{P}'} \geq T_{\mathcal{P}}$, and therefore, $y(T_{\mathcal{P}}) \in \mathcal{P}'$ by Theorem 3.2.2. Hence, by the definition of $\mathcal{P}'$, $x = y(T_{\mathcal{P}}) \vee 1_{e^*} \in \mathcal{P}$. $\square$

To complete the proof of Theorem 3.2.3, we are only left to lower bound $F(x)$.

Lemma 3.2.25. $F(x) \geq [1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})}] \cdot f(OPT)$.

Proof. Recall that one of the conditions of Theorem 3.2.3 is that $T_{\mathcal{P}}$ should be bounded. The only case in which $T_{\mathcal{P}}$ is unbounded is when $d(\mathcal{P})$ approaches 1. Hence, we know that $d(\mathcal{P})$ is bounded away from 1.

Let F' be the multilinear extension of f', and observe that $OPT - \{e^*\}$ is the optimal point in $\mathcal{P}'$. The above lemmata prove that f' and P' obey all the requirements of Theorem 3.2.2, and therefore, we can apply Lemma 3.2.20 to them, yielding $F'(y(T_{\mathcal{P}})) \geq$

$[1-(1-d(\mathcal{P}))^{1/d(\mathcal{P})}-cn^{-2}]\cdot f'(OPT-\{e^*\})$ for some constant c (assuming large enough n). Hence,

$$\begin{aligned}
F(x) &= \mathbb{E}[f(\{e^*\}\cup \mathtt{R}(y(T_{\mathcal{P}})))] = \mathbb{E}[f(\{e^*\}) + f'(\mathtt{R}(y(T_{\mathcal{P}})))] = f(\{e^*\}) + F'(y(T_{\mathcal{P}})) \\
&\geq f(\{e^*\}) + [1-(1-d(\mathcal{P}))^{1/d(\mathcal{P})} - cn^{-2}]\cdot f'(OPT-\{e^*\}) \\
&= ((1-d(\mathcal{P}))^{1/d(\mathcal{P})} + cn^{-2})\cdot f(\{e^*\}) + (1-(1-d(\mathcal{P}))^{1/d(\mathcal{P})} - cn^{-2})\cdot f(OPT) \\
&\geq ((1-d(\mathcal{P}))^{1/d(\mathcal{P})} + cn^{-2})\cdot f(OPT)/n + (1-(1-d(\mathcal{P}))^{1/d(\mathcal{P})} - cn^{-2})\cdot f(OPT) \\
&\geq [1-(1-d(\mathcal{P}))^{1/d(\mathcal{P})} + n^{-1}((1-d(\mathcal{P}))^{1/d(\mathcal{P})} - cn^{-1})]\cdot f(OPT) \\
&\geq [1-(1-d(\mathcal{P}))^{1/d(\mathcal{P})}]\cdot f(OPT)\ ,
\end{aligned}$$

where the last inequality holds for large enough n since c is a constant and $d(\mathcal{P})$ is bounded away from 1. □

3.3 Applications of the Measured Continuous Greedy

Theorems 3.2.1, 3.2.2 and 3.2.3 immediately provide improved approximations for various problems. We elaborate now on a few of these, starting with the non-monotone case. Theorem 3.2.1, gives an improved $(1/e-o(1))$-approximation for finding a fractional solution for any down-monotone and solvable polytope $\mathcal{P}$. Examples of some well-studied problems for which this provides improved approximation are maximization of a non-monotone submodular function f over a single matroid [20, 37, 81] and over $O(1)$ knapsack constraints [20, 61, 57]. For both we provide an improved approximation of about $1/e$. Note that both problems are known to have an approximation of roughly ≈ 0.325 [20] via the simulated annealing technique of [37].

For the monotone case, Theorems 3.2.2 and 3.2.3 can be immediately used to obtain improved approximations for various problems. Most notable is the well studied `Submodular Welfare` (SW) problem (refer to [16, 17, 25, 26, 29, 31, 32, 54, 63, 73, 68, 81] for previous results on SW and additional closely related variants of the problem). The above theorems provide *tight* approximations for *any* number of players k, which exactly matches the $(1-(1-1/k)^k)$-hardness result [68]. This improvement is most significant for small values of k. Another problem we consider is `Submodular Max-SAT` (SSAT). SSAT is a generalization of both `Max-SAT` and SW with two players, in which a monotone submodular function f is defined over the clauses of a CNF formula, and the goal is to find an assignment maximizing the value of f over the set of satisfied clauses. For SSAT we get a 3/4 approximation. The above main applications are summarized in Table 3.1. For other applications involving our algorithm and the framework of [20], see Section 6.

Table 3.1: Main applications.

Constraint	**This Thesis**	**Previous Result**		**Hardness***	
Matroid (non-monotone)	$1/e-o(1)$	0.325	[20]	0.478	[37]
$O(1)$-Knapsacks (non-monotone)	$1/e-\varepsilon$	0.325	[20]	$1/2^{**}$	
SW (k players)	$1-(1-1/k)^k$	$\max\left\{1-\frac{1}{e}, \frac{k}{(2k-1)}\right\}$	[26, 16]	$1-(1-1/k)^k$	[68]
SSAT	3/4	2/3	[5]	3/4	[81]

* All hardness results are for the value oracle model, and are information theory based.
** Can be derived from the method of [81].

3.3.1 Maximizing a Submodular Function Subject to Matroid Constraint

In this section we consider the problem of maximizing a submodular function subject to a matroid constraint. Formally, given a matroid $M = (\mathcal{N}, \mathcal{I})$ and a submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, the objective is to find an independent set $S \in \mathcal{I}$ maximizing $f(S)$. Calinescu et al. [16] considered the case that the submodular function is monotone. They used the continuous greedy algorithm to prove the following theorem.

Theorem 3.3.1. *There is a polynomial time $(1 - 1/e)$-approximation algorithm for maximizing a normalized monotone submodular function subject to a matroid constraint.*

The above theorem is tight even for uniform matroids[6] [71]. Using the measured continuous greedy algorithm instead of the traditional continuous greedy, we get the following result for general non-monotone functions.

Theorem 3.3.2. *There is a polynomial time $(1/e - o(1))$-approximation algorithm for maximizing a general non-negative submodular function subject to a matroid constraint.*

Proof. Let $\mathcal{P}(M)$ be the matroid polytope corresponding to M. Since $\mathcal{P}(M)$ is a solvable down monotone polytope, by Theorem 3.2.1, applying the measured continuous greedy to it, with stopping time $T = 1$, produces a point $x \in \mathcal{P}(M)$ such that $F(x) \geq [1/e - o(1)] \cdot f(OPT)$.

The point x can then be rounded using pipage rounding. This rounding procedure returns a random set S which is an independent set of M and obeys $\mathbb{E}[f(S)] \geq F(x) \geq [e^{-1} - o(1)] \cdot f(OPT)$. □

This improves over the previous 0.325-approximation of [20].

3.3.2 Maximizing a Submodular Function Subject to Knapsack Constraints

In this section we consider the problem of maximizing a submodular function subject to a constant number of knaspack constraints. Formally, we are given a ground set $\mathcal{N}$, a set of d knapsack constraints over this ground set (where d is considered to be a constant) and a non-negative submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$. The objective is to find a set $S \subseteq \mathcal{N}$ satisfying all knapsack constraints and maximizing $f(S)$.

Let $\mathcal{P}$ be the polytope defined by the d knapsack constraints and the cube $[0, 1]^{\mathcal{N}}$. Observe that $\mathcal{P}$ is a down monotone solvable polytope. The following theorem shows that it is possible to round fractional points in $\mathcal{P}$.

Theorem 3.3.3 (Theorem 2.6 in [57])**.** *Suppose there is an α-approximation polynomial time algorithm for finding a point $x \in \mathcal{P}$ maximizing $F(x)$. Then, for every constant $\varepsilon > 0$, there is a polynomial time randomized $(\alpha - \varepsilon)$-approximation algorithm for maximizing a non-monotone submodular function subject to d knapsack constraints.*

Theorem 3.3.3 assumes the existence of a fractional approximation algorithm. We observe that the measured continuous greedy algorithm can be used as such an algorithm. For monotone submodular functions, this gives $(1 - 1/e - \varepsilon)$-approximation, which is the result obtained by [57]. For general non-monotone functions, we get the following result.

[6] In a uniform matroid, a set is independent if it contains at most k elements of the ground set, for some fixed k.

Corollary 3.3.4. *For any constant $\varepsilon > 0$ and constant integer d, there is a polynomial time $(1/e-\varepsilon)$-approximation algorithm for maximizing a general non-negative submodular function subject to d knapsack constraints.*

Proof. By Theorem 3.2.1, applying the measured continuous greed to $\mathcal{P}$, with $T = 1$, produces a point $x \in \mathcal{P}$ such that $F(x) \geq [1/e - o(1)] \cdot f(OPT)$. The corollary now follows from Theorem 3.3.3. □

This improves over the previous 0.325-approximation of [57].

3.3.3 The (d, r)-`Submodular Partition` Problem

A (d, r)-partition matroid is a matroid defined over a groundset $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2 \cup \ldots \cup \mathcal{N}_m$, where $|\mathcal{N}_i| = r$ for every $1 \leq i \leq m$. A set $S \subseteq \mathcal{I}$ is independent if it contains up to d elements of each subset $\mathcal{N}_i$. In the (d, r)-`Submodular Partition` problem (considered by [5]), given a (d, r)-partition matroid M and a normalized monotone submodular function f, the objective is to find an independent set S maximizing $f(S)$.

Observation 3.3.5. *Let M be a (d, r)-partition matroid, then $\mathcal{P}(M)$ is a matroid with density $d(\mathcal{P}(\mathcal{M})) = d/r$. For $d = 1$, it is also a binary packing matroid.*

Proof. $\mathcal{P}(M)$ is a polytope defined by m constraints, one for each subset $\mathcal{N}_i$. The constraint of $\mathcal{N}_i$ is $\sum_{u \in \mathcal{N}_i} x_u \leq d$. The number of terms in the sum is r, and therefore, the density imposed by this constraint is: $d / \sum_{u \in \mathcal{M}_i} d = d/r$.

For $d = 1$, observe that all coefficients in the above constraint belong to $\{0, 1\}$, and therefore, the matroid is a binary packing matroid. □

Theorem 3.3.6. *There is a polynomial time $(1 - (1 - d/r)^{r/d} - o(1))$-approximation algorithm for (d, r)-`Submodular Partition`. For $d = 1$, the approximation ratio improves to $(1 - (1 - 1/r)^r$.*

Proof. Observation 3.3.5 together with Theorem 3.2.2 gives a polynomial time approximation algorithm that finds a point $x \in \mathcal{P}(M)$ with $F(x) \geq [1-(1-d/r)^{r/d}-o(1)] \cdot f(OPT)$. The point x can then be rounded using pipage rounding. This rounding returns an independent random set S of M, such that: $\mathbb{E}[f(S)] \geq F(x) \geq [1-(1-d/r)^{r/d}-o(1)] \cdot f(OPT)$. This completes the proof of the first part of the theorem.

For $d = 1$, we know that the (d, r)-partition matroid is a binary packing matroid, and therefore, we can replace Theorem 3.2.2 with Theorem 3.2.3 in the proof of the first part of the theorem, yielding the second part of the theorem. □

The last result matches the hardness result given by [5] for every pair of d and r, up to low order terms.

3.3.4 The `Submodular Max-SAT` Problem

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

Usually, an assignment ϕ can give each variable exactly a single truth value. However, for the sake of the algorithm we extend the notion of assignments, and think of an extended assignment ϕ' which is a relation $\phi' \subseteq \mathcal{N} \times \{0, 1\}$. That is, the assignment ϕ' can assign up to 2 truth values to each variable. A clause C is satisfied by an (extended) assignment

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

Lemma 3.3.7. *The function g is a normalized monotone submodular function.*

Proof. It is easy to see that g is normalized and monotone, however, proving it is also submodular requires some work. Consider two sets $A, B \subseteq \mathcal{N} \times \{0,1\}$. Using the submodularity and monotonicity of f, we get:

$$\begin{aligned} g(A) + g(B) &= f\left(C(A)\right) + f\left(C(B)\right) \geq f\left(C(A) \cup C(B)\right) + f\left(C(A) \cap C(B)\right) \\ &\geq f\left(C(A \cup B)\right) + f\left(C(A \cap B)\right) = g(A \cup B) + g(A \cap B) \ . \end{aligned}$$ □

Notice that we have just restated SSAT as an instance of $(1,2)$-`Submodular Partition`. Hence, we get the following corollary.

Corollary 3.3.8. *There is an α-approximation for SSAT if $(1,2)$-`Submodular Partition` has such an approximation. Hence, by Theorem 3.3.6, there is a $3/4$-approximation algorithm for SSAT.*

Since we identified SSAT with $(1,2)$-`Submodular Partition`, the hardness of [5] applies also to SSAT, and shows that the last corollary is tight.

3.3.5 The `Submodular Welfare` Problems

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

Lemma 3.3.9. *An α-approximation algorithm for $(1,k)$-`Submodular Partition` implies an α-approximation for SW with k players.*

Proof. Given an instance I of SW with k players, let us transform it into an equivalent instance I' of $(1,k)$-`Submodular Partition`. Let $u_1, \ldots, u_n$ denote the elements of $\mathcal{N}$. The ground set of I' is $\mathcal{N}' = \cup_{i=1}^{n} \mathcal{N}'_i$, where $\mathcal{N}'_i = \{u_i\} \times \{1, 2, \ldots, k\}$. The objective function of I' is $g(S) = \sum_{i=1}^{k} f_i(\{u | (u,i) \in S\})$. It is easy to see that g is normalized and monotone, however, proving it is submodular requires some work. Consider two sets

$A, B \subseteq \mathcal{N}'$. Using the submodularity of f, we get:

$$\begin{aligned} g(A) + g(B) &= \sum_{i=1}^{k} f_i(\{u|(u,i) \in A\}) + \sum_{i=1}^{k} f_i(\{u|(u,i) \in B\}) \\ &\geq \sum_{i=1}^{k} f_i(\{u|(u,i) \in A\} \cup \{u|(u,i) \in B\}) \\ &\quad + \sum_{i=1}^{k} f_i(\{u|(u,i) \in A\} \cap \{u|(u,i) \in B\}) \\ &= \sum_{i=1}^{k} f_i(\{u|(u,i) \in A \cup B\}) + \sum_{i=1}^{k} f_i(\{u|(u,i) \in A \cap B\}) \\ &= g(A \cup B) + g(A \cap B) \ . \end{aligned}$$

Next, show how to transform any solution of I to a feasible set of I', and vice versa. Consider a solution $\mathcal{N}_1, \ldots, \mathcal{N}_k$ of I, and let us construct from it the set: $S = \cup_{i=1}^{k} \{(u,i)|u \in \mathcal{N}_i\}$. Observe that:

$$g(S) = \sum_{i=1}^{k} f_i(\{u|(u,i) \in S\}) = \sum_{i=1}^{k} f_i(\mathcal{N}_i) \ .$$

Thus, S has exactly the same value as $\mathcal{N}_1, \ldots, \mathcal{N}_k$. On the other hand, given a set S which is a feasible solution for I', let us create a solution $\mathcal{N}_1, \ldots, \mathcal{N}_k$ for I.

$$\mathcal{N}_i = \{u|(u,i) \in S\} \ .$$

By the definition of g, $g(S)$ is equal to the value of the solution $\mathcal{N}_1, \ldots, \mathcal{N}_k$. The fact that any feasible solution of I can be translated into a feasible set S of I' with the same value, and vice versa implies that both instances have the same optimal value.

We are now ready to use the above transformation of I into I' to translate any α-approximation algorithm for $(1,k)$-`Submodular Partition` into an α-approximation algorithm for SW with k players. Given an instance of SW with k players, transform it into an instance of $(1,k)$-`Submodular Partition`, find an approximate solution for the resulting instance, and then use the procedure described above to convert it into an approximate solution for the original SW instance. □

Corollary 3.3.10. *There is a polynomial time $1 - (1 - 1/k)^k$-approximation algorithm for SW.*

Proof. Follows immediately from Theorem 3.3.6 and Lemma 3.3.9. □

The last corollary is tight by the result of [68].

3.A Down Monotone Polytopes in the Hypercube $[0,1]^{\mathcal{N}}$

Let $\mathcal{P} \subseteq [0,1]^{\mathcal{N}}$ be a down monotone polytope defined by positivity constraints and m additional inequality constraints. Let $\sum_{u \in \mathcal{N}} a_{i,u} x_u \leq b_i$ be the i^{th} constraint defining $\mathcal{P}$. In this section we prove that one can assume the coefficients of the inequality constraints are all non-negative and that $0 < d(\mathcal{P}) \leq 1$.

Observation 3.A.1. *We may assume $\mathcal{P}$ has a positive sign constraint $x_u \geq 0$, and an inequality constraint of the form $x_u \leq 1$ for every $u \in \mathcal{N}$.*

Proof. Since $\mathcal{P} \subseteq [0,1]^{\mathcal{N}}$, adding these constraints will not effect $\mathcal{P}$. □

Lemma 3.A.2. *For every $1 \leq i \leq m$, b_i is non-negative.*

Proof. Since $\mathcal{P}$ is down monotone, $1_\varnothing \in \mathcal{P}$. The point $1_\varnothing$ induce the value of 0 on the left hand side of all constraints, and therefore, the free coefficients must be non-negative. □

Lemma 3.A.3. *For every $1 \leq i \leq m$ and $u \in \mathcal{N}$, $a_{i,u}$ is non-negative without loss of generality.*

Proof. Assume this is not the case, then let $a_{j,u'}$ be a negative coefficient. Let $\mathcal{P}'$ be the polytope defined by the same set of constraints as $\mathcal{P}$ with the sole modification that $a_{j,u'}$ is changed to 0. By Observation 3.A.1, the change we made only tightens the constraint, and therefore, $\mathcal{P}' \subseteq \mathcal{P}$. Let us show that the last containment is in fact an equality. Let x be a point in $\mathcal{P}$, and let $\sum_{u\in\mathcal{N}} a'_{i,u}x_u \leq b'_i$ be the i^{th} constraint of $\mathcal{P}'$.

For every constraint $i \neq j$, $\mathcal{P}$ and $\mathcal{P}'$ share the same coefficients, and therefore,

$$\sum_{u\in\mathcal{N}} a'_{i,u}x_u = \sum_{u\in\mathcal{N}} a_{i,u}x_u \leq b_i = b'_i \ .$$

Let $x' = x \wedge 1_{\mathcal{N}-\{u'\}}$. Observe that $x' \in \mathcal{P}$ due to the down monotonicity of $\mathcal{P}$. Also, $\mathcal{P}$ and $\mathcal{P}'$ share all coefficients of the j^{th} constraint except for $a_{j,u'}$, and therefore,

$$\sum_{u\in\mathcal{N}} a'_{j,u}x_u = \sum_{u\in\mathcal{N}-\{u'\}} a'_{j,u}x_u = \sum_{u\in\mathcal{N}-\{u'\}} a_{j,u}x'_u = \sum_{u\in\mathcal{N}} a_{j,u}x'_u \leq b_j = b'_j \ .$$

Where the inequality holds since $x' \in \mathcal{P}$. We proved that $x \in \mathcal{P}$ implies $x \in \mathcal{P}'$, and therefore, $\mathcal{P} = \mathcal{P}'$. Hence, replacing a negative coefficient $a_{j,u'}$ by 0 does not change the polytope. The lemma now follows by repeating the argument for every negative coefficient. □

Lemma 3.A.4. *For every constraint i, $b_i > 0$ without loss of generality.*

Proof. By Lemma 3.A.2, $b_i \geq 0$. Let $\mathcal{N}' = \{u \in \mathcal{N} | a_{u,i} > 0\}$. If $b_i = 0$, the constraint implies that for every $u \in \mathcal{N}'$, the coordinate x_u must be zero everywhere in $\mathcal{P}$, and therefore, the elements of $\mathcal{N}'$ can be removed from the ground set. After the removal of $\mathcal{N}'$'s elements, we are left with a constraint of the form:

$$\sum_{u\in\mathcal{N}} 0 \cdot x_u \leq 0 \ ,$$

and such constraints are meaningless, and can be removed without effecting $\mathcal{P}$. □

Lemma 3.A.5. *For every constraint i, $\sum_{u\in\mathcal{N}} a_{u,i} > 0$ without loss of generality.*

Proof. By Lemma 3.A.3, $\sum_{u\in\mathcal{N}} a_{u,i} \geq 0$. If $\sum_{u\in\mathcal{N}} a_{u,i} = 0$, the constraints is satisfied for every assignment, and therefore, can be removed. □

Lemma 3.A.6. *For every constraint i, $\sum_{u\in\mathcal{N}} a_{u,i} > b_i$ without loss of generality.*

Proof. For every point $x \in [0,1]^{\mathcal{N}}$ it holds that $\sum_{u\in\mathcal{N}} a_{u,i}x_u \leq \sum_{u\in\mathcal{N}} a_{u,i}$. Hence, if $\sum_{u\in\mathcal{N}} a_{u,i} \leq b_i$, then the i^{th} constraint is redundant for points in $[0,1]^{\mathcal{N}}$. By Observation 3.A.1, the removal of this constraint will not introduce to $\mathcal{P}$ points outside of $[0,1]^{\mathcal{N}}$, and therefore, will not modify $\mathcal{P}$. □

Corollary 3.A.7. $0 < d(\mathcal{P}) \leq 1$ *without loss of generality.*

Proof. Fix an inequality constraint i. Lemmata 3.A.4 and 3.A.5 guarantee $b_i > 0$ and $\sum_{u\in\mathcal{N}} a_{u,i} > 0$, respectively. Hence, $b_i / \sum_{u\in\mathcal{N}} a_{u,i}$ is positive. On the other hand, from Lemma 3.A.6 we get $\sum_{u\in\mathcal{N}} a_{u,i} > b_i$. Since this is true for every inequality constraint of $\mathcal{P}$, its density must be in the range $(0,1]$. □

Chapter 4

Unconstrained Submodular Maximization

`Unconstrained Submodular Maximization` (USM) is perhaps the most basic submodular maximization problem. Given a non-negative submodular fucntion f, the goal in this problem is to find a subset $S \subseteq \mathcal{N}$ maximizing $f(S)$. Note that there is no restriction on the choice of S, as any subset of $\mathcal{N}$ is a feasible solution. USM captures many well studied problems such as Max-Cut, Max-DiCut [38, 43, 45, 51, 53, 78], and variants of Max-SAT and maximum facility location [1, 23, 24]. Moreover, USM has various applications in other, more practical, settings such as marketing in social networks [44], revenue maximization with discrete choice [3], and algorithmic game theory [28, 75].

USM has been studied starting from the 60's in the Operations Research community [3, 21, 39, 40, 41, 52, 60, 67]. Not surprisingly, as USM captures NP-hard problems, all these works provide algorithms that either solve specific cases of the problem, provide exact algorithms whose time complexity cannot be efficiently bounded, or provide efficient algorithms whose output has no provable guarantee.

The first rigorous study of the problem was conducted by Feige et al. [30], who provided several constant approximation factor algorithms for USM. They proved that a subset S chosen uniformly at random constitutes a $(1/4)$-approximation. Additionally, they also described two local search algorithms. The first uses f as the objective function, and provides an approximation of $1/3$. The second uses a noisy version of f as the objective function, and achieves an improved approximation guarantee of $2/5$. Gharan and Vondrák [37] showed that an extension of the last method, known as *simulated annealing*, can provide an improved approximation of roughly 0.41. Their algorithm, like that of Feige et al. [30], uses local search with a noisy objective function. However, in [37] the noise decreases as the algorithm advances, as opposed to being constant as in [30]. Feldman et al. [33] observed that if the simulated annealing algorithm of [37] outputs a relatively poor solution, then it must generate at some point a set S which is *structurally similar* to some optimal solution. Moreover, they showed that this structural similarity can be traded for value, providing an overall improved approximation of roughly 0.42.

It is important to note that for many special cases of USM better approximation factors are known. For example, the seminal work of Goemans and Williamson [38] provides an 0.878-approximation for Max-Cut based on a semidefinite programming approach, and Ageev and Sviridenko [1] provide an approximation of 0.828 for the maximum facility location problem.

On the negative side, Feige et al. [30] studied the hardness of USM assuming the function f is given via a value oracle. They proved that for any constant $\varepsilon > 0$, any

algorithm achieving an approximation of $(1/2 + \varepsilon)$ requires an exponential number of oracle queries. This hardness result holds even if f is symmetric, in which case it is known to be tight [30]. Recently, Dobzinski and Vondrák [27] proved that even if f has a compact representation (which is part of the input), the above hardness still holds assuming $RP \neq NP$.

In this section we resolve the approximability of USM. This section is based on [14]. We design a tight linear time (1/2)-approximation for the problem. Let us begin by presenting a simple greedy-based algorithm that provides a (1/3)-approximation for USM.

Theorem 4.0.8. *There exists a deterministic linear time* (1/3)*-approximation algorithm for the* `Unconstrained Submodular Maximization` *problem.*

We show that our analysis of the last algorithm is tight by providing an instance for which the algorithm achieves an approximation of $1/3 + \varepsilon$ for an arbitrary small $\varepsilon > 0$. To improve the algorithm, we incorporate randomness into its choices. The result is an optimal algorithm for USM with the same time complexity.

Theorem 4.0.9. *There exists a randomized linear time* (1/2)*-approximation algorithm for the* `Unconstrained Submodular Maximization` *problem.*

In both Theorems 4.0.8 and 4.0.9 we assume that a single query to the value oracle takes $O(1)$ time. Thus, a linear time algorithm is an algorithm which makes $O(n)$ oracle queries plus $O(n)$ other operations, where n is the size of the ground set $\mathcal{N}$.

Building on the above two theorems, we provide two additional approximation algorithms for `Submodular Max-SAT` and `Submodular Welfare` with 2 players (for the exact definition of these problems please refer to Section 3.3). A tight approximation was already given for both problems in Section 3.3. However, the algorithms given here run in linear time, thus, significantly improving the time complexity, while achieving the same performance guarantee.

Theorem 4.0.10. *There exists a randomized linear time* (3/4)*-approximation algorithm for the* `Submodular Max-SAT` *problem.*

Theorem 4.0.11. *There exists a randomized linear time* (3/4)*-approximation algorithm for the* `Submodular Welfare` *problem with* 2 *players .*

4.1 A Deterministic Linear Time (1/3)-Approximation Algorithm for `Unconstrained Submodular Maximization`

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

Despite the failure of the greedy algorithm when applied separately to either f or $\bar{f}$, we show that a correlated execution on both f and $\bar{f}$ provides a much better result. The

algorithm proceeds in n iterations that correspond to some arbitrary order $u_1, \ldots, u_n$ of the ground set $\mathcal{N}$. The algorithm maintains two solutions X and Y. Initially, we set the solutions to $X_0 = \emptyset$ and $Y_0 = \mathcal{N}$. In the i^{th} iteration the algorithm either adds u_i to X_{i-1} or removes u_i from Y_{i-1}. This decision is done greedily based on the marginal gain of each of the two options. Eventually, after n iterations both solutions coincide, and we get $X_n = Y_n$; this is the output of the algorithm. A formal description of the algorithm appears as Algorithm 4.

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

$X_0 \leftarrow \emptyset$, $Y_0 \leftarrow \mathcal{N}$.
for $i = 1$ *to* n **do**
  $a_i \leftarrow f(X_{i-1} + u_i) - f(X_{i-1})$.
  $b_i \leftarrow f(Y_{i-1} - u_i) - f(Y_{i-1})$.
  if $a_i \geq b_i$ **then** $X_i \leftarrow X_{i-1} + u_i$, $Y_i \leftarrow Y_{i-1}$.
  else $X_i \leftarrow X_{i-1}$, $Y_i \leftarrow Y_{i-1} - u_i$.
return X_n (or equivalently Y_n).

The rest of this section is devoted for proving Theorem 4.0.8, *i.e.*, we prove that the approximation guarantee of Algorithm 4 is 1/3. We start with the following useful lemma.

Lemma 4.1.1. *For every* $1 \leq i \leq n$,

$$a_i + b_i \triangleq f(X_{i-1} + u_i) - f(X_{i-1}) + f(Y_{i-1} - u_i) - f(Y_{i-1}) \geq 0 \ . \tag{4.1}$$

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

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

□

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

Lemma 4.1.2. *For every* $1 \leq i \leq n$, $f(OPT_{i-1}) - f(OPT_i) \leq [f(X_i) - f(X_{i-1})] + [f(Y_i) - f(Y_{i-1})]$.

Before proving Lemma 4.1.2, let us show that Theorem 4.0.8 follows from it.

Proof of Theorem 4.0.8. Summing up Lemma 4.1.2 for every $1 \leq i \leq n$ gives:

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

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

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

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

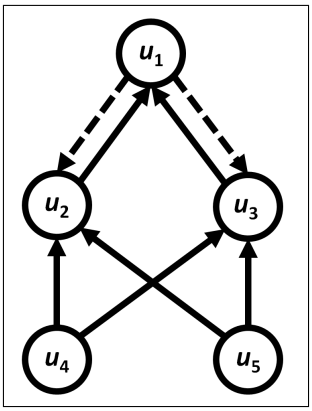

Figure 4.1: Tight example for Algorithm 4. The weight of the dashed edges is $1-\varepsilon$. All other edges have weight of 1.

It is left to prove Lemma 4.1.2.

Proof of Lemma 4.1.2. Assume w.l.o.g that $a_i \geq b_i$, *i.e.*, $X_i \leftarrow X_{i-1} + u_i$ and $Y_i \leftarrow Y_{i-1}$ (the other case is analogous). Notice that in this case $OPT_i = (OPT \cup X_i) \cap Y_i = OPT_{i-1} + u_i$ and $Y_i = Y_{i-1}$. Using both, the inequality that we need to prove becomes:

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

We now consider two cases. If $u_i \in OPT$, then the left hand side of the last inequality is 0, and all we need to show is that a_i is non-negative. This is true since Lemma 4.1.1 gives that $a_i + b_i \geq 0$, and we assumed that $a_i \geq b_i$.

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

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

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

4.1.1 Tight Example

In this section we show that the analysis of Algorithm 4 is tight.

Theorem 4.1.3. *For an arbitrarily small constant $\varepsilon > 0$, there exists a submodular function for which Algorithm 4 provides only $(1/3 + \varepsilon)$-approximation.*

Proof. The proof is done by analyzing Algorithm 4 on the cut function of the weighted digraph given in Figure 4.1. The maximum weight cut in this digraph is $\{u_1, u_4, u_5\}$. This cut has weight of $6-2\varepsilon$. We claim that Algorithm 4 outputs the cut $\{u_2, u_3, u_4, u_5\}$, whose value is only 2 (assuming the nodes of the graph are considered at a given order). Hence, the approximation guarantee of Algorithm 4 on the above instance is:

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

Let us track the execution of the algorithm. Let X_i, Y_i be the solutions maintained by the algorithm. Initially $X_0 = \emptyset, Y_0 = \{u_1, u_2, u_3, u_4, u_5\}$. Note that in case of a tie ($a_i = b_i$), Algorithm 4 takes the node u_i.

1. In the first iteration the algorithm considers u_1. Adding this node to X_0 increases the value of this solution by $2 - 2\varepsilon$. On the other hand, removing this node from Y_0 increases the value of Y_0 by 2. Hence, $X_1 \leftarrow X_0, Y_1 \leftarrow Y_0 - u_1$.

2. Let us inspect the next two iterations in which the algorithm considers u_2, u_3. One can easily verify that these two iterations are independent, hence, we consider only u_2. Adding u_2 to X_1 increases its value by 1. On the other hand, removing u_2 from $Y_1 = \{u_2, u_3, u_4, u_5\}$ also increases the value of Y_1 by 1. Thus, the algorithm adds u_2 to X_1. Since u_2 and u_3 are symmetric, the algorithm also adds u_3 to X_1. Thus, at the end of these two iterations $X_3 = X_1 \cup \{u_2, u_3\} = \{u_2, u_3\}, Y_3 = \{u_2, u_3, u_4, u_5\}$.

3. Finally, the algorithm considers u_4 and u_5. These two iterations are also independent so we consider only u_4. Adding u_4 to X_3 does not increase the value of X_3. Also removing u_4 from Y_3 does not increase the value of Y_3. Thus, the algorithm adds u_4 to X_3. Since u_4 and u_5 are symmetric, the algorithm also adds u_5 to X_3. Thus, we get $X_5 = Y_5 = \{u_2, u_3, u_4, u_5\}$. □

4.2 A Randomized Linear Time $(1/2)$-Approximation Algorithm for `Unconstrained Submodular Maximization`

In this section we present a randomized algorithm achieving a tight $(1/2)$-approximation for USM, and by doing so prove Theorem 4.0.9. Algorithm 4 presented in Section 4.1 compared the marginal profits a_i and b_i. Based on this comparison the algorithm made a greedy deterministic decision whether to include or exclude u_i from its output. The random algorithm we present next makes a "smoother" decision, based on the values a_i and b_i. In each step, it randomly chooses whether to include or exclude u_i from the output with probability that is based on the values a_i and b_i. A formal description of the algorithm appears as Algorithm 5.

Algorithm 5: RandomizedUSM$(f, \mathcal{N})$

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

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

The rest of this section is devoted to proving that Algorithm 5 provides an approximation guarantee of $1/2$ to USM. Let us start the analysis of Algorithm 5 with the introduction of some notation. Notice that for every $1 \leq i \leq n$, X_i and Y_i are random variables denoting the sets of elements in the two solutions generated by the algorithm at the end of the i-th iteration. As in Section 4.1, let us define the following random variable: $OPT_i \triangleq (OPT \cup X_i) \cap Y_i$. Note that as before, $X_0 = \emptyset$, $Y_0 = \mathcal{N}$ and $OPT_0 = OPT$. Moreover, the following always happens: $OPT_n = X_n = Y_n$. The proof

idea is similar to that of the deterministic algorithm in Section 4.1. We consider the sequence $\mathbb{E}[f(OPT_0)], \ldots, \mathbb{E}[f(OPT_n)]$. This sequence starts with $f(OPT)$ and ends with the expected value of the algorithm's output. The following lemma upper bounds the loss of two consecutive elements in the sequence. Formally, $\mathbb{E}[f(OPT_{i-1}) - f(OPT_i)]$ is upper bounded by the *average* expected change in the value of the two solutions maintained by the algorithm, *i.e.*, $1/2 \cdot \mathbb{E}\left[(f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})\right]$.

Lemma 4.2.1. *For every $1 \leq i \leq n$,*

$$\mathbb{E}[f(OPT_{i-1}) - f(OPT_i)] \leq \frac{1}{2} \cdot \mathbb{E}\left[(f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})\right] \ , \tag{4.2}$$

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

Before proving Lemma 4.2.1, let us show that Theorem 4.0.9 follows from it.

Proof of Theorem 4.0.9. Summing up Lemma 4.2.1 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 the definitions of OPT_0 and OPT_n, we obtain $\mathbb{E}[f(X_n)] = \mathbb{E}[f(Y_n)] \geq f(OPT)/2$. □

It is left to prove Lemma 4.2.1.

Proof of Lemma 4.2.1. Notice that it suffices to prove Inequality (4.2) conditioned on any event of the form $X_{i-1} = S_{i-1}$, where $S_{i-1} \subseteq \{u_1, \ldots, u_{i-1}\}$ and the probability that $X_{i-1} = S_{i-1}$ is non-zero. Hence, fix such an event corresponding to a set S_{i-1}. The rest of the proof implicitly assumes everything is conditioned on this event. Notice that due to the conditioning, the following quantities become constants:

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

Moreover, by Lemma 4.1.1, $a_i + b_i \geq 0$. Thus, it cannot be that both a_i, b_i are strictly less than zero. Hence, we only need to consider the following 3 cases:

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

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

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

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

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

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

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

$$\begin{aligned}
&\mathbb{E}[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})] = \\
&\quad \frac{a_i}{a_i + b_i} \cdot [f(X_{i-1} + u_i) - f(X_{i-1}) + f(Y_{i-1}) - f(Y_{i-1})] + \\
&\quad \frac{b_i}{a_i + b_i} \cdot [f(X_{i-1}) - f(X_{i-1}) + f(Y_{i-1} - u_i) - f(Y_{i-1})] \\
&= \frac{a_i}{a_i + b_i} \cdot [f(X_{i-1} + u_i) - f(X_{i-1})] + \frac{b_i}{a_i + b_i} \cdot [f(Y_{i-1} - u_i) - f(Y_{i-1})] = \frac{a_i^2 + b_i^2}{a_i + b_i} \ .
\end{aligned} \tag{4.3}$$

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

$$\begin{aligned}
\mathbb{E}[f(OPT_{i-1}) - f(OPT_i)] &= \frac{a_i}{a_i + b_i} \cdot [f(OPT_{i-1}) - f(OPT_{i-1} + u_i)] \\
&\quad + \frac{b_i}{a_i + b_i} \cdot [f(OPT_{i-1}) - f(OPT_{i-1} - u_i)] \\
&\leq \frac{a_i b_i}{a_i + b_i} \ .
\end{aligned} \tag{4.4}$$

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

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

If $u_i \in OPT_{i-1}$ then the first term of the left hand side of Inequality (4.4) equals zero, and we also have $X_{i-1} \subseteq ((OPT \cup X_{i-1}) \cap Y_{i-1}) - u_i = OPT_{i-1} - u_i$. Thus, by submodularity,

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

Plugging (4.3) and (4.4) into Inequality (4.2), we get the following:

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

which can be easily verified. □

4.3 A Tight $(1/2)$-Approximation for USM using Fractional Values

In Section 4.2 we presented Algorithm 5 which is a randomized optimal algorithm for USM. In this section we present Algorithm 6, which is the continuous counterpart of the Algorithm 5. This algorithm achieves the same approximation ratio (up to low order terms), but keeps a fractional inner state.

We abuse notations both in the description of the algorithm and in its analysis, and unify a set with its characteristic vector. As usual, we assume that we have an oracle access to the multilinear extension F. If this is not the case, then the value of F can be approximated arbitrarily well using sampling.

Algorithm 6: MultilinearUSM($f, \mathcal{N}$)

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

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

The main difference between Algorithms 5 and 6 is that Algorithm 5 chooses each element with some probability, whereas Algorithm 6 assigns a fractional value to the element. This requires the following modifications to the algorithm:

- The sets $X_i, Y_i \subseteq 2^{\mathcal{N}}$ are replaced by the vectors $x_i, y_i \in [0,1]^{\mathcal{N}}$.
- Algorithm 6 uses the multilinear extension F instead of the original submodular function f.

Theorem 4.3.1. *If one has an oracle access to F, Algorithm 6 is a $1/2$-approximation algorithm for USM.*

The rest of this section is devoted to proving Theorem 4.3.1. Similarly to Section 4.2, define $OPT_i \triangleq (OPT \vee x_i) \wedge y_i$. Examine the sequence $F(OPT_0), \ldots, F(OPT_n)$. Notice that $OPT_0 = OPT$, *i.e.*, the sequence starts with the value of an optimal solution, and that $OPT_n = x_n = y_n$, *i.e.*, the sequence ends at a fractional point whose value is the expected value of the algorithm's output. The following lemma upper bounds the loss of two consecutive elements in the sequence. Formally, $F(OPT_{i-1}) - F(OPT_i)$ is upper bounded by the *average* change in the value of the two solutions maintained by the algorithm, *i.e.*, $1/2 \cdot [F(x_i) - F(x_{i-1}) + F(y_i) - F(y_{i-1})]$.

Lemma 4.3.2. *For every $1 \leq i \leq n$, $F(OPT_{i-1}) - F(OPT_i) \leq \frac{1}{2} \cdot [F(x_i) - F(x_{i-1}) + F(y_i) - F(y_{i-1})]$.*

Before proving Lemma 4.3.2, let us show that Theorem 4.3.1 follows from it.

Proof of Theorem 4.3.1. Summing up Lemma 4.3.2 for every $1 \leq i \leq n$ gives:

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

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

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

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

It is left to prove Lemma 4.3.2.

Proof of Lemma 4.3.2. By Lemma 4.1.1, $a_i + b_i \geq 0$, therefore, it cannot be that both a_i, b_i are strictly less than zero. Thus, we have 3 cases to consider.

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

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

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

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

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

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

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

$$\begin{aligned} F(x_i) - F(x_{i-1}) &= \left[\frac{a_i}{a_i + b_i} \cdot F(x_{i-1} \vee \{u_i\}) + \frac{b_i}{a_i + b_i} \cdot F(x_{i-1})\right] - F(x_{i-1}) \qquad (4.5) \\ &= \frac{a_i}{a_i + b_i} \cdot [F(x_{i-1} \vee \{u_i\}) - F(x_{i-1})] = \frac{a_i^2}{a_i + b_i} \ . \end{aligned}$$

A similar argument shows that:

$$F(y_i) - F(y_{i-1}) = \frac{b_i^2}{a_i + b_i} \ . \qquad (4.6)$$

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

$$\begin{aligned} F(OPT_{i-1}) - F(OPT_i) &= F(OPT_{i-1}) - F(OPT_{i-1} \vee \frac{a_i}{a_i + b_i} \cdot \{u_i\}) \qquad (4.7) \\ &= F(OPT_{i-1}) - \left[\frac{a_i}{a_i + b_i} \cdot F(OPT_{i-1} \vee \{u_i\}) \right. \\ &\quad \left. + \frac{b_i}{a_i + b_i} \cdot F(OPT_{i-1})\right] \\ &= \frac{a_i}{a_i + b_i} \cdot [F(OPT_{i-1}) - F(OPT_{i-1} \vee \{u_i\})] \\ &\leq \frac{a_i}{a_i + b_i} \cdot [F(y_{i-1} - \{u_i\}) - F(y_{i-1})] = \frac{a_i b_i}{a_i + b_i} \ . \end{aligned}$$

The inequality follows from the submodularity of f since,

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

(note again that in this case $(y_{i-1})_{u_i} = 1$ and $(OPT_{i-1})_{u_i} = 0$). Plugging (4.5), (4.6) and (4.7) into the inequality that we need to prove, we get the following:

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

which can be easily verified. □

4.4 Linear Time Approximations for `Submodular Max-SAT` and `Submodular Welfare` with 2 Players

In this section we build upon ideas from the previous sections to obtain linear time tight (3/4)-approximation algorithms for both the `Submodular Max-SAT` (SSAT) and the `Submodular Welfare` (SW) with 2 players problems. Tight approximations for these problems were also given in Sections 3.3.4 and 3.3.5 above. However, the algorithms we present here, in addition to having optimal approximation ratios, also run in linear time.

4.4.1 A Linear Time Tight (3/4)-Approximation for `Submodular Max-SAT`

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

Like in Section 3.3.4, we extend the notion of assignments, and think of an extended assignment ϕ' which is a relation $\phi' \subseteq \mathcal{N} \times \{0, 1\}$. That is, the assignment ϕ' can assign up to 2 truth values to each variable. A clause C is satisfied by an (extended) assignment ϕ', if there exists a positive literal in the clause which is assigned to truth value 1, or there exists a negative literal in the clause which is assigned a truth value 0. Note again that it might happen that some variable is assigned both 0 and 1, or no value at all. Note also, that an assignment is a feasible solution to the original problem if and only if it assigns exactly one truth value to every variable of $\mathcal{N}$. Let $C(\phi')$ be the set of clauses satisfied by ϕ'. We define $g : \mathcal{N} \times \{0, 1\} \to \mathbb{R}^+$ using $g(\phi') \triangleq f(C(\phi'))$. Using the above notation, it is possible to restate SSAT as the problem of maximizing the function g over the set of feasible assignments. Recall the following lemma from Section 3.3.4.

Lemma 3.3.7. *The function g is a normalized monotone submodular function.*

The algorithm we design for SSAT conducts n iterations. It maintains at each iteration $1 \leq i \leq n$ two assignments X_i and Y_i which always satisfy: $X_i \subseteq Y_i$. Initially, $X_0 = \emptyset$ assigns no truth values to the variables, and $Y_0 = \mathcal{N} \times \{0, 1\}$ assigns both truth values to all variables. The algorithm considers the variables in an arbitrary order $u_1, u_2, \ldots, u_n$. For every variable u_i, the algorithm evaluates the marginal profit from assigning it only the truth value 0 in both assignments, and assigning it only the truth value 1 in both assignments. Based on these marginal values, the algorithm makes a random decision on the truth value assigned to u_i. After the algorithm considers a variable u_i, both assignments X_i and Y_i agree on a single truth value for u_i. Thus, when the algorithm terminates both assignments are identical and feasible. A formal statement of the algorithm appears as Algorithm 7.

Algorithm 7: RandomizedSSAT(f, Ψ)

$X_0 \leftarrow \emptyset$, $Y_0 \leftarrow \mathcal{N} \times \{0,1\}$.
for $i = 1$ *to* n **do**
  $a_{i,0} \leftarrow g(X_{i-1} + (u_i, 0)) - g(X_{i-1}), a_{i,1} \leftarrow g(X_{i-1} + (u_i, 1)) - g(X_{i-1})$.
  $b_{i,0} \leftarrow g(Y_{i-1} - (u_i, 0)) - g(Y_{i-1}), b_{i,1} \leftarrow g(Y_{i-1} - (u_i, 1)) - g(Y_{i-1})$.
  $s_{i,0} \leftarrow \max\{a_{i,0} + b_{i,1}, 0\}$, $s_{i,1} \leftarrow \max\{a_{i,1} + b_{i,0}, 0\}$.
  with probability $s_{i,0}/(s_{i,0} + s_{i,1})^*$ **do**: $X_i \leftarrow X_{i-1} + (u_i, 0)$, $Y_i \leftarrow Y_{i-1} - (u_i, 1)$.
  else (with the compliment probability $s_{i,1}/(s_{i,0} + s_{i,1})$) **do**:
  $X_i \leftarrow X_{i-1} + (u_i, 1)$, $Y_i \leftarrow Y_{i-1} - (u_i, 0)$.
return X_n (or equivalently Y_n).

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

The proof of the Theorem 4.0.10 uses similar ideas to previous proofs in this paper, however, unlike for the previous algorithms, here, the fact that the algorithm runs in linear time requires a proof. The source of the difficulty is that we have an oracle access to f, but use the function g in the algorithm. Therefore, we need to prove that we may implement all the oracle queries to g that are conducted during the execution of the algorithm in linear time. As a first step we state the following useful lemma.

Lemma 4.4.1. *For every* $1 \leq i \leq n$,

$$\mathbb{E}[g(OPT_{i-1}) - g(OPT_i)] \leq \frac{1}{2} \cdot \mathbb{E}\left[(g(X_i) - g(X_{i-1}) + g(Y_i) - g(Y_{i-1})\right] \ , \tag{4.8}$$

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

Before proving Lemma 4.4.1, let us show that Theorem 4.0.10 follows from it.

Proof of Theorem 4.0.10. The proof has two parts: bounding the approximation ratio of the algorithm, and analyzing its running time.

Proof of the approximation ratio of the algorithm: Summing up Lemma 4.4.1 for every $1 \leq i \leq n$ we get,

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

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

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

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

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

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

Proof of the linear running time of the algorithm: We explain here how to answer all oracle queries of the algorithm of the forms $g(X_{i-1})$ and $g(X_{i-1}+(u_i,v))$ in linear time. Using an analogous idea, one can also answer all oracle queries of the of the forms $g(Y_{i-1})$ and $g(Y_{i-1}-(u_i,v))$ in linear time.

The algorithm first pre-process the clauses, and creates for every variable a list of the clauses that are satisfied if the truth value of the variable is set to 0, and another list with the clauses that are satisfied if the truth value of the variable is set to 1. Notice that this can be done in time linear in the length of the formula Ψ.

Additionally, the algorithm maintains a set $C \subset \mathcal{C}$ of clauses satisfied by X_{i-1}. Initially this set is empty. To evaluate the queries of the forms $g(X_{i-1})$ and $g(X_{i-1}+(u_i,v))$, the algorithm has to respond to three types of events.

- A query of the form $g(X_{i-1})$ is answered by making an oracle query $f(C)$.
- A query of the form $g(X_{i-1}+(u_i,v))$ is answered using a three steps process.
  1. Adding all the clauses from the list of clauses satisfied by $\{u_i,v\}$ to C. The clauses that are new to C are kept in a side list L.
  2. Making an oracle query $f(C)$.
  3. Removing the clauses of L from C.
- When the assignment X_{i-1} is replaced with a new assignment $X_{i-1}+(u_i,v)$, the algorithm adds to C all clauses satisfied by $\{u_i,v\}$.

Notice that each list is used for responding to at most 2 events. Hence, the time required to respond to all events is $O(n)$ plus time proportional to the total length of the lists (which is linear in the length of Ψ).

Remark: In order for the above implementation to work efficiently, we must maintain the set C using a data structure which supports performing the following operations in constant time: "adding an element", "removing an element" and "checking the existence of an element". One possible such data structure is, *e.g.*, an array. □

It is left to prove Lemma 4.4.1.

Proof. Notice that it suffices to prove Inequality (4.8) conditioned on any event of the form $X_{i-1}=S_{i-1}$, where $S_{i-1}\subseteq\{(u_1,*),(u_2,*),\ldots,(u_{i-1},*)\}$ and the probability that $X_{i-1}=S_{i-1}$ is non-zero (note that according to the algorithm's definition X_{i-1} contains exactly a single element of the form $(u_j,*)$ for every $1\leq j\leq i-1$). Hence, fix such an event corresponding to a S_{i-1}. The rest of the proof implicitly assumes everything is conditioned on this event. Notice that due to the conditioning, the following quantities become constants:

- $Y_{i-1}=S_{i-1}\cup\{(u_i,0),(u_i,1),\ldots,(u_n,0),(u_n,1)\}$.
- $OPT_{i-1}=(OPT\cup X_{i-1})\cap Y_{i-1}$.
- $a_{i,0}$, $a_{i,1}$, $b_{i,0}$, and $b_{i,1}$.

Moreover, by Lemma 4.1.1, $a_{i,0}+b_{i,0}\geq 0$ and $a_{i,1}+b_{i,1}\geq 0$. Thus, $s_{i,0}+s_{i,1}\geq 0$, and it cannot be that both $s_{i,0},s_{i,1}$ are strictly less than zero. Hence, we only need to prove the lemma for the following 3 cases:

Case 1 ($s_{i,0} \geq 0$ and $s_{i,1} \leq 0$): In this case $\frac{s_{i,0}}{s_{i,0}+s_{i,1}} = 1$, and the following always happen: $Y_i = Y_{i-1} - (u_i, 1)$ and $X_i \leftarrow X_{i-1} + (u_i, 0)$. Hence,

$$g(X_i) - g(X_{i-1}) + g(Y_i) - g(Y_{i-1}) = s_{i,0} = a_{i,0} + b_{i,1} \ .$$

By our definition $OPT_i = (OPT \cup X_i) \cap Y_i = OPT_{i-1} + (u_i, 0) - (u_i, 1)$. Thus, we are left to prove that:

$$g(OPT_{i-1}) - g(OPT_{i-1} + (u_i, 0) - (u_i, 1)) \leq \frac{1}{2} \cdot [g(X_i) - g(X_{i-1}) + g(Y_i) - g(Y_{i-1})] = s_{i,0}/2 \ .$$

There are two case. If $(u_i, 0) \in OPT, (u_i, 1) \notin OPT$, then the left hand side of the last expression is 0, which is clearly no larger than the non-negative $s_{i,0}/2$. If $(u_i, 0) \notin OPT, (u_i, 1) \in OPT$, then,

$$\begin{aligned} g(OPT_{i-1}) - g(OPT_{i-1} + (u_i, 0) - (u_i, 1)) &= (g(OPT_{i-1}) - g(OPT_{i-1} + (u_i, 0)) \\ &\quad + (g(OPT_{i-1} + (u_i, 0) - g(OPT_{i-1} + (u_i, 0) - (u_i, 1))) \\ &\leq g(X_{i-1} + (u_i, 1)) - g(X_{i-1}) + g(Y_{i-1} - (u_i, 0)) - g(Y_{i-1}) = s_{i,1} \leq 0 \leq s_{i,0}/2 \ . \end{aligned}$$

The first inequality follows from submodularity since $X_{i-1} \subseteq OPT_{i-1} + (u_i, 0) - (u_i, 1)$, $OPT_{i-1} \subseteq Y_{i-1} - (u_i, 0)$.

Case 2 ($s_{i,0} < 0$ and $s_{i,1} \geq 0$): This case is analogous to the previous one, and therefore, we omit its proof.

Case 3 ($s_{i,0} \geq 0$ and $s_{i,1} > 0$): In this case $s_{i,0} = a_{i,0} + b_{i,1}, s_{i,1} = a_{i,1} + b_{i,0}$ and so,

$$\begin{aligned} &\mathbb{E}[g(X_i) - g(X_{i-1}) + g(Y_i) - g(Y_{i-1})] \\ &\quad = \frac{s_{i,0}}{s_{i,0} + s_{i,1}} \cdot [g(X_{i-1} + (u_i, 0)) - g(X_{i-1}) + g(Y_{i-1} - (u_i, 1)) - g(Y_{i-1})] \\ &\qquad + \frac{s_{i,1}}{s_{i,0} + s_{i,1}} \cdot [g(X_{i-1} + (u_i, 1)) - g(X_{i-1}) + g(Y_{i-1} - (u_i, 0)) - g(Y_{i-1})] \\ &\quad = \frac{s_{i,0}^2 + s_{i,1}^2}{s_{i,0} + s_{i,1}} \ . \end{aligned} \tag{4.9}$$

Next, we upper bound $\mathbb{E}[g(OPT_{i-1}) - g(OPT_i)]$. As $OPT_i = (OPT \cup X_i) \cap Y_i$ and given the random choices of the algorithm that modify X_{i-1} and Y_{i-1}, we get:

$$\begin{aligned} \mathbb{E}[g(OPT_{i-1}) - g(OPT_i)] &= \frac{s_{i,0}}{s_{i,0} + s_{i,1}} \cdot [g(OPT_{i-1}) - g(OPT_{i-1} + (u_i, 0)\} - (u_i, 1))] \\ &\quad + \frac{s_{i,1}}{s_{i,0} + s_{i,1}} \cdot [g(OPT_{i-1}) - g(OPT_{i-1} + (u_i, 1) - (u_i, 0))] \\ &\leq \frac{s_{i,0} s_{i,1}}{s_{i,0} + s_{i,1}} \ . \end{aligned} \tag{4.10}$$

The final inequality follows by considering two cases. Note first that $\{(u_i, 0), (u_i, 1)\} \subseteq Y_{i-1}$ and $\{(u_i, 0), (u_i, 1)\} \cap X_{i-1} = \varnothing$. If $(u_i, 0) \notin OPT_{i-1}$ and $(u_i, 1) \in OPT_{i-1}$ then the second term of the left hand side of the last inequality is zero. Moreover, $OPT_{i-1} = ((OPT \cup X_{i-1}) \cap Y_{i-1}) \subseteq Y_{i-1} - (u_i, 0)$. Thus, by submodularity,

$$\begin{aligned} &g(OPT_{i-1}) - g(OPT_{i-1} + (u_i, 0) - (u_i, 1)) \\ &\qquad \leq g(X_{i-1} + (u_i, 1)) - g(X_{i-1}) + g(Y_{i-1} - (u_i, 0)) - g(Y_{i-1}) = s_{i,1} \ . \end{aligned}$$

The other case is analogous. Plugging (4.9) and (4.10) into Inequality (4.8), we get the following:

$$\frac{s_{i,0}s_{i,1}}{s_{i,0}+s_{i,1}} \leq \frac{1}{2} \cdot \left(\frac{s_{i,0}^2 + s_{i,1}^2}{s_{i,0}+s_{i,1}} \right) \ ,$$

which can be easily verified. □

A Note on `Max-SAT`: The well known `Max-SAT` problem is in fact a special case of SSAT where f is a linear function. We note that Algorithm 7 can be applied to `Max-SAT` in order to achieve a (3/4)-approximation in linear time, however, this is not immediate. This result is summarized in the following theorem.

Theorem 4.4.2. *Algorithm 7 has a linear time implementation for instances of `Max-SAT`.*

Proof. Consider the way oracle queries of g are answered in the proof of Theorem 4.0.10. The only use of queries to f made by this proof is in order to evaluate $f(C)$ - the total weight of the clauses in the set C. In order to avoid these queries, one can use a counter $w(C)$ which holds the total weight of the clauses in C. Such a counter can be updated in constant time whenever a clause is either added or removed from C. Using this counter, one has an immediate access to the value of $f(C)$ at all times, in $O(1)$ time. □

4.4.2 A Linear Time Tight (3/4)-Approximation for `Submodular Welfare` with 2 Players

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

We give below two different short proofs of Theorem 4.0.11 via reductions to SSAT and USM, respectively. The second proof is due to Vondrák [79].

Proof of Theorem 4.0.11. We provide here two proofs.

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

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

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

Moreover, queries of f can be answered in constant time using the following technique. We track for every subset $C \subseteq \mathcal{C}$ in the algorithm the subsets $C \cap P$ and $C \cap (\mathcal{C} \setminus P)$. For Algorithm 7 this can be done without effecting its running time. Then, whenever the value of $f(C)$ is queried, answering it simply requires making two oracle queries: $f_1(C \cap P)$ and $f_2(C \cap (\mathcal{C} \setminus P))$.

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

Observe that the function g is a submodular function, but unlike f_1 and f_2, it is possibly non-monotone. Moreover, we can answer queries to the function g using only two oracle queries to f_1 and f_2.[1] Thus, we obtain an instance of USM. We apply Algorithm 5 to this instance. Using the analysis of Algorithm 5, as is, provides only a $(1/2)$-approximation for our problem. However, by noticing that $g(\varnothing) + g(\mathcal{N}) \geq f_1(\mathcal{N}) + f_2(\mathcal{N}) \geq g(OPT)$, and plugging this into the analysis, the claimed $(3/4)$-approximation is obtained. □

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

Chapter 5

k-Exchange Systems

Recall that a set system is a pair $(\mathcal{N}, \mathcal{I})$, where $\mathcal{N}$ is, as usual, a ground set, and $\mathcal{I} \subseteq 2^{\mathcal{N}}$ is a collection of subsets of $\mathcal{N}$. The collection $\mathcal{I}$ must obey the following two properties:

- Non-empty: $\mathcal{I} \neq \emptyset$.
- Monotone: If $A \subseteq B \in \mathcal{I}$, then $A \in \mathcal{I}$.

If $S \in \mathcal{I}$, we say that S is *independent*. A minimally dependent set is called *circuit*. A matroid is a set system with the following extra property:

- Matroid Exchange: If $A, B \in \mathcal{I}$ and $|A| < |B|$, then there exists an element $u \in B$ for which $A + u \in \mathcal{I}$.

In this section we consider other classes of set systems. The most common classes of set systems form a hierarchy:

$$k\text{-intersection} \subseteq k\text{-circuit bound} \subseteq k\text{-extendible} \subseteq k\text{-system} .$$

A set system belongs to the k-intersection class if it is the intersection of k matroids defined over a common ground set. The other classes extend this definition. The class of k-circuit bound contains all set systems in which adding a single element to an independent set creates at most k circuits. It is known that in a matroid, adding an element to an independent set creates at most a single circuit [74]. Therefore, adding an element to an independent set in a k-intersection system closes at most k circuits, one per matroid. This shows why the k-circuit bound class generalizes k-intersection. The class of k-extendible, intuitively, captures all set systems in which adding an element to an independent set requires throwing away at most k other elements from the set (in order to keep it independent). This generalizes k-circuit bound because in k-circuit bound we need to throw at most one element per circuit closed (*i.e.*, up to k elements). Finally, the class of k-system contains all set systems in which for every set, not necessarily independent, the ratio of the sizes of the largest base of the set (a base is a maximal independent subset) to the smallest base of the set is at most k.

It is no surprise that for any given k, the best approximation factors known for finding an independent set maximizing a submodular function are known for k-intersection. Moreover, this is the case also for linear and monotone objective functions. An interesting question is whether it is possible to obtain improved approximations (*e.g.*, those known for k-intersection) to combinatorial optimization problems defined by set systems *not* belonging to the k-intersection class.

Algorithms maximizing submodular and linear objective functions for set systems within the hierarchy usually use one of two techniques. The first technique is to use a greedy rule. This approach was used by [16, 36] on the k-system class, resulting in approximation ratios of $(k+1)^{-1}$ (for monotone submodular objectives) and k^{-1} (for linear objectives). Of course, these approximation ratios extend to all classes in the hierarchy.

The second approach involves a more delicate local search argument. This technique has been used, *e.g.*, by Lee et al. [62] to improve upon the two previously mentioned results. However, it is known to work only for the k-intersection class, and the proof does not seem to be extendible to other classes in the hierarchy since it relays heavily on the structure of matroids intersection.

In this section we define the new k-exchange class. This class contains many interesting set systems which do not fall into the k-intersection class. Yet, we are able to show that the algorithm of Lee et al. [62] works for this class also, and provides for this class exactly the same approximation which it is proved to provide for the k-intersection class.

5.1 k-exchange: definition and relations with other set systems

Here is the promised definition of k-exchange.

Definition 5.1.1 (k-exchange system)**.** *An set system $(\mathcal{N}, \mathcal{I})$ is a k-exchange system if, for all S and T in $\mathcal{I}$, there exists a multiset $Y = \{Y_u \subseteq S \setminus T \mid u \in T \setminus S\}$ such that:*

(K1) $|Y_u| \leq k$ *for each* $u \in T \setminus S$.

(K2) *Every* $u' \in T \setminus S$ *appears in at most* k *sets of* Y.

(K3) *For all* $T' \subseteq T \setminus S$, $(S \setminus (\bigcup_{u \in T'} Y_u)) \cup T' \in \mathcal{I}$

Mestre shows that 1-extendible systems are matroids, and vice versa [66]. We can provide a similar motivation for k-exchange systems in terms of *strongly base orderable matroids* (which is derived from a work by Brualdi and Scrimger on exchange systems [13, 11, 12]).

Definition 5.1.2 (strongly base orderable matroid [12])**.** *A matroid M is strongly base orderable if for all bases S and T of M there exists a bijection $\pi : S \to T$ such that for all $S' \subseteq S$, $(T \setminus \pi(S')) \cup S'$ is a base.*

If we restrict S' to be a singleton set in this definition, *i.e.*, a replacement of a single pair of elements between the bases, then the above definition always holds due to a well-known result of Brualdi [10]. In a strongly base orderable matroid, the above definition holds for arbitrary S', *i.e.*, multiple replacements can be performed simultaneously, while keeping the independence of the bases. This simultaneous replacement property is exactly what we want for local search, as it allows us to extend the local analysis of single replacements to the larger sets of replacements required by our algorithms.

The following theorem is easily obtained by equating Y_u in Definition 5.1.1 with the singleton set $\{\pi(u)\}$, where π is as in Definition 5.1.2.

Theorem 5.1.1. *An independence system $(\mathcal{N}, \mathcal{I})$ is a strongly base orderable matroid if and only if it is a 1-exchange system.*

One can easily check that the graphic matroid of the cycle C_4 is not strongly base orderable, and so provides a proof for the following corollary.

Corollary 5.1.2. *There exists a matroid which is not a* 1*-exchange system.*

In other words, the above corollary implies that k-intersection is not a subset of k-exchange. This naturally raises the question whether the opposite is true, *i.e.*, whether k-exchange is a subset of k-intersection. Before answering this question, let us recall that k-intersection is a subset of k-circuit bound, formally defined next.

Definition 5.1.3. *A set system $(\mathcal{N}, \mathcal{I})$ belongs to the class of k-circuit bound if for every $S \in \mathcal{I}$ and every $u \in \mathcal{N} \setminus S$, $S + u$ contains at most k circuits (minimally dependent set).*

Theorem 5.1.3. *There exists a* 2*-exchange set system which is not* 2*-circuit bound.*

Proof. Consider the ground set $\mathcal{N} = \{a, b, c, d\}$, and the following collection of maximal independent sets: $\mathcal{I}' = \{ \{a, b\}, \{a, c\}, \{a, d\}, \{b, c, d\} \}$. Let the collection $\mathcal{I}$ of independent sets be all the subsets of sets in $\mathcal{I}'$. Note that $\{b, c, d\}$ is independent, yet $\{a\} \cup \{b, c, d\}$ contains three circuits: $\{a, b, c\}$, $\{a, b, d\}$ and $\{a, c, d\}$. Therefore, $(\mathcal{N}, \mathcal{I})$ is not 2-circuit bound.

We are left to prove that $(\mathcal{N}, \mathcal{I})$ is 2-exchange. Clearly, this is a set system since it is non-empty and monotone. Hence, we are left to prove the properties introduced by Definition 5.1.1. Notice that it is enough to check pairs of sets from $\mathcal{I}'$ because if two sets S and T satisfy the above properties, then so does any two subsets of them. Moreover, since elements b, c and d are symmetric, we have to consider only the following cases. For $S = \{a, b\}$ and $T = \{b, c, d\}$, choosing $Y_c = Y_d = \{a\}$ completes the case. For $S = \{b, c, d\}$ and $T = \{a, b\}$, choosing $Y_a = \{c, d\}$ completes the case. And lastly, for $S = \{a, b\}$ and $T = \{a, c\}$, choosing $Y_c = \{b\}$ completes this case also. We conclude that $(\mathcal{N}, \mathcal{I})$ is a 2-exchange set system. □

The above theorem proves that k-exchange is not a subset of k-circuit bound, and therefore, also not a subset of the smaller k-intersection. Conversely, the next theorem proves that the next class of set systems in the hierarchy, k-extendible, is general enough to capture all k-exchange systems.

Definition 5.1.4. *A set system $(\mathcal{N}, \mathcal{I})$ belongs to the class of k-extendible if for every $S \subset T \in \mathcal{I}$ and every $u \in \mathcal{N}$ such that $S + u \in \mathcal{I}$, there exists a set $Y \subseteq T \setminus S$ of size $|Y| \leq k$ such that $T \setminus Y + u \in \mathcal{I}$.*

Theorem 5.1.4. *Every k-exchange system is a k-extendible system.*

Proof. By definition, $(\mathcal{N}, \mathcal{I})$ is non empty and monotone. Let $S, T \in \mathcal{I}$ and $u \in \mathcal{N}$ such that $T + u \in \mathcal{I}$. Assume $u \notin S$, otherwise the proof is finished. By property 3 of Definition 5.1.1 for S and $T + u$, there exists $Y_u \subseteq S \setminus (T + u) = S \setminus T$ such that $|Y_u| \leq k$ and $S \setminus Y_u + u \in \mathcal{I}$ (choose $T' = \{u\}$). □

Figure 5.1 depicts the relations between the different set system classes.

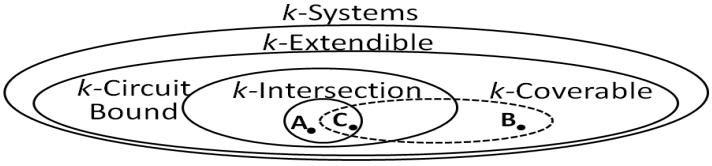

Figure 5.1: Exact characterization of the k-exchange class within the standard set system hierarchy. A - Corollary 5.1.2, B - Theorem 5.1.3, C - bipartite matching.

5.2 Maximizing a Monotone Submodular Function

The problem of optimizing a normalized monotone submodular function over the intersection of k matroids was considered by Fisher et al. [36] who gave a greedy algorithm with an approximation factor of $1/(k+1)$. Fisher et al. [36] also state that their proof extends to the more general class of k-system using the outline of [50] (the extended proof is explicitly given by [16]). Lee et al. [62], showed that, for intersection of k matroids, a natural local search algorithm improves over the above result, and achieves $1/(k+\delta)$-approximation, for any constant $\delta > 0$. Their analysis make heavy use of the exchange properties of the underlaying combinatorial structure, and therefore, cannot be extended to the more general classes of k-circuit bound and k-extendible. For these classes, the current best known approximation is still the $1/(k+1)$ of [36].

In this section we show that the local search algorithms of [62] provides the same approximation ratio of $1/(k+\delta)$ also for maximizing a normalized monotone submodular function over a k-exchange system. However, unlike the analysis of [62] which uses matroid intersection techniques, our analysis goes through a counting argument applied to an auxiliary graph. Finding a unified analysis that works for both types of set systems is an interesting open question. It should be noted that the time complexity of the algorithm we analyze is exponential in k, thus, k is assumed to be a constant. Indeed, k is a small constant in our applications (refer to Table 5.1 for the exact values of k).

Let us present the algorithm we analyze. The following directed graph is used by the algorithm:

Definition 5.2.1. *Given a k-intersection/k-exchange set system $(\mathcal{N},\mathcal{I})$, $S,T \in \mathcal{I}$, $\varepsilon > 0$ and $p \in \mathbb{N}$, T is* (ε,p)-reachable *from S if the following conditions are satisfied:*

1. $|T \setminus S| \leq p$.
2. $|S \setminus T| \leq (k-1)p + 1$.
3. $f(T) \geq (1 + \varepsilon/|\mathcal{N}|)\, f(S)$.

Definition 5.2.2. *Given a k-exchange system $(\mathcal{N},\mathcal{I})$, $\varepsilon > 0$ and $p \in \mathbb{N}$, the* (ε,p)-graph *of $(\mathcal{N},\mathcal{I})$ is a directed graph $\mathcal{G} = (\mathcal{I},\mathcal{E})$ where $(S \to T) \in \mathcal{E}$ if and only if T is (ε,p)-reachable from S.*

Algorithm 8 starts from a vertex in $\mathcal{G}$ and tours the graph arbitrarily until finding a sink vertex S. The algorithm than outputs S.

Algorithm 8: Local-Search-k-Exchange$((\mathcal{N},\mathcal{I}),\varepsilon,p)$

```
   // Starting Point
1  u ← argmax {f({u}) | u ∈ N}.
2  S ← {u}.
   // Touring G
3  Let G = (I, E) be the (ε,p)-graph of (N, I).
4  while ∃(S → T) ∈ E do
5  |  S ← T.
6  Output S.
```

It is important to note that the starting point of S is an independent set. Otherwise $u \in \mathcal{N}$ chosen in step 1 does not belong to any independent set (recall that $(\mathcal{N},\mathcal{I})$ is monotone), and therefore, can be removed from $(\mathcal{N},\mathcal{I})$.

The size of of $\mathcal{G}$ might be exponential since $|\mathcal{I}|$ might be exponential. However, one can show that the algorithm terminates in polynomial time.

Lemma 5.2.1. *For any constants k, $0 < \varepsilon < |\mathcal{N}|$ and $p \in \mathbb{N}$, Algorithm 8 terminates in polynomial time.*

Proof. First, consider the time needed to perform a single step while touring $\mathcal{G}$. In such a single step, the goal is to determine whether there is an edge $(S \to T) \in \mathcal{E}$. For any given $S \in \mathcal{I}$, there are at most $|\mathcal{N}|^{O(kp)}$ possible $T \in \mathcal{I}$ for which T might be (ε, p)-reachable from S. Since k and p are constants, the time needed to perform a single step is polynomial in n.

Second, we bound the number of steps while touring $\mathcal{G}$. Denote by S_i the vertex in $\mathcal{G}$ reached after conducting i steps and by S_0 the initial vertex. Let M be the number of steps the algorithm makes. Since the value of the current vertex improves in each step (condition 3 in Definition 5.2.1), we get that: $f(S_i) \geq (1 + \varepsilon/|\mathcal{N}|)^i f(S_0)$. On the other hand, by the choice of S_0 and the submodularity of f, for any $S \in \mathcal{I}$: $f(S) \leq |\mathcal{N}| f(S_0)$. Combining both bounds we can conclude that:

$$(1 + \varepsilon/|\mathcal{N}|)^M f(S_0) \leq |\mathcal{N}| f(S_0) \quad \Rightarrow \quad M \leq \frac{\ln |\mathcal{N}|}{\ln (1 + \varepsilon/|\mathcal{N}|)} = O\left(\frac{|\mathcal{N}| \log |\mathcal{N}|}{\varepsilon}\right) \ .$$

Therefore, since ε is a constant, the number of steps the algorithm makes while touring $\mathcal{G}$ is polynomial in $|\mathcal{N}|$. □

The following is the main result of [62].

Theorem 5.2.2 (From [62])**.** *Algorithm 8 provides $(k + k\varepsilon + 1/p)^{-1}$-approximation for maximizing a monotone submodular function over k-intersection.*

Using the right choice of ε and p, this approximation guarantee can be made $(k+\delta)^{-1}$ for an arbitrary small δ. The analyze we present next shows that the same approximation holds also for k-exchange set systems. The following theorem is a key ingredient of the analysis. Its use is restricted to the analysis only. No actual construction of $\mathcal{P}(G, k, h)$ is needed.

Theorem 5.2.3. *Let G be an undirected graph whose maximum degree is at most $k \geq 2$. Then, for every $h \in \mathbb{N}$ there exists a multiset $\mathcal{P}(G, k, h)$ of simple paths in G and a labeling $\ell : V \times \mathcal{P}(G, k, h) \to \{\emptyset, 1, 2, \ldots, h\}$ such that:*

1. *For every $P \in \mathcal{P}(G, k, h)$, the labeling ℓ of the nodes of P is consecutive and increasing with labels from $\{1, 2, \ldots, h\}$. Vertices not in P receive label $\emptyset$.*
2. *For every $P \in \mathcal{P}(G, k, h)$ and v in P, if $deg_G(v) = k$ and $\ell(v, P) \notin \{1, h\}$, then at least two of the neighbors of v are in P.*
3. *For every $v \in V$ and label $i \in \{1, 2, \ldots, h\}$, there are $n(k, h) = k \cdot (k-1)^{h-2}$ paths $P \in \mathcal{P}(G, k, h)$ for which $\ell(v, P) = i$.*

Note for condition 2, v might be an end vertex of a path P, but still have a label different from 1 and h. This might happen since paths might contain less than h vertices and start with a label different from 1 or end with a label different from h.

Let us provide some intuition as to why the construction of $\mathcal{P}(G, k, h)$ is possible. Assume that the degree of every vertex in G is exactly k and that G's girth is at least h. Construct the multiset $\mathcal{P}(G, k, h)$ in the following way. From every vertex $u \in V$, choose

all possible paths starting at v and containing exactly h vertices. Number the vertices of these paths consecutively, starting from 1 up to h. First, note that all these paths are simple since the girth of G is at least h and all paths contain exactly h vertices. Second, the number of paths starting from u is: $n(k,h) = k \cdot (k-1)^{h-2}$, since all vertices have degree of exactly k. Third, the number of times each label is given in the graph is exactly $n \cdot n(k,h)$. Since the number of vertices at distance i, $1 \leq i \leq h$, from each vertex u is identical, the labels are distributed equally among the vertices. Thus, the number of paths in which a given vertex u appears with a given label, is exactly $n(k,h)$. This concludes the proof of the theorem in case G has the above properties. For a proof that works for general graphs, see Section 5.2.1.

Let $S, T \in \mathcal{I}$ be two arbitrary independent sets. Construct the following bipartite graph $G_{S,T} = (S \setminus T, T \setminus S, E)$, where $E = \{(u, u') \mid u \in T \setminus S, u' \in Y_u\}$. The following observation implies that Theorem 5.2.3 can be applied to $G_{S,T}$.

Observation 5.2.4. *For any k-exchange system $(\mathcal{N}, \mathcal{I})$ and $S, T \in \mathcal{I}$, the maximum degree in $G_{S,T}$ is at most k.*

Proof. Follows from the first two properties of Definition 5.1.1. □

Choosing $h = 2p$ gives a multiset $\mathcal{P}(G_{S,T}, k, 2p)$ of simple paths in $G_{S,T}$ and a labeling $\ell : (S \triangle T) \times \mathcal{P}(G_{S,T}, k, 2p) \to \{\emptyset, 1, 2, \ldots, 2p\}$ with all the properties guaranteed by Theorem 5.2.3. We use $\mathcal{P}(G_{S,T}, k, 2p)$ to construct a new multiset $\mathcal{P}'$ of subsets of vertices of $G_{S,T}$. Intuitively, $\mathcal{P}'$ is the collection of all paths in $\mathcal{P}(G_{S,T}, k, 2p)$ with an extra "padding" of vertices from S that surround P, excluding "paths" composed of a single vertex from S.

Formally, let $P \in \mathcal{P}(G_{S,T}, k, 2p)$. If P contains at least one vertex from T[1], then add to $\mathcal{P}'$ the set $P \cup \delta_{S \setminus T}(P)$, where $\delta_{S \setminus T}(P) \subseteq S \setminus T$ is the set of vertices in $S \setminus T$ which neighbor at least one vertex of P. If P does not contain any vertex of T, do not add anything to $\mathcal{P}'$.

The following Lemma is an application of the properties Theorem 5.2.3 give to $\mathcal{P}$.

Lemma 5.2.5. *Every vertex $u \in T \setminus S$ appears in $2p \cdot n(k, 2p)$ sets of $\mathcal{P}'$, and every vertex $u' \in S \setminus T$ appears in at most $2\left((k-1)p + 1\right) \cdot n(k, 2p)$ sets of $\mathcal{P}'$.*

Proof. By property 3 of Theorem 5.2.3, every vertex $u \in T \setminus S$ appears in $n(k, 2p)$ paths of $\mathcal{P}(G_{S,T}, k, 2p)$ for every possible label. Since there are $2p$ possible labels, the number of appearances is exactly $2p \cdot n(k, 2p)$. In the creation of $\mathcal{P}'$ from $\mathcal{P}(G_{S,T}, k, 2p)$, no vertex from $T \setminus S$ is added or removed from any path $P \in \mathcal{P}(G_{S,T}, k, 2p)$, thus, this is also the number of appearances of every vertex $u \in T \setminus S$ in $\mathcal{P}'$. This completes the proof of the first part of the lemma.

Let $u' \in S \setminus T$. By the construction of $\mathcal{P}'$, a set in $\mathcal{P}'$ that contains u' must contain a vertex $u \in T \setminus S$ where $(u, u') \in E$ (u is a neighbor of u' in $G_{S,T}$). Every such neighboring vertex u, by the first part of the lemma, appears in exactly $2p \cdot n(k, 2p)$ sets in $\mathcal{P}'$. Therefore, the number of appearances of u' in sets of $\mathcal{P}'$ is at most: $deg_{G_{S,T}}(u') \cdot 2p \cdot n(k, 2p) \leq 2pk \cdot n(k, 2p)$ (here we use the fact that $deg_{G_{S,T}}(u') \leq k$ by Observation 5.2.4). We can improve this bound by observing that some of the sets in $\mathcal{P}'$ are counted more than once. Consider $P \in \mathcal{P}(G_{S,T}, k, 2p)$ where $u' \in P$. If $\ell(u', P) \neq \{1, 2p\}$, by property 2 of Theorem 5.2.3, u' has at least two neighbors in $T \setminus S$ which belong to P itself. Hence, $P \cup \delta_{S \setminus T}(P) \in \mathcal{P}'$ should be counted only once while in the above counting it was counted at least twice. The number of such $P \in \mathcal{P}(G_{S,T}, k, 2p)$ is exactly $2(p-1) \cdot n(k, 2p)$ (by

[1]Note that this implies that this vertex is from $T \setminus S$ since $G_{S,T}$ does not contain vertices from $S \cap T$.

property 3 of Theorem 5.2.3). Removing the double counting from the bound, we can conclude that for every $u' \in S \setminus T$, the number of sets in $\mathcal{P}'$ it appears in is at most:

$$2pk \cdot n(k,2p) - 2(p-1) \cdot n(k,2p) = 2\left((k-1)p+1\right) \cdot n(k,2p) \ . \qquad \square$$

Note: In the proof of Theorem 5.2.9 we need each vertex $u' \in S \setminus T$ to appear in *exactly* $2\left((k-1)p+1\right) \cdot n(k,2p)$ sets in $\mathcal{P}'$. This can be achieved by adding "dummy" sets to $\mathcal{P}'$ containing u' alone.

Given two independent sets $S, T \in \mathcal{I}$, the following lemma shows that the symmetric difference of $S \triangle P'$ is also an independent set, for any $P' \in \mathcal{P}'$.

Lemma 5.2.6. *Let $(\mathcal{N}, \mathcal{I})$ be a k-exchange system, $S, T \in \mathcal{I}$ and $P' \in \mathcal{P}'$. Then $S \triangle P' \in \mathcal{I}$.*

Proof. Let $u \in P'$ where $u \in T \setminus S$. Note that P' contains all neighbors of u in $G_{S,T}$ (since $P' \in \mathcal{P}'$). By the definition of $G_{S,T}$ these neighbors are exactly the set Y_u. Therefore,

$$S \triangle P' \subseteq \left(S \setminus \left(\cup_{e \in P'} Y_e\right)\right) \cup \left(P' \cap \left(T \setminus S\right)\right) \ .$$

Since $S, T \in \mathcal{I}$ and $(\mathcal{N}, \mathcal{I})$ is a k-exchange system, by Definition 5.1.1, the right hand side of the above expression is an independent set. By monotonicity, we conclude that $S \triangle P' \in \mathcal{I}$. $\square$

At this point we need the following two technical lemmata from [62]:

Lemma 5.2.7 (Lemma 1.1 in [62])**.** *Let f be a non-negative submodular function of $\mathcal{N}$. Let $S' \subseteq S \subseteq \mathcal{N}$ and let $\{T_\ell\}_{\ell=1}^t$ be a collection of subsets of $S \setminus S'$ such that every elements of $S \setminus S'$ appears in exactly k of these subsets. Then, $\sum_{\ell=1}^t \left[f(S) - f(S \setminus T_\ell)\right] \leq k\left(f(S) - f(S')\right)$.*

Lemma 5.2.8 (Lemma 1.2 in [62])**.** *Let f be a non-negative submodular function of $\mathcal{N}$. Let $S \subseteq \mathcal{N}$, $C \subseteq \mathcal{N}$ and let $\{T_\ell\}_{\ell=1}^t$ be a collection of subsets of $C \setminus S$ such that every elements of $C \setminus S$ appears in exactly k of these subsets. Then, $\sum_{\ell=1}^t \left[f(S \cup T_\ell) - f(S)\right] \geq k\left(f(S \cup C) - f(S)\right)$.*

We are now ready to state our main theorem. Denote by S_{ALG} the output of Algorithm 8.

Theorem 5.2.9. *For every $T \in \mathcal{I}$ and every submodular f:*

$$f(S_{ALG} \cup T) + \left(k - 1 + \frac{1}{p}\right) \cdot f(S_{ALG} \cap T) \leq \left(k + \frac{1}{p} + k\varepsilon\right) \cdot f(S_{ALG}) \ .$$

Let us give some intuition as to how the above theorem is proved. Given S_{ALG} and an independent set $T \in \mathcal{I}$, Lemma 5.2.6 states that $S_{ALG} \triangle P'$ is independent for every $P' \in \mathcal{P}'$. Additionally, note that the construction of $\mathcal{P}'$ from $G\left(G_{S_{ALG},T}, k, 2p\right)$ implies that all size conditions of Definition 5.2.1 are met. Thus, $S_{ALG} \triangle P'$ is (ε, p)-reachable from S_{ALG}, i.e., $(S_{ALG} \to S_{ALG} \triangle P') \in \mathcal{E}$. However, since Algorithm 8 terminated with S_{ALG}, it must be the case that S_{ALG} is approximately "locally optimal", and therefore, approximately $f(S_{ALG}) \geq f(S_{ALG} \triangle P')$.

Every element of $\mathcal{P}'$ represents a part of the difference between S_{ALG} and T. The above discussion implies that none of these elements improves the value of S_{ALG} significantly, and therefore, we expect that all of them together will not make a significant improvement also; or in other words we expect T itself to provide only an insignificant improvement over S_{ALG}. Applying now the quantitative bounds of Lemma 5.2.5 to $\mathcal{P}'$, along with some additional observations, is what is needed to make this argument formal, and complete the proof of Theorem 5.2.9.

Proof of Theorem 5.2.9. Let $P' \in \mathcal{P}'$ where $\mathcal{P}'$ is the set generated from $\mathcal{P}(G_{S_{ALG},T}, k, 2p)$. Note that P' contains only vertices from $S_{ALG} \triangle T$, by the construction of $\mathcal{P}'$. Additionally, $(S_{ALG} \triangle P') \setminus S_{ALG} = P' \setminus S_{ALG}$, and by the construction of P': $|P' \setminus S_{ALG}| \leq p$. This implies that $|(S_{ALG} \triangle P') \setminus S_{ALG}| \leq p$. Also, note that $S_{ALG} \setminus (S_{ALG} \triangle P') = S_{ALG} \cap P'$, and by the construction of P': $|S_{ALG} \cap P'| \leq (k-1)p + 1$. This implies that $|S_{ALG} \setminus (S_{ALG} \triangle P')| \leq (k-1)p + 1$.

By Lemma 5.2.6, $S_{ALG} \triangle P' \in \mathcal{I}$. If $f(S_{ALG} \triangle P') \geq (1 + \varepsilon/|\mathcal{N}|)\, f(S_{ALG})$, then by Definition 5.2.1, $S_{ALG} \triangle P'$ is (ε, p)-reachable from S_{ALG}, contradicting the fact that Algorithm 8 stopped with S_{ALG}. Thus, we can conclude:

$$f(S_{ALG} \triangle P') < (1 + \varepsilon/|\mathcal{N}|)\, f(S_{ALG}) \ . \tag{5.1}$$

By submodularity of f, the fact that $S_{ALG} \setminus P' \subseteq S_{ALG} \triangle P'$ and the fact that all vertices in $S_{ALG} \cap P'$ do not belong to either $S_{ALG} \setminus P'$ or $S_{ALG} \triangle P'$, we get:

$$f(S_{ALG} \cup P') - f(S_{ALG} \triangle P') \leq f(S_{ALG}) - f(S_{ALG} \setminus P') \ . \tag{5.2}$$

Adding Inequalities 5.1 and 5.2 gives:

$$f(S_{ALG} \cup P') - (1 + \varepsilon/|\mathcal{N}|)\, f(S_{ALG}) \leq f(S_{ALG}) - f(S_{ALG} \setminus P') \ . \tag{5.3}$$

Inequality 5.3 holds for every $P' \in \mathcal{P}'$. Summing over all such sets yields:

$$\begin{aligned} \sum_{P' \in \mathcal{P}'} \left[f(S_{ALG} \cup P') - f(S_{ALG}) \right] - \frac{\varepsilon \cdot |\mathcal{P}'|}{|\mathcal{N}|} f(S_{ALG}) \\ \leq \sum_{P' \in \mathcal{P}'} \left[f(S_{ALG}) - f(S_{ALG} \setminus P') \right] \ . \end{aligned} \tag{5.4}$$

Let us focus first on the right hand side of Inequality 5.4. Since any given P' contains only vertices from $S_{ALG} \triangle T$, the right hand side can be rewritten as:

$$\sum_{P' \in \mathcal{P}'} \left[f(S_{ALG}) - f(S_{ALG} \setminus \left(P' \cap (S_{ALG} \setminus T) \right)) \right] \ .$$

By Lemma 5.2.5 (and the note after it), each vertex of $S_{ALG} \setminus T$ belongs to exactly $n(k, 2p) \cdot 2\left((k-1)p + 1\right)$ sets in $\mathcal{P}'$. Thus, applying Lemma 5.2.7 gives us that:

$$\begin{aligned} \sum_{P' \in \mathcal{P}'} [f(S_{ALG}) - f(S_{ALG} \setminus \left(P' \cap (S_{ALG} \setminus T) \right))] \\ \leq 2\left((k-1)p + 1\right) n(k, 2p)\left(f(S_{ALG}) - f(S_{ALG} \cap T) \right) \ . \end{aligned} \tag{5.5}$$

Let us focus now on the summation part of the left hand side of Inequality 5.4. Since any given P' contains only vertices from $S_{ALG} \triangle T$ the summation part of the left hand side can be rewritten as:

$$\sum_{P' \in \mathcal{P}'} \left[f(S_{ALG} \cup \left(P' \cap (T \setminus S_{ALG}) \right)) - f(S_{ALG}) \right].$$

By Lemma 5.2.5, each vertex in $T \setminus S_{ALG}$ appears in exactly $2p \cdot n(k, 2p)$ sets in $\mathcal{P}'$. Hence, applying Lemma 5.2.8 gives us that:

$$\begin{aligned} \sum_{P' \in \mathcal{P}'} [f(S_{ALG} \cup \left(P' \cap (T \setminus S_{ALG}) \right)) - f(S_{ALG})] \\ \geq 2p \cdot n(k, 2p)\left(f(S_{ALG} \cup T) - f(S_{ALG}) \right) \ . \end{aligned} \tag{5.6}$$

Plugging Inequalities 5.5 and 5.6 into 5.4 results in:

$$2p \cdot n(k,2p)(f(S_{ALG} \cup T) - f(S_{ALG})) - \frac{\varepsilon \cdot |\mathcal{P}'|}{|\mathcal{N}|} f(S_{ALG}) \leq \\ 2\left((k-1)p+1\right) n(k,2p) \left(f(S_{ALG}) - f(S_{ALG} \cap T)\right) \ .$$

Rearranging terms, we get:

$$f(S_{ALG} \cup T) + \left(k - 1 + \frac{1}{p}\right) f(S_{ALG} \cap T) \\ \leq \left(k + \frac{1}{p}\right) f(S_{ALG}) + \frac{\varepsilon |\mathcal{P}'|}{2p \cdot n(k,2p)|\mathcal{N}|} f(S_{ALG}) \ . \tag{5.7}$$

Observe that every set $P' \in \mathcal{P}'$ contains at least a single vertex from $G_{S_{ALG},T}$. Lemma 5.2.5 states that every vertex in $G_{S_{ALG},T}$ appears in exactly $2p \cdot n(k,2p)$ or $2\left((k-1)p+1\right) \cdot n(k,2p)$ sets of $\mathcal{P}'$ (the exact number is determined by whether the vertex is in $T \setminus S_{ALG}$ or $S_{ALG} \setminus T$). Therefore, in the worst case we can conclude that $|\mathcal{P}'| \leq |S_{ALG} \triangle T| \cdot 2n(k,2p) \max\{p, (k-1)p+1\} \leq 2|\mathcal{N}|n(k,2p)kp$. Plugging this bound into Inequality 5.7 completes the proof. □

The approximation ratio of Algorithm 8 for maximizing a normalized monotone submodular function subject to k-exchange constraint now follows.

Theorem 5.2.10. *Given a k-exchange set system $(\mathcal{N}, \mathcal{I})$ and a normalized monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, for any constant $\delta > 0$ there is an approximation of $1/(k+\delta)$.*

Proof. Choose constants $p \in \mathbb{N}$ and $0 < \varepsilon < |\mathcal{N}|$ such that $1/p + k\varepsilon \leq \delta$.

$$\begin{aligned} f(S_{OPT}) &\overset{(1)}{\leq} f(S_{OPT} \cup S_{ALG}) \\ &\overset{(2)}{\leq} f(S_{OPT} \cup S_{ALG}) + \left(k - 1 + \frac{1}{p}\right) \cdot f(S_{OPT} \cap S_{ALG}) \\ &\overset{(3)}{\leq} \left(k + \frac{1}{p} + k\varepsilon\right) \cdot f(S_{ALG}) \ . \end{aligned}$$

Inequality (1) is by the monotonicity of f. Inequality (2) is by the fact that f is non-negative (since it is normalized and monotone). Inequality (3) is by Theorem 5.2.9 used with $T = S_{OPT}$. □

5.2.1 Proof of Theorem 5.2.3

We described above a simple construction proving Theorem 5.2.3 for graphs of high girth and uniform degree. For general graphs, we have to "correct" this construction. Since general graphs might have vertices with a degree lower than k or simple cycles shorter than h, we add "dummy" vertices to the start and end of paths whenever a vertex has a low degree or we encounter a short simple cycle. We present a procedure for constructing the multiset $\mathcal{P}(G,k,h)$. This procedure uses the following subroutine we call EXPAND.

The procedure EXPAND gets a prefix, P, of a path and expands it recursively. The boolean flag EXPAND gets is 1 if and only if the prefix P is composed of a single vertex *and* its previous vertex in the path was a dummy vertex. Lines $2-3$ of the procedure finish the path if it is fully expanded. Line 4 deals with the case that the degree of v is smaller than

Algorithm 9: EXPAND(P, flag)

```
1  Denote by v the last vertex in P and by ℓ(v, P) its label.
2  if ℓ(v, P) = h then
3      Add P to 𝒫(G, k, h) and terminate.
4  Add P to 𝒫(G, k, h) with multiplicity of (k − deg_G(v) − flag) · (k − 1)^{h−ℓ(v,P)−1}.
5  if P contains only v then
6      Let u be a dummy vertex.
7  else
8      Let u be the predecessor of v in P.
9  for every w ≠ u neighbor of v do
10     if w ∉ P then
11         ℓ(w, P) ← ℓ(v, P) + 1.
12         EXPAND(P − w, 0).
13     else
14         Add P to 𝒫(G, k, h) with multiplicity of (k − 1)^{h−ℓ(v,P)−1}.
```

k by adding P to $\mathcal{P}(G,k,h)$ the appropriate number of times. The term $k-deg_G(v)-flag$ in the number of calls counts the number of neighboring dummy vertices of v which are unused yet. The term $(k-1)^{h-\ell(v,P)-1}$ counts the number of appendices needed to fully expand the prefix P. The loop starting at line 9 goes over all real (non-dummy) neighbors of v which differ from u. If such a neighbor is not in P yet (line 10) it is added to the prefix P with the next consecutive label and a recursive call to EXPAND is made. In case such a neighbor is already in P, *i.e.*, there is a simple cycle shorter than h, P is continued with dummy vertices. The term $(k-1)^{h-\ell(v,P)-1}$ counts the number of appendices needed to fully expand the prefix P.

The procedure CONSTRUCT construct the multiset $\mathcal{P}(G,k,h)$ using EXPAND. The call to EXPAND in line 3 of CONSTRUCT is for creating all paths that start with a real (non-dummy) vertex labeled 1. Lines $4-5$ consider the case that the path started in a dummy vertex and reached v. The term $k-deg_G(v)$ in the number of calls is for the case that the degree of v is smaller than k, and the term $(k-1)^{i-2}$ in the number of calls is for the number of possible prefixes before reaching v. Lines $6-9$ deal with paths that start with dummy vertices and enter short simple cycles. The number of calls in line 9 is $(k-1)^{i-1}$ since this is the number of possible prefixes to P.

Lemma 5.2.11. *Consider a call to EXPAND with a prefix P and denote P's last vertex by v. If $\ell(v,P)=1$ then $k\cdot(k-1)^{h-2}$ paths are added to $\mathcal{P}(G,k,h)$, otherwise $(k-1)^{h-\ell(v,P)}$ paths are added to $\mathcal{P}(G,k,h)$.*

Proof. The proof is by reverse induction on $\ell(v,P)$.
Base: The base case is that $\ell(v,P)=h$. In this case EXPAND adds P only once to $\mathcal{P}(G,k,h)$ (line 2 of EXPAND). This completes the base of the reverse induction.
Step: Consider the case where $\ell(v,P)\notin\{1,h\}$. It must be the case that either $flag=0$ and P contains more than a single vertex, or $flag=1$ and P contains a single vertex (this is true since a call with $flag=1$ and P containing more than a single vertex is never made, and a call with $flag=0$ and P containing a single vertex is made only in line 3 of CONSTRUCT in which case $\ell(v,P)=1$). Let us consider the above two possible cases:

1. $flag=0$ **and P contains more than a single vertex:** Line 4 of EXPAND adds

Algorithm 10: CONSTRUCT$(G = (V, E), k, h)$

```
1  P(G,k,h) ← ∅.
2  for every v ∈ V do
3  |  EXPAND(P which contains only v with label 1,0).
4  |  for i = 2 to h do
5  |  |  Call EXPAND(P which contains only v with label i,1),
   |  |  (k − deg_G(v)) · (k − 1)^{i−2} times.
6  for every simple cycle C of length at most h do
7  |  for every simple path P of |C| vertices obtained from C by removing a single
   |  edge and choosing a start vertex do
8  |  |  for i = 1 to h − |C| do
9  |  |  |  Call EXPAND(P where its first node is assigned label i + 1, 0), (k − 1)^{i−1}
   |  |  |  times.
10 Return P(G,k,h).
```

$(k - deg_G(v)) \cdot (k-1)^{h-\ell(v,P)-1}$ paths to $\mathcal{P}(G,k,h)$. The loop in lines $8-14$ of EXPAND executes $deg_G(v) - 1$ iterations, one for each neighbor $w \neq u$ of v. In the case $w \notin P$, we apply the induction hypothesis with $P - w$ and $\ell(w,P) = \ell(v,P)+1$ and get that $(k-1)^{h-\ell(v,P)-1}$ paths are added to $\mathcal{P}(G,k,h)$ (line 12 of EXPAND). Otherwise, $w \in P$ and $(k-1)^{h-\ell(v,P)-1}$ paths are added to $\mathcal{P}(G,k,h)$ (line 14 of EXPAND). Hence, in every iteration of the loop in lines $9-14$ exactly $(k-1)^{h-\ell(v,P)-1}$ paths are added to $\mathcal{P}(G,k,h)$. Therefore, we can conclude that the total number of paths added to $\mathcal{P}(G,k,h)$ in this case is:

$$(k - deg_G(v)) \cdot (k-1)^{h-\ell(v,P)-1} + (deg_G(v) - 1) \cdot (k-1)^{h-\ell(v,P)-1} = (k-1)^{h-\ell(v,P)} \ .$$

2. $flag = 1$ **and** P **contains a single vertex:** When comparing this case to the previous one, line 3 of EXPAND adds $(k-1)^{h-\ell(v,P)-1}$ *less* paths to $\mathcal{P}(G,k,h)$. However, the loop in lines $9-14$ has one more iteration, therefore adding $(k-1)^{h-\ell(v,P)-1}$ *more* paths to $\mathcal{P}(G,k,h)$. This completes the second case.

We are left with the case where $\ell(v,P) = 1$. In this case it must be that $flag = 0$ and P contains only v, since EXPAND must have been called from line 3 of CONSTRUCT. EXPAND adds $(k - deg_G(v)) \cdot (k-1)^{h-2}$ paths to $\mathcal{P}(G,k,h)$ in line 4. The loop in lines $9-14$ executes $deg_G(v)$ iterations, since u is a dummy vertex. As in the proof of the above two cases, in each iteration $(k-1)^{h-2}$ paths are added to $\mathcal{P}(G,k,h)$. Thus, we can conclude that the total number of paths added to $\mathcal{P}(G,k,h)$ when $\ell(v,P) = 1$ is:

$$(k - deg_G(v)) \cdot (k-1)^{h-2} + deg_G(v) \cdot (k-1)^{h-2} = k \cdot (k-1)^{h-2} \ . \qquad \square$$

As a corollary of Lemma 5.2.11 we get that every vertex v is assigned to label 1 in exactly $n(k,h) = k \cdot (k-1)^{h-2}$ paths in $\mathcal{P}(G,k,h)$. The reason this is true is that EXPAND is called with label 1 only in line 3 of CONSTRUCT. Such a call is made only once for each vertex.

Lemma 5.2.12. *For every vertex $v \in V$ and label $i \in \{2,3,\ldots,h\}$, the number of paths P in $\mathcal{P}(G,k,h)$ for which $\ell(v,P) = i$ is equal to the number of paths P in $\mathcal{P}(G,k,h)$ for which $\ell(v,P) = 1$.*

Proof. For a path P denote by $prefix_i(P)$ the prefix of the first i nodes of P (this includes also the labels assigned to the vertices). If P contains less than i vertices, set $prefix_i(P)$ to be P. Let $\mathcal{P}_1$ be the sub multiset of $\mathcal{P}(G,k,h)$ of all paths P where $\ell(v,P)=1$. Partition $\mathcal{P}_1$ according to $prefix_i$, *i.e.*, for every instance of a path $P \in \mathcal{P}_1$ assign it to the multiset $\mathcal{P}_{1,prefix_i(P)}$ of the partition.

For a path P denote by $prefix_v(P)$ the prefix of P until vertex v (this includes also the labels assigned to the vertices). Let $\mathcal{P}_i$ be the sub multiset of $\mathcal{P}(G,k,h)$ of all paths P where $\ell(v,P)=i$. Partition $\mathcal{P}_i$ according to $prefix_v$, i.e. for every instance of a path $P \in \mathcal{P}_i$ assign it to the multiset $\mathcal{P}_{i,prefix_v(P)}$ of the partition.

Note that in both partitions, the second index defining the set in the partition is a path Q containing at most i vertices with labels in $\{1,2,\ldots,i\}$. We prove that for any such Q: $|\mathcal{P}_{1,Q}| = |\mathcal{P}_{i,rev_i(Q)}|$, where $rev_i(Q)$ is the path Q reversed with each label j replaced with label $i-j+1$. Proving this completes the proof of the lemma.

First, consider the case where $|Q| = i-1$, *i.e.*, Q contains i vertices. All paths in the multiset $\mathcal{P}_{1,Q}$ contain v as the first vertex with label 1. Therefore, all of these paths must have originated from a *single* call to EXPAND in line 3 of CONSTRUCT. This call, by Lemma 5.2.11, added $(k-1)^{h-i}$ paths to $\mathcal{P}_{1,Q}$. The exact same argument works for $\mathcal{P}_{i,rev_i(Q)}$ since the label of the first vertex in $rev_i(Q)$ is also 1 (recall that $|Q| = i-1$). Thus, the proof is complete for the case where Q contains i vertices.

The other case to consider is where $|Q| < i-1$. Note that the same argument of the case where $|Q| = i-1$ does not work here, since the label of the first vertex in $rev_i(Q)$ is *not* 1. All paths in the multiset $\mathcal{P}_{1,Q}$ contain v as the first vertex with label 1. Therefore, all these paths must have originated from a *single* call to EXPAND in line 3 of CONSTRUCT. Moreover, since all these paths are in fact equal to Q, they all must have been added either in line 4 or 14 of a *single* EXPAND call. Denote by v_Q the last vertex in Q and by u_Q its predecessor (if one exists). In line 4, $(k - deg_G(v_Q)) \cdot (k-1)^{h-2-|Q|}$ paths are added to $\mathcal{P}_{1,Q}$. Every time line 14 is executed, $(k-1)^{h-2-|Q|}$ paths are added to $\mathcal{P}_{1,Q}$. The number of times line 14 is executed is the number of neighbors of v_Q which are not u_Q and appear in Q already, *i.e.*, $|(N(v_Q) - u_Q) \cap Q|$ times. Thus, we can conclude that the total number of paths in $\mathcal{P}_{1,Q}$ is:

$$(k - deg_G(v_Q) + |(N(v_Q) - u_Q) \cap Q|) \cdot (k-1)^{h-2-|Q|} \ .$$

Let us count the number of instances in the multiset $\mathcal{P}_{i,rev_i(Q)}$. The label of the first vertex in $rev_i(Q)$ is $i - |Q| > 1$, therefore, any path in $\mathcal{P}_{i,rev_i(Q)}$ originated from a call to EXPAND in which the first vertex was assigned a label greater than 1. There are two types for such calls. The first is in line 5 of CONSTRUCT and the second is in line 9 of CONSTRUCT. There are $(k - deg_G(v_Q)) \cdot (k-1)^{i-|Q|-2}$ calls of the first type. Each such call, by Lemma 5.2.11, adds $(k-1)^{h-i}$ paths. Thus, we can conclude that there are exactly $(k - deg_G(v_Q)) \cdot (k-1)^{h-|Q|-2}$ paths added by calls of the first type. Calls of the second type originate from a simple cycle C. There are exactly $|(N(v_Q) - u_Q) \cap Q|$ such cycles (one for every neighbor of v_Q which is not u_Q and is in Q). Since the label of the first vertex in a call of the second type is $i - |Q|$, there are $(k-1)^{i-|Q|-2}$ such calls. The label of the last vertex in $rev_i(Q)$ is i, therefore, Lemma 5.2.11 states that $(k-1)^{h-i}$ paths are added for any such call. Thus, we can conclude that the total number of paths added to $\mathcal{P}_{i,rev_i(Q)}$ is:

$$(k - deg_G(v_Q) + |(N(v_Q) - u_Q) \cap Q|) \cdot (k-1)^{h-2-|Q|} \ . \qquad \square$$

This completes the proofs of properties 1 and 3 of Theorem 5.2.3. We focus now on property 2. Let $P \in \mathcal{P}(G,k,h)$, and let $v \in P$ be a vertex with $deg_G(v) = k$ such that

$\ell(v,P) \notin \{1,h\}$. If v is not an end vertex of P the proof is completed. Otherwise, there are two cases. If v is an end vertex of P and minimizes $\ell(v,P)$ among all vertices in P, since $\ell(v,P) > 1$, it must be the case that P originated from a call to EXPAND in line 9 of CONSTRUCT (a call line line 5 of CONSTRUCT is not possible since $deg_G(v) = k$). In this case v has two neighbors in P because P contains all the vertices of the cycle C that originated this call to EXPAND. The other case is that v is an end vertex of P that maximizes $\ell(v,P)$ among all vertices in P. Since $\ell(v,P) < h$, it must be the case that some neighbors of v other than u, the previous node in P, is already in P (refer to the loop in lines $9-14$ in EXPAND), otherwise v would not be an end vertex of P.

5.3 Maximizing a Non-monotone Submodular Function

Gupta et al. [42] presented a technique for using monotone submodular optimization for non-monotone submodular problems. Their technique can be used, for example, for converting the greedy algorithm into an algorithm achieving an approximation ratio of $1/(3(k+2+1/k))$ for maximizing non-monotone submodular functions subject to general k-system. When optimizing over the intersection of k matroids, the current best result is $(k-1)/(k^2+\delta)$, and is due to [62]. In this section we give an approximation algorithm for maximizing a general non-negative submodular function over a k-exchange set system, assuming $k \geq 2$. The following algorithm uses Algorithm 8 as a procedure. This algorithm is based on Algorithm A of [61] and its analysis.

Algorithm 11: NON-MONOTONE-k-EXCHANGE$((\mathcal{N},\mathcal{I}),\varepsilon,p)$

```
1 N_1 ← N
2 for i = 1 to k do
3 |   S_i → LOCAL-SEARCH-k-EXCHANGE((N_i, I ∩ 2^{N_i}), ε, p)
4 |   N_{i+1} ← N_i \ S_i
5 Output the best set in {S_i}_{i=1}^k.
```

Recall that LOCAL-SEARCH-k-EXCHANGE is Algorithm 8. The set system on which Algorithm 8 is applied is the original set system $(\mathcal{N},\mathcal{I})$ constrained to a subset $\mathcal{N}_i$ of the elements.

Observe that the only place where the proof of Theorem 5.2.9 uses the monotonicity of f is for proving that it is non-negative. Hence, it holds also for general non-negative submodular functions. Let $S_{OPT,i} = S_{OPT} \cap \mathcal{N}_i$. Observe that $S_{OPT,i}$ is a feasible solution of the k-system on which Algorithm 8 is applied at the i^{th} iteration. Using Theorem 5.2.9 with $T = S_{OPT,i}$, we get

$$\left(k + p^{-1} + k\varepsilon\right) \cdot f(S_i) \geq f(S_i \cup S_{OPT,i}) + (k - 1 + p^{-1}) f(S_i \cap S_{OPT,i}) \ . \tag{5.8}$$

Let S_{ALG} denote the output of Algorithm 11. Since $f(S_{ALG}) \geq f(S_i)$ for every i, we can add the k instances of Equation 5.8 (for the k values of i), and get:

$$k\left(k + p^{-1} + k\varepsilon\right) \cdot f(S_{ALG}) \geq \left[\sum_{i=1}^{k} f(S_i \cup S_{OPT,i}) + (k - 1 + p^{-1}) \cdot \sum_{i=1}^{k} f(S_i \cap S_{OPT,i})\right] \ . \tag{5.9}$$

The following lemma simplifies the righthand side of the last inequality.

Lemma 5.3.1. *For every* $1 \leq \ell \leq k$,

$$k(k+p^{-1}+k\varepsilon)\cdot f(S_{ALG}) \geq (\ell-1)\cdot f(S_{OPT}) + f(\cup_{i=1}^{\ell} S_i \cup S_{OPT}) + \sum_{i=\ell+1}^{k} f(S_i \cup S_{OPT,i}) + \sum_{i=1}^{\ell-1}(k-1+p^{-1}-\ell+i)\cdot f(S_i \cap S_{OPT,i}) + (k-1+p^{-1})\cdot \sum_{i=\ell}^{k} f(S_i \cap S_{OPT,i}) \ .$$

Proof. We prove the lemma by induction. For $\ell = 1$, the claim that we need to prove becomes identical to Equation 5.9 once we observe that $S_{OPT} = S_{OPT,1}$. Assume now that the lemma holds for $\ell - 1$, let us prove it for ℓ. By the induction hypothesis we get:

$$k(k+p^{-1}+k\varepsilon)\cdot f(S_{ALG}) \geq (\ell-2)\cdot f(S_{OPT}) + f(\cup_{i=1}^{\ell-1} S_i \cup S_{OPT}) + \sum_{i=\ell}^{k} f(S_i \cup S_{OPT,i}) + \sum_{i=1}^{\ell-2}(k+p^{-1}-\ell+i)\cdot f(S_i \cap S_{OPT,i}) + (k-1+p^{-1})\cdot \sum_{i=\ell-1}^{k} f(S_1 \cap S_{OPT,i}) \ .$$

By the submodularity of f, we get the following inequality:

$$\begin{aligned} & f(\cup_{i=1}^{\ell-1} S_i \cup S_{OPT}) + f(S_i \cup S_{OPT,\ell}) + \sum_{i=1}^{\ell-1} f(S_i \cap S_{OPT,i}) \\ \geq\ & f(\cup_{i=1}^{\ell} S_i \cup S_{OPT}) + f(S_{OPT,\ell}) + \sum_{i=1}^{\ell-1} f(S_i \cap S_{OPT,i}) \\ \geq\ & f(\cup_{i=1}^{\ell} S_i \cup S_{OPT}) + f(S_{OPT}) \ . \end{aligned}$$

The proof of the lemma for ℓ follows by combining the last two inequalities. □

Theorem 5.3.2. *Given a k-exchange set system* $(\mathcal{N}, \mathcal{I})$ *and a non-negative submodular* $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, *for any constant* $\delta > 0$ *there is an approximation of* $(k-1)/(k^2+\delta)$ *for the problem of finding an independent set* S *maximizing* $f(S)$.

Proof. Algorithm 11 has a polynomial time complexity because Algorithm 8 does. In order to lower bound its performance, let us plug $\ell = k$ into Lemma 5.3.1. We get:

$$k(k+p^{-1}+k\varepsilon)\cdot f(S_{ALG}) \geq (k-1)\cdot f(S_{OPT}) + f(\cup_{i=1}^{k} S_i \cup S_{OPT}) + \sum_{i=1}^{k}(i-1+p^{-1})\cdot f(S_i \cap S_{OPT,i}) \ .$$

Since f is non-negative, this implies:

$$f(S_{ALG}) \geq \frac{k-1}{k(k+p^{-1}+k\varepsilon)}\cdot f(S_{OPT}) = \frac{k-1}{k^2+kp^{-1}+k^2\varepsilon}\cdot f(S_{OPT}) \ . \qquad \square$$

Choosing p and ε such that $kp^{-1} + k^2\varepsilon \leq \delta$ completes the proof of the theorem.

5.4 Maximizing a Linear Function

The previous sections analyzed Algorithm 8 for the case that the function f is submodular. Better results can be guaranteed when f is linear.

Theorem 5.4.1. *Given a k-exchange set system $(\mathcal{N}, \mathcal{I})$ and a linear $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, for any constant $\delta > 0$ there is an approximation of $1/(k - 1 + \delta)$ for the problem of finding an independent set S maximizing $f(S)$.*

Proof. Theorem 5.2.9 applies also to the case where f is linear. Choose constants $p \in \mathbb{N}$ and $0 < \varepsilon < |\mathcal{N}|$ such that $1/p + k\varepsilon \leq \delta$.

$$\begin{aligned} f(S_{OPT}) &\overset{(1)}{=} f(S_{OPT} \cup S_{ALG}) - f(S_{ALG} \setminus S_{OPT}) \\ &\overset{(2)}{\leq} \left(k + \frac{1}{p} + k\varepsilon\right) \cdot f(S_{ALG}) - \left(k - 1 + \frac{1}{p}\right) \cdot f(S_{ALG} \cap S_{OPT}) - f(S_{ALG} \setminus S_{OPT}) \\ &\overset{(3)}{\leq} \left(k + \frac{1}{p} + k\varepsilon\right) \cdot f(S_{ALG}) - (f(S_{ALG} \cap S_{OPT}) + f(S_{ALG} \setminus S_{OPT})) \\ &= \left(k - 1 + \frac{1}{p} + k\varepsilon\right) \cdot f(S_{ALG}) \ . \end{aligned}$$

Equality (1) is by the linearity of f. Inequality (2) is by Theorem 5.2.9 used with $T = S_{OPT}$. Inequality (3) holds since $k \geq 2$. □

The cardinality function is an interesting special case of a linear function. A variant of Algorithm 8 achieves a $2/(k + \delta)$-approximation for maximizing the cardinality function over a k-exchange set system, assuming $k \geq 3$. In the analysis of this algorithm we use the following theorem.

Theorem 5.4.2 (Theorem 1 of [49])**.** *Let p, k be integers with $k \geq 3$. Let $S_1, S_2, \ldots, S_m$ be subsets of a set $\mathcal{N}$ of size n such that the following holds:*

- *Each element of $\mathcal{N}$ is contained in at most k subsets among $S_1, S_2, \ldots, S_m$.*
- *For every $p' \leq p$, the union of every p' subsets among $S_1, S_2, \ldots, S_m$ is of size at least p'.*

Then, we have:

$$\begin{aligned} &(i) \quad \tfrac{m}{n} \leq \tfrac{k(k-1)^r - k}{2(k-1)^r - k} \quad \text{for } p = 2r - 1; \\ &(ii) \quad \tfrac{m}{n} \leq \tfrac{k(k-1)^r - 2}{2(k-1)^r - 2} \quad \text{for } p = 2r. \end{aligned}$$

Before we can describe the algorithm, we need the following definition.

Definition 5.4.1. *Given a k-exchange system $(\mathcal{N}, \mathcal{I})$, $S, T \in \mathcal{I}$ and $p \in \mathbb{N}$, T is p-*reachable *from S if the following conditions are satisfied:*

1. $|T \setminus S| \leq p$.
2. $|S \setminus T| \leq (k - 1)p + 1$.
3. $|T| > |S|$.

Remark: Notice that Definition 5.4.1 is different from Definition 5.2.1 only in property 3.

We can now define p-graphs in an analog way to the definition of (ε, p)-graphs in Definition 5.2.2. Replacing the (ε, p)-graph in Algorithm 8 with a p-graph, we get an algorithm for the cardinality objective function. Notice that the number of elements in S strictly increases in every iteration of the algorithm, hence, it makes at most n iteration and terminates in polynomial time.

Theorem 5.4.3. *For any constant $\delta > 0$, there exists an efficient algorithm for the cardinality objective function giving $2/(k+\delta)$-approximation.*

Proof. Choose p such that $\frac{2k^2}{(k-1)^{\lfloor (p+1)/2 \rfloor}} \leq \delta$. For every set $T' \subseteq S_{OPT} - S_{ALG}$ of at most p elements, consider the set $Y_{T'} = \cup_{e \in T'} Y_e$. Assume for the sake of contradiction that $|Y_{T'}| < |T'|$. Then $S_{ALG} - Y_{T'} \cup T$ has the following properties:

- It is independent because $(\mathcal{N}, \mathcal{I})$ is a k-exchange system.
- It is p-reachable from S (because $|S_{ALG} - Y_{T'} \cup T| = |S_{ALG}| - |Y_{T'}| + |T| > |S_{ALG}|$).

This contradicts the fact that S_{ALG} is the output of the algorithm. Hence, $|Y_{T'}| \geq |T'|$.

Now, observe that the set $S_{ALG} - S_{OPT}$ with the subsets $\{Y_e | e \in S_{OPT} - S_{ALG}\}$, k and p obeys all the conditions of Theorem 5.4.2. Therefore,

$$\frac{|S_{OPT}|}{|S_{ALG}|} \leq \frac{|S_{OPT} - S_{ALG}|}{|S_{ALG} - S_{OPT}|} \leq \frac{k(k-1)^{\lfloor (p+1)/2 \rfloor} - k}{2(k-1)^{\lfloor (p+1)/2 \rfloor} - k} \leq \frac{k}{2 - \frac{k}{(k-1)^{\lfloor (p+1)/2 \rfloor}}}.$$

Since $k/(k-1)^{\lfloor (p+1)/2 \rfloor} \leq 1.5$, it is easy to see that the above inequality implies:

$$\frac{|S_{OPT}|}{|S_{ALG}|} \leq \frac{k + \frac{2k^2}{(k-1)^{\lfloor (p+1)/2 \rfloor}}}{2} .$$

The theorem now follows from the choice of p. □

5.5 Applications

In this section we present a few examples of k-exchange systems. Table 5.1 summarizes the applications, the k for which they are k-exchange systems and the resulting approximation ratios. The table also summarizes the results proved above for general k-exchange systems.

Let us begin with our first application: strongly base orderable matroid k-parity.

Definition 5.5.1. *In the matroid k-parity problem, we are given a collection $\mathcal{N}$ of disjoint k-element subsets from a ground set $\mathcal{G}$ and a matroid $(\mathcal{G}, \mathcal{M})$ defined on the ground set. The goal is to find a collection S of subsets in $\mathcal{N}$ maximizing a function $f : \mathcal{N} \to \mathbb{R}^+$, subject to the constraint $\bigcup_{u \in S} u \in \mathcal{M}$.*

`Strongly base orderable matroid` k-`parity` (`SBO matroid` k-`parity`) is a special case of `matroid` k-`parity` where the given matroid is strongly base orderable. Any `matroid` k-`parity` problem can be expressed as the independence system $(\mathcal{N}, \mathcal{I})$ where $\mathcal{I} = \{S \subseteq \mathcal{N} \mid \bigcup_{u \in S} u \in \mathcal{M}\}$.

Theorem 5.5.1. *`SBO matroid` k-`parity` is a k-exchange system.*

Table 5.1: $k \geq 2$ is a constant and $\delta > 0$ is an arbitrary positve constant. f: NMS - normalized monotone submodular, NS - general non-negative submodular, L - linear, C - cardinality.

Maximization Problem	f	k	**This Thesis**	**Previous Result**	
k-exchange	NMS	k	$1/(k+\delta)$	$1/(k+1)$	[36]
	NS		$(k-1)/(k^2+\delta)$	$1/[3(k+2+1/k)]$	[42]
	L		$1/(k-1+\delta)$	$1/k$	[50]
	C[a]		$2/(k+\delta)$	$1/k$	[50]
Main Applications					
SBO matroid k-parity	NMS	k	$1/(k+\delta)$	$1/k$	[36]
	NS		$(k-1)/(k^2+\delta)$	$1/[3(k+2+1/k)]$	[42]
	L		$1/(k-1+\delta)$	$1/k$	[50]
b-Matching	NMS	2	$1/(2+\delta)$	$1/3$	[36]
	NS		$1/(4+\delta)$	$2/27$	[42]
k-Set Packing	NMS	k	$1/(k+\delta)$	$1/(k+1)$	[36]
	NS		$(k-1)/(k^2+\delta)$	$1/[3(k+2+1/k)]$	[42]
Additional Applications					
independent set in $(k+1)$-claw free graphs	NMS	k	$1/(k+1+\delta)$	$1/k$	[36]
	NS		$(k-1)/(k^2+\delta)$	$1/[3(k+2+1/k)]$	[42]
job interval selection with identical lengths	NMS	2	$1/(2+\delta)$	$1/3$	[66]
	NS	3	$2/(9+\delta)$	$1/16$	[42]
asymmetric traveling salesperson	NMS	3	$1/(3+\delta)$	$1/4$	[66]
	NS		$2/(9+\delta)$	$1/16$	[42]
frequency allocation on lines	NMS	3	$1/(3+\delta)$	$1/4$	[36]
	NS		$2/(9+\delta)$	$1/16$	[42]
	L		$1/(2+\delta)$	$1/3$	[50]
	C		$2/(3+\delta)$	$1-1/e$	[76]

[a] The result applies for $k \geq 3$.

Proof. It is easy to see that the above set system is non empty and monotone. Thus, we focus on the properties given by Definition 5.1.1. Consider two independent sets $S, T \in \mathcal{I}$. By definition, $\bigcup_{u\in S} u \in M$ and $\bigcup_{u\in T} u \in \mathcal{M}$. Let $\pi : \bigcup_{u\in S} u \to \bigcup_{u\in T} u$ be the bijection guaranteed by Definition 5.1.2. For any set $u \in S$ define $Y_u = \{u' \in T \mid u' \cap \pi(u) \neq \emptyset\}$. Recall that the sets in $\mathcal{N}$ are disjoint and contain at most k elements. Since π is a bijection, we must, therefore, have $|Y_u| \leq k$ for all $u \in S$. Moreover, each $u' \in T$ appears in at most k sets Y_u. Thus, Properties (K1) and (K2) of Definition 5.1.1 are satisfied. Consider a set $\mathcal{C} \subseteq S$, and let $S' = (S \setminus \bigcup_{u\in\mathcal{C}} Y_u) \cup \mathcal{C}$. From the definition of π we have $(\bigcup_{u\in S} u \setminus \pi(\bigcup_{u\in\mathcal{C}} u)) \cup \bigcup_{u\in\mathcal{C}} u \in \mathcal{M}$. Observe that $\bigcup_{u\in S'} u$ is a subset of this set, and thus, $\bigcup_{u\in S'} u \in \mathcal{M}$, implying $S' \in \mathcal{I}$. This complete the proof that property (K3) is also satisfied. □

SBO matroid k-parity generalizes our other two main applications: b-Matching and k-Set Packing. However, it is simpler to analyze them directly than to show the reductions. Consider first b-Matching.

Definition 5.5.2. *Given a graph $G = (V, E)$ and a function $b : V \to \mathbb{N}$, a* b-matching *is a set of edges $M \subseteq E$ such that for every node $|M \cap \delta(v)| \leq b(v)$ (where $\delta(v)$ is the*

set of edges hitting v). The maximum b-`Matching` problem is the problem of finding a b-matching M maximizing $f(M)$ for a given $f : 2^E \to \mathbb{R}^+$.

Lemma 5.5.2. *For any instance of b-`Matching`, the set of all feasible b-matchings is a 2-exchange set system.*

Proof. It is easy to see that the set system associated with b-`Matching` is non empty and monotone. Thus, we focus on the properties given by Definition 5.1.1. Let $S, T \subseteq E$ be two b-matchings of G. For every $v \in V$, number the edges of $(S \setminus T) \cap \delta(v)$ (respectively, $(T \setminus S) \cap \delta(v)$) from 1 to $|(S \setminus T) \cap \delta(v)|$ (respectively, $|(T \setminus S) \cap \delta(v)|$), denote the number edge $e \in (S \setminus T) \cap \delta(v)$ (respectively, $e \in (T \setminus S) \cap \delta(v)$) receives by $g_{S \setminus T}(v, e)$ (respectively, $g_{T \setminus S}(v, e)$). For every $e = (u, v) \in T \setminus S$ define $Y_e = \{e' \in S \setminus T \mid g_{T \setminus S}(u, e) = g_{S \setminus T}(u, e') \vee g_{T \setminus S}(v, e) = g_{S \setminus T}(v, e')\}$.

Notice that the numbering is unique among the edges hitting a single vertex. Thus, by definition $|Y_e| \leq 2$, and every $e' \in S \setminus T$ belongs to at most 2 such Y sets. Let $T' \subseteq T \setminus S$, and consider $M = S \setminus (\cup_{e \in T'} Y_e) \cup T'$. For every vertex $v \in V$, the numbers of the edges from M hitting v are distinct. Note that the numbers are in the range of 1 to at most $b(v) - |S \cap T|$, therefore there are at most $b(v)$ edges on total from M hitting v. We conclude that M is a b-matching of G. □

The last main application we consider is k-`Set Packing`.

Definition 5.5.3. *Given a hypergraph $H = (V, E)$ having only hyperedges of size at most k, A* set packing *is a subset of hyperedges $M \subseteq E$ such that every two hyperedges in M do not intersect. The maximum k-`Set Packing` problem is the problem of finding a set packing M maximizing $f(M)$ for a given $f : 2^E \to \mathbb{R}^+$.*

Lemma 5.5.3. *For any instance of k-`Set Packing`, the set of all set packings is a k-exchange set system.*

Proof. Set $\mathcal{N} = V$, and set $\mathcal{I}$ to be all feasible set packings of $H = (V, E)$. Clearly, the set system $(\mathcal{N}, \mathcal{I})$ is non-empty and monotone. Hence, we focus on the properties of Definition 5.1.1. Set $S, T \in \mathcal{I}$ and choose Y_e, for a given $e \in T \setminus S$, to be all hyperedges of $S \setminus T$ that intersect hyperedge e. Since e contains at most k vertices and $S \setminus T$ is a feasible set packing (*i.e.*, each vertex belongs to at most one hyperedge in $S \setminus T$), $|Y_e| \leq k$. Fix a hyperedge $e' \in S \setminus T$. Since T is a feasible set packing and e' contains at most k vertices, e' belongs to at most k of the Y sets. Fix $T' \subseteq T \setminus S$. We have to prove that $S \setminus (\cup_{e \in T'} Y_e) \cup T' \in \mathcal{I}$. Assume there is a vertex $v \in V$ that belongs to two hyperedges from $S \setminus (\cup_{e \in T'} Y_e) \cup T' \in \mathcal{I}$. Since $S, T \in \mathcal{I}$, it must be the case that there are two hyperedges, $e_1 \in S \setminus (\cup_{e \in T'} Y_e)$ and $e_2 \in T'$, that contain v. However, this is a contradiction to the construction of Y_{e_2}. We conclude that $(\mathcal{N}, \mathcal{I})$ is a k-exchange set system. □

In the rest of this section we consider the additional applications enlisted in Table 5.1.

Definition 5.5.4. *Given an undirected graph G, a k-claw is a set of $k + 1$ nodes in G which induces a star. If a graph contains no k-claw, we say it is a* k-claw free graph.

Lemma 5.5.4. *The independent sets of size at most k of a $(k + 1)$-claw free graph form a k-exchange system.*

Proof. Fix some $(k+1)$-claw free graph G, let $\mathcal{N}$ be the set of nodes of the graph, and let $\mathcal{I}$ be the collection of independent sets of size k or less. Since the subset of an independent set is also an independent set, the set system $(\mathcal{N}, \mathcal{I})$ is non-empty and monotone. Hence,

we focus on the properties of Definition 5.1.1. Consider two independent sets $S, T \in \mathcal{I}$. For every node $u \in T$, let us define Y_u to be the set of all neighbors of u in S. Notice that there are no edges between the nodes of Y_u because they all belong to the independent S. Hence, $|Y_u|$ must be of size at most k, otherwise, u together with any set of $k+1$ nodes from Y_u would have formed a claw. A symmetric argument also shows that the collection $\{Y_v | v \in Y_u\}$ is of size at most k for every $u \in T$.

Finally, for every subset $T' \subseteq T$, consider the set $A = S \setminus \cup_{u \in T'} Y_u \cup T'$. Since both S and T' are independent sets, any edge in A must connect nodes of S and T'. However, all neighbors of the nodes of T' belong to the set $\cup_{u \in T'} Y_u$, and therefore, are missing from A. Thus, the set A is independent. □

The `maximum job interval selection with identical lengths` problem (`JISIL`) is the next application we consider.

Definition 5.5.5. *Given a set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ of n jobs, where job J_i is associated with a release time r_i, a deadline d_i, and a common length $L \in \mathbb{N}$, a schedule $S \subseteq \mathcal{J}$ is feasible if every $J_i \in S$ is assigned an interval of length L inside $[r_i, d_i]$ and for any two jobs $J_i, J_r \in S$ the intervals assigned to them do not intersect. The* `maximum job interval selection with identical lengths` *(JISIL) problem is the problem of finding a feasible schedule S maximizing $f(S)$ for a given $f : 2^{\mathcal{J}} \to \mathbb{R}^+$.*

Lemma 5.5.5. *For any instance of JISIL, the set of feasible schedules is a 3-exchange set system. Moreover, if f is monotone and submodular, then this set is also 2-exchange.*

Proof. For every $J_i \in \mathcal{N}$, the ground set $\mathcal{N}$ contains all pairs: $(J_i, r_i), \ldots, (J_i, d_i - L)$. If a pair (J_i, t_i) belongs to a schedule S, it indicates that job J_i is scheduled at time t. Like in Definition 5.5.5, two pairs (J_i, t_i) and (J_r, t_r) intersect if $|t_i - t_r| < L$. A feasible schedule $S \subseteq \mathcal{N}$ is a set of pairs such that any two pairs in S do not intersect, and every job appears in at most one pair. We say that job $J_i \in \mathcal{J}$ is scheduled in S if S contains at least one pair of the form (J_i, t_i).

If f is monotone and submodular then we can drop the requirement that at most one pair of every job is in S. Instead, we define a new objective function $\bar{f}$ such that $\bar{f}(S)$ equals the value of f over the set of jobs scheduled in S. Since f is monotone and submodular, so is $\bar{f}$. Choosing more than a single pair associated with a job $J_i \in \mathcal{J}$ does not change the value of $\bar{f}$, thus, we can always remove the extra pairs from the schedule.

We set $\mathcal{I}$ to be the collection of all feasible schedules. Clearly, $(\mathcal{N}, \mathcal{I})$ is non-empty and monotone. Hence, we focus on the properties of Definition 5.1.1. Let $S, T \in \mathcal{I}$. For a pair $(J_i, t_i) \in T \setminus S$ choose $Y_{(J_i,t_i)}$ to be all pairs in $S \setminus T$ that intersect (J_i, t_i). If f is not monotone and submodular, we add to $Y_{(J_i,t_i)}$ also the single pair containing J_i in $S \setminus T$ if there is such pair. Since all intervals have length of exactly L, there are at most 2 pairs that can intersect a given pair $(J_i, t_i) \in T \setminus S$. This implies that $|Y_{(J_i,t_i)}| \leq 3$ (and $|Y_{(J_i,t_i)}| \leq 2$ if f is monotone and submodular). Symmetry shows also that a given $(J_r, t_r) \in S \setminus T$ belongs to at most three Y sets (two Y sets if f is monotone and submodular). Let $T' \subseteq T \setminus S$. We need to prove that $R = S \setminus \left(\cup_{(J_i,t_i) \in T'} Y_{(J_i,t_i)}\right) \cup T' \in \mathcal{I}$. Fix $(J_i, t_i) \in R$ and consider two cases. First, $(J_i, t_i) \in S \setminus T$. (J_i, t_i) does not intersect any pair in $S \setminus T$ (since S is a feasible schedule), and does not intersect any pair in T' (since if that was the case (J_i, t_i) would have belonged to $\cup_{(J_i,t_i) \in T'} Y_{(J_i,t_i)}$,which cannot be true $(J_i, t_i) \in R$). Moreover, for the same reasons, if f is not monotone and submodular, there is no other pair of job J_i in $S \setminus T$ or T'. Second, $(J_i, t_i) \in T \setminus S$. (J_i, t_i) does not intersect any pair of T' (since T is a feasible schedule) and does not intersect any pair in $S \setminus \left(\cup_{(J_i,t_i) \in T'} Y_{(J_i,t_i)}\right)$ (since if that was the case such a pair would have belonged to $Y_{(J_i,t_i)}$). Again, for the same reasons, if

f is not monotone and submodular, there is no other pair of job J_i in $S \setminus T$ or T'. We conclude that R is a feasible schedule, *i.e.*, $R \in \mathcal{I}$. □

For the special case of $L = 1$, we can prove a stronger claim.

Lemma 5.5.6. *Any instance of* `JISIL` *with* $L = 1$ *can be represented by a* 1*-exchange set system.*

Proof. Given a set $\mathcal{J}' \subseteq \mathcal{J}$ of jobs, Algorithm 12 can find a feasible schedule containing all the jobs of $\mathcal{J}'$, if such a schedule exists.

Algorithm 12: SCHEDULE-ALL($\mathcal{J}'$)

```
S ← {Empty Schedule}
i ← 0
while there are unscheduled jobs in J' do
    Let J_i be the set of unscheduled jobs from J' that can be scheduled to time i.
    if J_i ≠ ∅ then
        Let J_i be the job from J_i with the earliest deadline.
        Add J_i to S, and schedule it to time i.
Output S.
```

Let us prove that Algorithm 12 finds a feasible schedules for all the jobs of $\mathcal{J}'$, if such a schedule exists. For every iteration i of Algorithm 12, let S_i be the schedule constructed up to this iteration. We need to prove that if there exists a schedule S'_i such that $S_i \cup S'_i$ is a feasible schedule for all jobs in $\mathcal{J}'$, then there also exists a schedule S'_{i+1} such that $S_{i+1} \cup S'_{i+1}$ is a feasible schedule for all jobs in $\mathcal{J}'$.

If $\mathcal{J}_i$ is empty then $S_i = S_{i+1}$, and we are done. Otherwise, let J_i be the job scheduled by the algorithm to time i. J_i must be scheduled by S'_i to some time, say j. Let J_j be the job scheduled by S'_i to time i. Clearly, J_i can be scheduled to time i. Moreover, since both J_i and J_j belong to $\mathcal{J}_i$, the deadline of J_j is no earlier than that of J_i. Hence, J_j can be scheduled to time j. Thus, we can switch the times of jobs J_i and J_j in S'_i, and get a new schedule S''_i such that $S_i \cup S''_i$ is a feasible schedule for all jobs in $\mathcal{J}'$, and J_i is scheduled to time i by S''_i. By removing job J_i from S''_i we get a schedule with all the required properties to be S'_{i+1}.

At this point we can present our set system. $\mathcal{N}$ is the set of all jobs, and $\mathcal{I}$ is the collection of subset of jobs having a feasible schedule. By the above proof, it is possible to determine in polynomial time if a set of jobs is in $\mathcal{I}$. Clearly this set system is non-empty and monotone. Hence, we focus on the properties of Definition 5.1.1. Consider two sets $S, T \in \mathcal{I}$, and let $J \in T \setminus S$. We construct the set Y_J using the following process. Initially i is the time in which schedule T schedules job J. If S schedules no job at time i, then $Y_J = \varnothing$. Otherwise, If S schedules a job $J' \in S \setminus T$, then $Y_J = \{J'\}$. Finally, if S schedules a job $J' \in S \cap T$, then we update i to be the time in which schedule T schedules J', and start again.

Let us explain why the above process must terminate. Think of a digraph G containing the jobs of $T \cup S$ as nodes. For every job $J_1 \in T$ and $J_2 \in S$, there is an arc $(J_1 \to J_2)$ if J_1 is scheduled in T in the same time that J_2 is schedules in S. Observe that the in-degree and out-degree of each node in this graph is at most 1. Notice that the above process goes along a path (or cycle) on this graph. The only case that the above process can continue infinitely is when it goes along a cycle. However, the process starts with a node $J \in T \setminus S$

whose in-degree is 0, and therefore, there is no way to get back to this node once the process leaves it.

The size of each set Y_J is, by definition, at most 1 for every $J \in T \setminus S$. Consider some job $J \in T \setminus S$ with non-empty $Y_J = J'$. Notice that $J' \in S \setminus T$, and therefore, J and J' must be the end points of a path of the above graph, because all interval nodes of a path belong to $S \cap T$. This clearly implies that every node $J' \in S \setminus T$ can appear in at most one set of the form Y_J. Finally, consider a subset T' of jobs from T. We need to prove that there exists a schedule for the jobs of $S \cup T' \setminus \cup_{J \in T'} Y_J$. Consider the following schedule. For every job $J \in T \setminus S$, let P_J be the set of nodes along the path of the above graph whose end points are J and $J' \in Y_J$. Schedule all the jobs of $S \setminus \cup_{J \in T'} P_J$ at the time they are scheduled in S. Schedule all the jobs of $\cup_{J \in T'} P_J \setminus Y_J$ at the time they are scheduled in T. Let us prove that this is indeed a feasible schedule. Assume for the sake of contradiction that there exist two jobs in this schedule scheduled to the same time i. Clearly, one job of these two must belong to $S \setminus \cup_{J \in T'} P_J$ and the other to $P_J \setminus Y_J$ for some $J \in T'$. However, this implies that there is an edge in the above graph from a node of P_J to a node of $S \setminus \cup_{J \in T'} P_J$, which is, of course, a contradiction. □

Next, we consider the `maximum asymmetric traveling salesperson` problem (`MATS`).

Definition 5.5.6. *Given a complete directed graph $G = (V, E)$, `MATS` is the problem of finding a directed hamiltonian cycle $C \subseteq E$ maximizing $f(C)$ for a given $f : 2^E \to \mathbb{R}^+$.*

Lemma 5.5.7. *For any instance of `MATS`, the set of hamiltonian cycles (with all their subsets) is a 3-exchange set system.*

Proof. The set of edges is E is the ground set. A set $S \subseteq E$ is an independent set, *i.e.*, $S \in \mathcal{I}$, if the directed graph $G_S = (V, S)$ is either a directed hamiltonian cycle or a set of disjoint simple paths (the in and out degree of every vertex are at most 1, and G_S does not contain any cycles shorter than $|V|$). Fix $S, T \in \mathcal{I}$. For every $e = (u \to v) \in T \setminus S$ choose $Y_e \subseteq S \setminus T$ to be the following three types of edges (if they exist): an edge $e_1 \in S \setminus T$ that enters v, an edge $e_2 \in S \setminus T$ that leaves u, and an edge $e_3 \in S \setminus T$ which is the first edge of this set one encounters when leaving v and going along edges from $S \cap T$ only. We prove that this set system is 3-exchange.

Clearly, the above set system is non empty and monotone. Hence, we focus on the properties of Definition 5.1.1. First, $|Y_e| \leq 3$ for every $e \in T \setminus S$. Second, fix $e' = (u \to v) \in S \setminus T$. The edge e' take at most once each type of the above three types of edges defining the Y sets. Clearly, this is true for the first two types (since $T \in \mathcal{I}$, and therefore, the in and out degree of every vertex in G_T is at most 1). For e' to be of the third type, there must be an edge $e^* \in T \setminus S$ that can be found in the following way: start from u and go backwards along edges of $S \cap T$ until hitting a the first vertex w that has an edge $e^* \in T \setminus S$ entering it (note that this implies that there is no edge from $S \cap T$ entering w). Since there could be at most a single such e^* edge, e' can be at most once an edge of the third type. This proves the second property of Definition 5.1.1.

We focus now on the third property of Definition 5.1.1. Fix $T' \subseteq T \setminus S$, our goal it to prove that $S \setminus (\cup_{e \in T'} Y_e) \cup T' \in \mathcal{I}$. Note that for every $e = (u \to v) \in T'$, Y_e contains the edges from $S \setminus T$ that leave u or enter v (if they exist). Hence, the out degree of u and the in degree of v in $S \setminus (\cup_{e \in T'} Y_e)$ are 0. Therefore, these degrees in $S \setminus (\cup_{e \in T'} Y_e) \cup T'$ are at most 1. Assume now that $S \setminus (\cup_{e \in T'} Y_e) \cup T'$ contains a non-hamiltonian cycle C. C must contain edges from both $S \setminus T$ and $T \setminus S$, since $S, T \in \mathcal{I}$. Choose $e' \in C \cap (S \setminus T)$ and $e \in C \cap (T \setminus S)$ such that the path using C from e to e' contains only edges from $S \cap T$. Such a pair of edges must exist in C. However, this is a contradiction since by definition $e' \in Y_e$. This proves that $S \setminus (\cup_{e \in T'} Y_e) \cup T'$ does not contain non-hamiltonian cycles. □

The last application we consider is `maximum frequency allocation on lines` (`MFAL`).

Definition 5.5.7. *Given a set of frequencies $\mathcal{F}$, an interference radius r and a set $\mathcal{P}$ of points on a line, where every point $P \in \mathcal{P}$ is associated with a list of frequencies $\mathcal{L}_P \subseteq \mathcal{F}$ and a positive number M_P, a* frequency assignment $A \subseteq \mathcal{P} \times \mathcal{F}$ *assigns some of the frequencies to every point. A frequency assignment is* legal *if it assigns to every point P a set F_P of frequencies such that:*

- $F_P \subseteq \mathcal{L}_P$.
- $|F_P| \leq M_P$.
- *For every two points $P_1, P_2 \in \mathcal{P}$, $\mathsf{dist}(P_1, P_2) < r \Rightarrow F_{P_1} \cap F_{P_2} = \varnothing$.*

The `maximum frequency allocation on lines` *(`MFAL`) problem is the problem of finding a legal frequency assignment $A \subseteq \mathcal{P} \times \mathcal{F}$ maximizing $f(A)$ for a given $f : 2^{\mathcal{P}\times\mathcal{F}} \to \mathbb{R}^+$.*

Lemma 5.5.8. *For any instance of `MFAL`, the set of legal frequency assignments is a 3-exchange set system.*

Proof. Fix the ground set $\mathcal{N} = \mathcal{P} \times \mathcal{F}$. A set $A \subseteq F$ is independent (*i.e.*, $S \in \mathcal{I}$), if it is a legal frequency assignment. It is easy to see that this set system is non empty and monotone. Thus, we focus on the properties of Definition 5.1.1. Fix $S, T \in \mathcal{I}$. For every point $P \in \mathcal{P}$, number the frequencies assigned to P by $S \setminus T$ from 1 to $|(S \setminus T) \cap (\{P\} \times \mathcal{F})|$ (notice that a frequency might get different numbers in the numbering associated with different points). Denote the number of a frequency F in the numbering associated with point P by $g_{S\setminus T}(P, F)$. Similarly, for every point $P \in \mathcal{P}$, number the frequencies assigned to P by $T \setminus S$, and denote by $g_{T\setminus S}(P, F)$ this numbering.

Let $N_r(P)$ be the set of points whose distance from P is less than r. For every pair $u = (P, F) \in T \setminus S$, define Y'_u to be the set $\{(P', F') \in S \setminus T | g_{T\setminus S}(P, F) = g_{S\setminus T}(P, F')\}$. Also construct for u a set $Y_u = Y'_u \cup [(N_r(P) \times \{F\}) \cap (S \setminus T)]$. Informally, Y_u contains two types of pairs from $S \setminus T$: at most one pair involving P itself and pairs that contain the same frequency and a point too close to $\mathcal{P}$.

The size of Y'_u is at most 1 because there can be at most one frequency F' for which $g_{T\setminus S}(P, F) = g_{S\setminus T}(P, F')$. Since S is independent, there can be at most 2 points in $N_r(P)$ that are assigned the frequency F by S. Hence, $|Y_u| \leq 2 + |Y'_u| \leq 3$. Exactly the same arguments also imply that every pair $u = (P, F) \in S \setminus T$ can appear in at most 3 sets of $\{Y_u | u \in T \setminus S\}$.

Let $T' \subseteq T \setminus S$ and consider $A = S \setminus (\cup_{u \in T'} Y_u) \cup T'$. Consider some point P. Under A, every frequency assigned to P has a distinct number, and therefore, one of the sets S or T assigns to P at least as many frequencies as A does. Moreover, for every pair $u = (P, F)$ that we added to A, Y_u contained every pair $(P', F) \in S \setminus T$ such that $\mathsf{dist}(P', P) < r$. Therefore, every two points P_1, P_2 such that $\mathsf{dist}(P_1, P_2) < r$ have disjoint sets of frequencies under A. We can now conclude that $A \in \mathcal{I}$. □

5.6 Other Results

The concept of k-exchange set systems was developed independently by Justin Ward and by us. Justin Ward's work and ours was published in a joint paper [35]. Wards proved the following theorem:

Theorem 5.6.1. *For every $\delta > 0$, there exits a polynomial $2/(k + 1 + \delta)$ approximation algorithm for maximizing a linear function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$ over a k-exchange set system.*

Notice that the last theorem is at least as good as the result presented above for linear functions for any $k \geq 3$. For $k = 2$, this theorem gives only $2/(3+\delta)$ approximation, while the result presented above provides a PTAS. In a later paper [82], Wards used a similar method to get the following result for monotone submodular functions.

Theorem 5.6.2. *For every $\varepsilon > 0$, there exits a polynomial $(k+3)/2 + \varepsilon$ approximation algorithm for maximizing a monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$ over a k-exchange set system.*

Once again, the result given by [82] is at least as good as the result presented above for monotone submodular functions for any $k \geq 3$. For $k = 2$, this theorem gives only $5/2 + \varepsilon$ approximation, while the result presented above provides $2 + \varepsilon$ approximation. It is an open question whether the method of Wards can be used also to derive a result for general non-monotone submodular functions.

Chapter 6

Contention Resolution Schemes

As previously mentioned, many algorithms for submodular maximization problems are composed of two parts: a solver for a fractional relaxation of the problem, and a rounding method. Building upon [6], [20] proposes a general *contention resolution framework* for rounding fractional solutions. Intuitively, the scheme works as follows. First, an approximate fractional solution x is found for the multilinear extension relaxation of the problem. Second, x is re-normalized (all its coordinates are multiplied by some value $b \leq 1$), and a random subset of elements is sampled according to the probabilities determined by x. Third and last, some of the sampled elements are discarded to guarantee the feasibility of the solution.

The first step can be performed by any algorithm for finding approximate fractional solutions for the multilinear relaxation. Let α be the approximation guarantee of the algorithm used. The re-normalization factor and the decision which elements to discard are determined by a constraint specific *contention resolution scheme*. Formally, a (b, c)-balanced contention resolution scheme for a constraint represented by a set system $(\mathcal{N}, \mathcal{I})$ is an algorithm that gets a vector $x \in b\mathcal{P}(\mathcal{I})$ (where $\mathcal{P}(\mathcal{I})$ is the convex hull of $\mathcal{I}$), picks a random subset $\mathtt{R}(x)$ according to probabilities determined by x, and then outputs a set $S \in \mathcal{I}$ obeying $\Pr[u \in S | u \in \mathtt{R}(x)] \geq c$ for every $u \in \mathcal{N}$. If the contention resolution scheme is monotonic, *i.e.*, $\Pr[u \in S]$ only increases when other elements are removed from $\mathtt{R}(x)$, then the framework guarantees an αbc approximation for maximizing a submodular function subject to the set system $(\mathcal{N}, \mathcal{I})$. One advantage of this framework is the ease with which it deals with intersections of constraints of different types (*e.g.*, matroids, knapsack constraints and matchoids). Chekuri et al. [20] show that given contention resolution schemes for two types of constraints, there is a standard method to get a contention resolution scheme for the intersection of these constraints.

We extend the framework of [20] by showing that finding a fractional solution for the relaxation and the re-normalization step, can both be done simultaneously using the measured continuous greedy algorithm. Equipped with this observation, we can replace the expression αbc for the approximation ratio with an improved one for both the non-monotone and the monotone cases. The improvement achieved by the new expression is most significant for small values of b.

The idea behind our method is to use b as the stopping time of Theorems 3.2.1 and 3.2.2, hence directly getting a re-normalized fractional solution (as is guaranteed by both theorems). The following theorem presents the improved expressions for the approximation ratio. Its proof appears in Section 6.1.

Theorem 6.0.3. *If there is a monotone (b, c)-balanced contention resolution scheme for $\mathcal{I}$, then there is an approximation of $\left(e^{-b}bc - o(1)\right)$ for $\max_{S \in \mathcal{I}}\{f(S)\}$ assuming f is*

non-negative and submodular, and an approximation of $\left((1-e^{-b})c - o(1)\right)$ *assuming* f *is monotone.*

Note that the results of Theorem 6.0.3 are better than the (αbc)-approximation of [20]. This is true, since for the non-monotone case $e^{-b} > 0.325$ for every $b \in (0,1]$, and for the monotone case $1 - e^{-b} \geq (1-1/e)b$ for every $b \in (0,1]$.

We also provide monotone balanced contention resolution schemes for various matching, scheduling and packing problems. Using these schemes and Theorem 6.0.3, we are able to improve the known approximation ratios for these problems. A comprehensive list of our schemes and the problems for which they provide an improvement appears in Section 6.2. Among the results of Section 6.2, there are two that are especially notable:

- For job interval scheduling with k identical machines, and a *linear* objective function, we get an approximation ratio approaching 1 for large values of k. The previously best approximation ratio for this problems approached $1 - e^{-1}$ for large k's [9].
- For broadcast scheduling with a monotone submodular objective function, we get an approximation ratio of 1/4. This matches the best known approximation for the linear variant [8].

6.1 Combining the Framework with the Measured Continuous Greedy

Theorem 6.0.3 quantifies the approximation ratio that can be achieved by combining the measured continuous greedy and the contention resolution framework. This approximation ratio is better than what can be expected by a black-box combination of the two. We begin this section by proving Theorem 6.0.3. After the proof we discuss some applications of this theorem.

Proof of Theorem 6.0.3. Consider the case of a non-negative and submodular f. Apply Theorem 3.2.1 with stopping time $T = b$ (recall $0 \leq b \leq 1$). Then we obtain a fractional solution $x \in b\mathcal{P}$ whose value satisfies: $F(x) \geq (be^{-b} - o(1)) \cdot f(OPT)$. The rest of the proof is exactly as in Theorem 1.8 of [20], thus details are omitted.

For the case of a normalized, monotone and submodular f, apply Theorem 3.2.2 with stopping time $T = b$ (recall that, as before, $0 \leq b \leq 1$). Then we obtain a fractional solution $x \in b\mathcal{P}$ whose value satisfies: $F(x) \geq (1 - e^{-b} - o(1)) \cdot f(OPT)$. Again, the rest of the proof is exactly as in Theorem 1.8 of [20]. □

Theorem 6.0.3 implies improved approximation ratios for many problems considered in [20]. For example, consider the problem of maximizing a submodular function subject to k matroid constraints. Chekuri et al. [20] describe, for every $b \in (0,1]$ a monotone $(b, [(1 - e^{-b})/b]^k)$-balanced contention resolution scheme for this problem, using this contention resolution scheme they derive an approximation ratio of $0.19/k$ (for large enough k values). The following corollary improves this approximation ratio using Theorem 6.0.3 and a better choice of value for b.

Corollary 6.1.1. *For large enough values of* k*, there is a* $0.735/k - o(1)$ *approximation algorithm for maximizing a general non-monotone submodular function subject to* k *matroid constraints.*

Proof. By Theorem 6.0.3 and the above contention resolution scheme, there is a $e^{-b}b[(1-e^{-b})/b]^k - o(1)$ approximation algorithm for the problem of maximizing a general non-monotone submodular function subject to k matroid constraints. Choosing $b = 2/(k+1)$, we get:

$$\begin{aligned} e^{-b}b[(1-e^{-b})/b]^k - o(1) &\geq e^{-b}b[(b-b^2/2)/b]^k - o(1) = e^{-b}b[1-b/2]^k - o(1) \\ &= e^{-2/(k+1)} \cdot \frac{2}{k+1} \cdot \left[1 - \frac{1}{k+1}\right]^k - o(1) \\ &\geq e^{-2/(k+1)} \cdot \frac{2}{k+1} \cdot e^{-1} - o(1) \\ &= \frac{2}{e} \cdot \frac{e^{-2/(k+1)}}{k+1} - o(1) = 0.735/k - o(1) \ . \end{aligned}$$

□

Remark: Notice that the problem considered in the last corollary also has an approximation algorithm with an approximation ratio close to $1/k$ for large values of k [62], however, this result has polynomial time complexity only if k is fixed.

Table 6.1 summarizes a few other examples of the improvements achieved by Theorem 6.0.3 to problems considered in [20].

Table 6.1: A few examples of improved approximation ratios due to Theorem 6.0.3. All previous results in this table are due to [20]. The notation of ε denotes an arbitrarily small positive constant.

Problem	**This Thesis**	**Previous Result**
k-Uniform Matchoid (linear)	$\frac{2}{e}\frac{1}{k+1}$	$0.6/k$
k-Uniform Matchoid (monotone)	$\frac{2}{e}\frac{1}{k+1} - o(1)^*$	$0.38/k$
k-Uniform Matchoid (non-monotone)		$0.19/k$
k Matroid Intersection *and* $O(1)$ Knapsacks (linear)	$\left(\frac{2}{e} - \varepsilon\right)\frac{1}{k+1}$	$0.6/k$
k Matroid Intersection *and* $O(1)$ Knapsacks (monotone)	$\left(\frac{2}{e} - \varepsilon - o(1)\right)\frac{1}{k+1}^*$	$0.38/k$
k Matroid Intersection *and* $O(1)$ Knapsacks (non-monotone)		$0.19/k$

* The $o(1)$ term is with respect to $\min\{n, k\}$, *i.e.*, it vanishes when both n and k are large. The exact function hiding behind the $o(1)$ term depends on the type of the objective function.

6.2 Contention Resolution Schemes

In this section we provide improved contention resolution schemes for several matching, schedule and packing problems. These schemes imply improved approximation ratios for these problems using the framework of [20]. Moreover, Theorem 6.0.3 provides an additional improvement to these approximation ratios. Table 6.2 summarizes the approximation ratios proved in this section.

6.2.1 The `Submodular Independent Set in Interval Graphs` Problem

The first problem we consider is the `Submodular Independent Set in Interval Graphs` problem since many (constrained) submodular maximization problems can be reduced to it. In this problem we are given a set $\mathcal{N}$ of intervals, and a non-negative submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$. A solution $S \subseteq \mathcal{N}$ is feasible if no two intervals in S intersect, *i.e.*, the constraint family $\mathcal{I}$ contains all independent sets of the interval graph defined

Table 6.2: Results proved in Section 6.2. Objective functions: NMS - normalized monotone and submodular, NS - non-negative submodular, and L - linear.

Problem	**Objective function**	**This Paper**	**Previous Result**
`Submodular Independent Set in Interval Graphs`	NMS	$1/4$	$O(1)^*$ [20]
	NS	$1/(2e)$	
`Submodular` k-`Colorable Subgraph in Interval Graphs`	NMS	$1-e^{-1}-o(1)^{\triangle}$	
	NS	$e^{-1}-o(1)^{\triangle}$	
`Submodular Job Interval Selection`(single machine)	NMS	$1/4$	—
`Submodular Job Interval Selection`(k unrelated machines)	NMS	$1/4$	—
`Submodular Job Interval Selection`(k identical machines)	NMS	$1-e^{-1}-o(1)^{\triangle}$	—
	L	$1-o(1)^{\triangle}$	$1-e^{-1}-o(1)^{\triangle}$ [9]
`Submodular Multiple Knapsacks`	NMS	$1/4^{\square}$	—
`Submodular Multiple Knapsacks`(identical knapsack sizes)	NMS	$1-e^{-1}-o(1)^{\triangle,\square}$	—
`Submodular Broadcast Scheduling`	NMS	$1/4$	—
`Submodular Matching Scheduling`(edge degree $\leq k$)	NMS	$\frac{k}{e(k+1)^2}$	—

* The exact constant was not calculated by [20], but it is inferior to our corresponding results.
$^{\triangle}$ The $o(1)$ term is with respect to $\min\{n,k\}$, *i.e.*, it diminishes when both n and k are large.
$^{\square}$ Requires pseudo polynomial time. It is possible to get a polynomial time algorithm at the cost of some loss in the approximation ratio. For details, see the theorems below.

by $\mathcal{N}$. The goal is to find a feasible solution S which maximizes $f(S)$. Unlike its linear variant, `Submodular Independent Set in Interval Graphs` is NP-hard. This follows, *e.g.*, from the reduction proved by Lemma 6.2.5.

We note that `Submodular Independent Set in Interval Graphs` is a special case of the `unsplittable flow` problem where the graph is a path and all demands and capacities equal 1. For the unsplittable flow problem with general demands and capacities, [20] presents a $(b, 1-\rho b)$-balanced contention resolution scheme for some constant $\rho > 1$. This gives a $(1-e^{-b})(1-\rho b)$-approximation for the monotone case and a $(be^{-b}-o(1)(1-\rho b)$-approximation for the non-monotone case.[1] We show that `Submodular Independent Set in Interval Graphs` admits an improved monotone (b, e^{-b})-balanced contention resolution scheme for every $b \in (0,1]$, and thus, has better approximation ratios than the ones known for unsplittable flow. We get $(1-e^{-b})e^{-b}$ and $\left(be^{-2b}-o(1)\right)$ approximation for the monotone and non-monotone cases, respectively.

Algorithm 13 is the algorithm induced by our contention resolution scheme for the `Submodular Independent Set in Interval Graphs` problem. We give the complete algorithm, instead of only the contention resolution scheme because we believe the algorithm is more natural this way.

The first step of the algorithm is the relaxation solving and re-normalization steps

[1] In both cases, the approximation ratios presented are the ones achieved using our measured continuous greedy algorithm, and Theorem 6.0.3. These ratios are somewhat better than the original ones given by [20].

Algorithm 13: Contention Resolution Scheme for `Submodular Independent Set in Interval Graphs`$(\mathcal{N}, f, b)$

`// Computing Fractional Solution`

1 Use the *measured continuous greedy* algorithm with stopping time $T = b$ to obtain x.

`// Sampling`

2 Sample R where each interval $u \in \mathcal{N}$ is chosen independently with probability $1 - e^{-x_u}$.

`// Diluting`

3 For every $u \in \mathcal{N}$, mark interval u for deletion if there is a different interval $u' \in R$ that intersects the starting point of u.

`// Output`

4 Remove all marked intervals from R, and let S be the set of remaining intervals.

5 Output S.

of the contention resolution framework. The relaxation we use is the natural relaxations of the problem. The second step of the algorithm corresponds to the sampling step of the framework. We note that the probability of choosing an element into R in Line 2 is not as in [20]. However, since $1 - e^{-x} \leq x$, one can think of it as the sampling step of the framework, followed by a second sampling step at the beginning of the contention resolution scheme. More formally, the sampling step of the framework outputs a set R' containing every element $u \in \mathcal{N}$ with probability x_u. The contention resolution scheme starts by constructing a set R containing every element $u \in R'$ with probability $[1 - e^{-x_u}]/x_u$.

Lemma 6.2.1. *For every $b \in (0, 1]$, Algorithm 13 represents a monotone (b, e^{-b})-balanced contention resolution scheme for* ***`Submodular Independent Set in Interval Graphs`****.*

Proof. First, we prove that for every interval $u \in \mathcal{N}$:

$$\Pr[u \in S] \geq \left(1 - e^{-x_u}\right) e^{x_u - b} \ .$$

Let $\mathcal{L}_u$ be the set of intervals that intersect the starting point of u, excluding u itself. Theorems 3.2.1 and 3.2.2 guarantee that $x \in b\mathcal{P}$. Hence, $\sum_{u' \in \mathcal{L}_u} x_{u'} \leq b - x_u$. Therefore,

$$\begin{aligned}\Pr[u \in S] &= \left(1 - e^{-x_u}\right) \cdot \prod_{u' \in \mathcal{L}_u} e^{-x_{u'}} = \left(1 - e^{-x_u}\right) \cdot e^{-\sum_{u' \in \mathcal{L}_u} x_{u'}} \\ &\geq \left(1 - e^{-x_u}\right) \cdot e^{-(b - x_u)} = \left(1 - e^{-x_u}\right) \cdot e^{x_u - b} \ .\end{aligned}$$

Since $u \in S$ implies $u \in R'$, we get:

$$\begin{aligned}\Pr\left[u \in S \mid u \in R'\right] &= \frac{\Pr[u \in S \wedge u \in R']}{\Pr[u \in R']} \geq \frac{\Pr[u \in S]}{\Pr[u \in R']} \\ &\geq \frac{\left(1 - e^{-x_u}\right) e^{x_u - b}}{x_u} = \frac{\left(e^{x_u} - 1\right) e^{-b}}{x_u} \geq e^{-b} \ .\end{aligned}$$

The monotonicity of contention resolution scheme is clear from the description of Algorithm 13. □

Corollary 6.2.2. *For normalized monotone submodular f* ***`Submodular Independent Set in Interval Graphs`*** *has $1/4$-approximation, and for non-negative submodular f it has $(1/(2e) - o(1))$-approximation.*

Proof. For the former case apply Theorem 6.0.3 and Lemma 6.2.1 with $b = \ln 2$ to obtain an approximation of $1/4$. For the latter case apply Theorem 6.0.3 and Lemma 6.2.1 with $b = 1/2$ to obtain an approximation of $1/(2e) - o(1)$. □

We consider also a variation of `Submodular Independent Set in Interval Graphs` in which a valid solution can contain up to k intervals covering each time point on the line (as opposed to just a single one). We denote this problem as the `Submodular` k-`Colorable Subgraph in Interval Graphs` problem. Algorithm 13, and the contention resolution scheme implying it, can still be applied with a few minor changes.

Algorithm 14: Contention Resolution Scheme for `Submodular` k-`Colorable Subgraph in Interval Graphs` $(\mathcal{N}, f, k, b)$

`// Computing Fractional Solution`

1 Use the *measured continuous greedy* algorithm with stopping time $T = b$ to obtain x.

`// Sampling`

2 Sample R where each interval $u \in \mathcal{N}$ is chosen independently with probability x_u.

`// Diluting`

3 For every $u \in \mathcal{N}$, mark interval u for deletion if there are at least k other intervals $u' \in R$ that intersect the starting point of u.

`// Output`

4 Remove all marked intervals from R, and let S be the set of remaining intervals.

5 Output S.

The first step of the algorithm is the relaxation solving and re-normalization steps of the contention resolution framework. The relaxation we use is the natural relaxations of the problem. The second step of the algorithm corresponds to the sampling step of the framework. Note that here, as expected, and unlike the case in Algorithm 13, we sample every element $u \in \mathcal{N}$ with probability x_u.

Lemma 6.2.3. *Algorithm 14 represents a monotone* $(b, 1 - e^{-[1/b-1]^2 bk/3})$*-balanced contention resolution scheme for* `Submodular` k-`Colorable Subgraph in Interval Graphs` *for* $b \in (0, 1]$.

Proof. Let $\mathcal{L}_u$ be the set of intervals that intersect the starting point of u, excluding u itself. Theorems 3.2.1 and 3.2.2 guarantee that $x \in b\mathcal{P}$. Hence, $\sum_{u' \in \mathcal{L}_u} x_{u'} \leq b(k - x_u) \leq bk$. By using standard Chernoff bound on the upper tail, one can show that:

$$\Pr[|\mathcal{L}_u \cap R| \geq k - 1] = \Pr[|\mathcal{L}_u \cap R| > k] \leq e^{-[1/b-1]^2 bk/3} \ .$$

Observe that the above inequality is independent of the question whether $u \in R$. Hence,

$$\Pr\left[u \in S \mid u \in R\right] \geq 1 - e^{-[1/b-1]^2 bk/3} \ .$$

The monotonicity of contention resolution scheme is clear from the description of Algorithm 14. □

We can now prove the next corollary. The $o(1)$ term in this corollary is with respect to $\min\{k, n\}$, hence, it diminishes when both k and n are large.

Corollary 6.2.4. *For normalized monotone submodular f the above algorithm with $b = 1/[1+\sqrt{\log k/k}]$ provides a $(1-1/e-o(1))$-approximation for* ***Submodular k-Colorable Subgraph in Interval Graphs****. For non-negative submodular f it provides $(1/e-o(1))$-approximation.*

Proof. Notice that $b = 1-o(1)$, where the $o(1)$ term is, as usual, with respect to $\min\{k,n\}$. Let us now calculate:

$$b\cdot\left(1-e^{-[1/b-1]^2bk/3}\right) = b\cdot\left(1-e^{-[\sqrt{\log k/k}]^2bk/3}\right) = (1-o(1)\cdot\left(1-k^{-(1-o(1))/3}\right) = 1-o(1)\ .$$

By the result of [20], the approximation ratio is equal to αbc, where α is the approximation ratio of the relaxation solving algorithm. The last calculation shows that $bc = 1-o(1)$ in our case, hence, the approximation ratio we get is $\alpha - o(1)$. □

6.2.2 The `Submodular Job Interval Selection` Problem

The `Submodular Job Interval Selection` problem is defined as following. We are given a set $\mathcal{N}$ of n jobs, where each job $u \in \mathcal{N}$ is associated with a set $\mathcal{J}_u$ of possible intervals. Additionally, a normalized monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$ over the jobs is given (note that f is not defined over the intervals, but over the jobs). A feasible schedule is a subset $S \subseteq \cup_{u\in\mathcal{N}}\mathcal{J}_u$ of intervals, such that no two intervals in S intersect. A job u is *scheduled* in S if S contains an interval from $\mathcal{J}_u$. The goal is to find a feasible schedule S that maximizes the value of f over the set of jobs scheduled in S. We show that `Submodular Job Interval Selection` can be reduced via an approximation preserving reduction to `Submodular Independent Set in Interval Graphs`. This reduction works only for normalized *monotone* submodular objective functions, hence, we do not consider general non-negative submodular objectives.

Lemma 6.2.5. *For normalized monotone and submodular objective functions, there is an efficient approximation preserving reduction from* ***Submodular Job Interval Selection*** *to* ***Submodular Independent Set in Interval Graphs****.*

Proof. Fix an instance of `Submodular Job Interval Selection` with a normalized monotone and submodular objective f, and let us define the following instance of `Submodular Independent Set in Interval Graphs`. Set the interval set $\mathcal{N}'$ of the instance to be all intervals in $\mathcal{N}' = \cup_{u\in\mathcal{N}}\mathcal{J}_u$ while keeping multiplicities. Formally, $\mathcal{N}' = \{I^u | I \in \mathcal{J}_u, u \in \mathcal{N}\}$ (we keep the multiplicities by adding a superindex of u for every interval according to the job it belongs to). Define $\bar{f}$ as the objective function of the `Submodular Independent Set in Interval Graphs` instance by setting $\bar{f}(S)$ to be the value of f over all jobs u scheduled by S (*i.e.*, S contains an interval of $\mathcal{J}_u$). Given an oracle for f, one can construct an oracle for calculating $\bar{f}$ in polynomial time. Clearly every feasible solution to `Submodular Job Interval Selection` can be translated to a feasible solution to `Submodular Independent Set in Interval Graphs` while keeping the value of the objective function, and vice versa. Hence, we are left to prove that $\bar{f}$ is a normalized monotone submodular function.

First, we show that $\bar{f}$ is normalized. The empty schedule contains no intervals, and therefore, $\bar{f}(\varnothing) = f(\varnothing) = 0$.

Second, we show that $\bar{f}$ is monotone. Given two sets $S_1 \subseteq S_2 \subseteq \mathcal{N}'$, let $\mathcal{N}_1$ and $\mathcal{N}_2$ be the set of jobs that are scheduled by S_1 and S_2, respectively. Clearly, every job that has an interval in S_1 also has an interval in S_2, hence $\mathcal{N}_1 \subseteq \mathcal{N}_2$. Hence, due to the monotonicity of f, $\bar{f}(S_1) = f(\mathcal{N}_1) \leq f(\mathcal{N}_2) = \bar{f}(S_2)$.

Third, we show that $\bar{f}$ is submodular. Assume $S_1, S_2, \mathcal{N}_1$ and $\mathcal{N}_2$ are defined as above, and let $I \in \mathcal{N}' - S_2$ be an interval outside of S_2. It is enough to show that $\bar{f}(S_1 + I) - \bar{f}(S_1) \geq \bar{f}(S_2 + I) - \bar{f}(S_2)$. Let u denote the (only) job that is associated with I. Clearly, the set of jobs that have at least one interval in $S_1 + I$ ($S_2 + I$) is $\mathcal{N}_1 + J$ (respectively, $\mathcal{N}_2 + J$). Using the properties of f, and the fact $\mathcal{N}_1 \subseteq \mathcal{N}_2$, we conclude:

$$\bar{f}(S_1 + I) - \bar{f}(S_1) = f(\mathcal{N}_1 + J) - f(\mathcal{N}_1) \geq f(\mathcal{N}_2 + J) - f(\mathcal{N}_2) = \bar{f}(S_2 + I) - \bar{f}(S_2) \ .$$

The last inequality requires some explanation. If $J \notin \mathcal{N}_2$, then it follows from submodularity. If $J \in \mathcal{N}_1$ then both sides of the inequality are 0, and it trivially holds. Finally, if $J \in \mathcal{N}_2$ but $J \notin \mathcal{N}_1$, then the right side of the inequality is 0, and the left side is non-negative due to the monotonicity of f. □

Note: For linear objective functions, the above two problems are *not* equivalent. Specifically, the job interval selection problem is hard while the maximum weight independent set in interval graph problem can be solved in polynomial time.

Corollary 6.2.6. *There is a* $1/4$*-approximation algorithm for* `Submodular Job Interval Selection` *with a normalized monotone and submodular objective function.*

We consider three variants of `Submodular Job Interval Selection`. First, we consider the case where there are k *identical* machines. Second, we consider the case where there are k *unrelated* machines. Third and last, we consider the case where the objective function is a linear function, and there are k identical machines.

k Identical Machines

An instance of the k identical machines variant of `Submodular Job Interval Selection` is identical to a standard instance with the following modification. A schedule S is a k-tuple $(S_1, S_2, \ldots, S_k)$, where every S_i is a set of intervals. S is a *feasible* schedule if no two intervals of the same set S_i intersect. As before, the goal is to maximize the value of f over the set of jobs scheduled in S.

Lemma 6.2.7. *For normalized monotone and submodular objectives, there is an approximation preserving reduction from* `Submodular Job Interval Selection` *with* k *identical machines to* `Submodular` k`-Colorable Subgraph in Interval Graphs`*.*

Proof. Given an instance of `Submodular Job Interval Selection` with k *identical* machines which consists of $\mathcal{J}_u$ for every $u \in \mathcal{N}$, k and f, define an instance of `Submodular` k`-Colorable Subgraph in Interval Graphs` exactly as in the proof of Lemma (6.2.5). The proof that $\bar{f}$ is a normalized monotone submodular function, and that it can be calculated efficiently is identical to the equivalent part in the proof of Lemma (6.2.5). Additionally, from the construction of the `Submodular` k`-Colorable Subgraph in Interval Graphs` instance, it is clear that given any schedule of the `Submodular Job Interval Selection` instance, one can translate it into a solution for the `Submodular` k`-Colorable Subgraph in Interval Graphs` instance with the same objective value, and vice versa. □

Corollary 6.2.8. *There is a* $(1 - 1/e - o(1))$ *approximation algorithm for* `Submodular Job Interval Selection` *with* k *identical machines and a normalized monotone and submodular objective, where the* $o(1)$ *term is with respect to* $\min\{k, n\}$*.*

k Unrelated Machines

An instance of the k unrelated machines variant of `Submodular Job Interval Selection` is identical to a standard instance with the following modification. A job $u \in \mathcal{N}$ is associated with k sets of intervals: $\mathcal{J}_{u,1}, \mathcal{J}_{u,2}, \ldots, \mathcal{J}_{u,k}$, where set $\mathcal{J}_{u,i}$ is the collection of allowed intervals of job u on the i^{th} machine. A schedule S is a k-tuple $(S_1, S_2, \ldots, S_k)$, where every S_i is a subset of the intervals allowed for the i^{th} machine. Again, S is a *feasible* schedule if no two intervals of the same set S_i intersect.

Lemma 6.2.9. *For normalized, monotone and submodular objectives, there is an approximation preserving reduction from* ***`Submodular Job Interval Selection`*** *with k unrelated machines to* ***`Submodular Job Interval Selection`****.*

Proof. Given an instance of `Submodular Job Interval Selection` with k unrelated machines and a normalized monotone and submodular objective f, define the following instance of `Submodular Job Interval Selection`. Given an interval I, let $I+t$ denote the same interval shifted by t, *i.e.*, $I+t$ has the same length as I, but starts t time units later. We also denote by T the latest end time of any interval in the original instance. For every job u, its set of intervals in the new `Submodular Job Interval Selection` instance is: $\mathcal{J}'_u = \cup_{i=1}^{k} \{I + iT | I \in \mathcal{J}_{u,i}\}$. Informally, intervals of machine i are placed between times iT and $(i+1)T$. This guarantees that intervals which are originally from different machines never intersect. Define $\bar{f}$ as the objective function of the new `Submodular Job Interval Selection` instance by setting $\bar{f}(S)$ to be the value of f over all jobs u scheduled by S (*i.e.*, S contains an interval of $\mathcal{J}'_u$). Given an oracle for f, one can construct an oracle for calculating $\bar{f}$ in polynomial time.

Recall that intervals that are originally from different machines never intersect in the new instance. Using this observation, every feasible solution to the original instance of `Submodular Job Interval Selection` with k unrelated machines can be translated to a feasible solution of the new `Submodular Job Interval Selection` instance while keeping the value of the objective function, and vice versa. Hence, we are left to prove that $\bar{f}$ is a normalized monotone submodular function.

First, we show that $\bar{f}$ is normalized. The empty schedule contains no intervals, and therefore, $\bar{f}(\varnothing) = f(\varnothing) = 0$.

Second, we show that $\bar{f}$ is monotone. Let $\mathcal{I}'$ be the set of all intervals of the new instance of `Submodular Job Interval Selection`. Given two sets $S_1 \subseteq S_2 \subseteq \mathcal{I}'$, let $\mathcal{N}_1$ and $\mathcal{N}_2$ be the set of jobs that are scheduled by S_1 and S_2, respectively. Clearly, every job that has an interval in S_1 also has an interval in S_2, thus, $\mathcal{N}_1 \subseteq \mathcal{N}_2$. Hence, due to the monotonicity of f, $\bar{f}(S_1) = f(\mathcal{N}_1) \leq f(\mathcal{N}_2) = \bar{f}(S_2)$.

Third, we show that $\bar{f}$ is submodular. Assume $S_1, S_2, \mathcal{N}_1$ and $\mathcal{N}_2$ are defined as above, and let $I \in \mathcal{I}' - S_2$ be an interval outside of S_2. It is enough to show that $\bar{f}(S_1+I) - \bar{f}(S_1) \geq \bar{f}(S_2 + I) - \bar{f}(S_2)$. Let u denote the (only) job that is associated with I. Clearly, the set of jobs that have at least one interval in $S_1 + I$ ($S_2 + I$) is $\mathcal{N}_1 + J$ (respectively, $\mathcal{N}_2 + J$). Using the properties of f , and the fact $\mathcal{N}_1 \subseteq \mathcal{N}_2$, we get:

$$\bar{f}(S_1 + I) - \bar{f}(S_1) = f(\mathcal{N}_1 + J) - f(\mathcal{N}_1) \geq f(\mathcal{N}_2 + J) - f(\mathcal{N}_2) = \bar{f}(S_2 + I) - \bar{f}(S_2) \ .$$

The last inequality requires some explanations. If $J \notin \mathcal{N}_2$, then the inequality follows from the submodularity of f. If $J \in \mathcal{N}_1$, then both sides of the inequality are equal to 0, and the inequality trivially holds. Finally, if $J \in \mathcal{N}_2$, but $J \notin \mathcal{N}_1$, then the right hand side of the inequality is 0, while the left hand side is non-negative due to the monotinicity of f. □

Corollary 6.2.10. *`Submodular Job Interval Selection` with k unrelated machines has a $1/4$-approximation algorithm for normalized monotone and submodular objective functions.*

k Identical Machines with Linear Objective

An instance of this variant is an instance of the k identical machines variant, with a linear objective function. Since the objective is linear, one can use for this variant the following linear programming formulation instead of the multilinear extension formulation. For every job u, we denote by w_u its contribution to the objective. For every interval I, we denote by $\mathcal{L}_I$ the set of other intervals intersecting it.

$$\begin{array}{lll} \max & \sum_{u\in\mathcal{N}} w_u \cdot \left(\sum_{I\in\mathcal{J}_u} x_I\right) & \\ s.t. & \sum_{I\in\mathcal{J}_u} x_I \leq 1 & \forall u \in \mathcal{N} \\ & x_I + \sum_{I'\in\mathcal{L}_I} x_{I'} \leq k & \forall I \in \cup_{u\in\mathcal{N}}\mathcal{J}_u \\ & x_I \geq 0 & \forall I \in \cup_{u\in\mathcal{N}}\mathcal{J}_u \end{array}$$

The algorithm we use for this problem is Algorithm 15, which is a variant of Algorithm 14.

Algorithm 15: Algorithm for `Weighted Job Interval Selection` with k identical machines $(\mathcal{N}, f, k)$

`// Computing Fractional Solution`
1 Use an LP solver to find an x maximizing the linear relaxation of the problem.
`// Sampling`
2 For every job u sample at most one interval $I \in \mathcal{J}_u$, with probability $b \cdot x_I$ for every interval, where $b = 1/[1 + \sqrt{\log k/k}]$.
`// Diluting`
3 For every $I \in R$, mark interval I for deletion if there are at least k other intervals $I' \in R$ that intersect the starting point of I.
`// Output`
4 Remove all marked intervals from R, and let S be the set of remaining intervals.
5 Output S.

The $o(1)$ term in the next lemma is with respect to $\min\{k, n\}$, hence, it diminishes when both k and n are large.

Lemma 6.2.11. *`Weighted Job Interval Selection` with k identical machines has a polynomial time $(1 - o(1))$-approximation algorithm.*

The proof of this lemma is similar in spirit to that of Lemma 6.2.3. However here we do not need to evoke the framework of Theorem 6.0.3 and [20].

Proof. Fix an interval $I \in \mathcal{J}_u$. The constraints of the LP guarantee that $\sum_{I'\in\mathcal{L}_I} x_{I'} \leq k - x_I \leq k$. By using standard Chernoff bound on the upper tail, one can show that:

$$\Pr[|\mathcal{L}_I \cap R| \geq k - 1] = \Pr[|\mathcal{L}_I \cap R| > k] \leq e^{-[1/b-1]^2 bk/3} \ .$$

Observe that the above inequality is independent of the question whether $I \in R$. Hence,

$$\Pr[I \in S] = \Pr[I \in R] \cdot \Pr\left[I \in S \mid I \in R\right] \geq bx_I \cdot \left[1 - e^{-[1/b-1]^2 bk/3}\right] \ .$$

Notice that $b = 1 - o(1)$, where the $o(1)$ term is, as usual, with respect to $\min\{k, n\}$. Let us now calculate:

$$b \cdot \left(1 - e^{-[1/b-1]^2 bk/3}\right) = b \cdot \left(1 - e^{-[\sqrt{\log k/k}]^2 bk/3}\right) = (1-o(1) \cdot \left(1 - k^{-(1-o(1))/3}\right) = 1-o(1) \ .$$

Combing the two last equations, we get: $\Pr[I \in S] = x_I \cdot [1 - o(1)]$. Recall that the events $I_1 \in S$ and $I_2 \in S$ are disjoint for every two different intervals I_1 and I_2 of the same job. Thus, expected contribution of job u to the objective function is:

$$\sum_{I \in \mathcal{J}_u} w_u \cdot x_I \cdot [1 - o(1)] = [1 - o(1)] \cdot w_u \cdot \sum_{I \in \mathcal{J}_u} x_I \ .$$

By the linearity of the expectation, the expected value of S is equal to the objective of the linear program times $[1 - o(1)]$, which completes the proof of the lemma. □

Observe that this is a linear problem for which we improve the best known approximation ratio. The previously best approximation ratio for this problem approached $1 - e^{-1}$ for large k values [7].

6.2.3 The `Submodular Multiple Knapsacks` Problem

The `Submodular Multiple Knapsacks` problem is defined as following. We are given a collection $\mathcal{N}$ of n elements and k knapsacks, where the size of the i^{th} knapsack is $B_i \in \mathbb{N}$. We note that the number of knapsacks (k) might not be a constant. Each element $u \in \mathcal{N}$ has a given size $s_u \in \mathbb{N}$. In addition, we are also given a normalized monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$ defined over the elements. A feasible packing S is a k-tuple $(S_1, S_2, \ldots, S_k)$ such that the total size (*i.e.*, the sum of the sizes) of the elements in each set S_i is at most B_i. An element u is *packed* by S if there exists an S_i such that $u \in S_i$. The goal is to find a feasible packing S that maximizes the value of f over the set of elements packed by S. Note that `Submodular Multiple Knapsacks` differs from the usual k-knapsack constraints, discussed earlier in this work, in the sense that here we are asked to pack each element into up to one of several possible knapsack, while the problem of k-knapsack constraints ask to pack each element either to no knapsack or to all of them at the same time.

We first consider the variant of `Submodular Multiple Knapsacks` where all knapsacks have equal size B. We show that in this case `Submodular Multiple Knapsacks` can be reduced via an approximation preserving reduction to `Submodular` k-`Colorable Subgraph in Interval Graphs` (again, the reduction works only for normalized, monotone and submodular objective functions). However, the time complexity of this reduction depends on the sizes of the knapsacks.

Lemma 6.2.12. *For normalized, monotone and submodular objectives, there exists an approximation preserving reduction from* `Submodular Multiple Knapsacks` *with identical knapsack sizes, to* `Submodular` k-`Colorable Subgraph in Interval Graphs`*. The time complexity of this reduction is pseudo polynomial.*

Proof. Given an instance of `Submodular Multiple Knapsacks` with identical knapsack sizes, define an instance of `Submodular` k-`Colorable Subgraph in Interval Graphs` as follows. The number of machines (k) in the new instance is equal to the number of knapsacks in the original one. For each element $u \in \mathcal{N}$, create a collection of $B - s_u + 1$ intervals by constructing for every possible (integral) starting point from 0 up to $B - s_j$ an interval of length s_j that starts at that point. Denote this collection of intervals by $\mathcal{N}_u$.

Define: $\mathcal{N}' = \cup_{e_u \in \mathcal{N}} \mathcal{N}_u$. N' is the set of intervals in the constructed instance. Define $\bar{f}$ as the objective function of the `Submodular k-Colorable Subgraph in Interval Graphs` instance by setting $\bar{f}(S)$ to be the value of f over all elements $u \in \mathcal{N}$ for which $S \cap \mathcal{N}_u \neq \varnothing$. One can construct an oracle for calculating $\bar{f}$ in polynomial time, let us prove that $\bar{f}$ is a normalized, monotone and submodular function.

First, we show that $\bar{f}$ is normalized. The empty solution contains no intervals, and therefore, $\bar{f}(\varnothing) = f(\varnothing) = 0$.

Second, we show that $\bar{f}$ is monotone. Given two sets $S_1 \subseteq S_2 \subseteq \mathcal{N}'$, let $\mathcal{N}_1$ and $\mathcal{N}_2$ be the sets of elements that are covered by S_1 and S_2, respectively. Clearly, every element that has an interval in S_1 also has an interval in S_2, hence $\mathcal{N}_1 \subseteq \mathcal{N}_2$. Therefore, due to the monotonicity of f, $\bar{f}(S_1) = f(\mathcal{N}_1) \leq f(\mathcal{N}_2) = \bar{f}(S_2)$.

Third, we show that $\bar{f}$ is submodular. Assume $S_1, S_2, \mathcal{N}_1$ and $\mathcal{N}_2$ are defined as before, and let $I \notin S_2$ be an interval outside of S_2. It is enough to show that $\bar{f}(S_1 + I) - \bar{f}(S_1) \geq \bar{f}(S_2 + I) - \bar{f}(S_2)$. Let u denote the (only) element that is associated with I. Clearly, the set of elements that have at least one interval in $S_1 + I$ ($S_2 + I$) is $\mathcal{N}_1 + u$ (respectively, $\mathcal{N}_2 + u$). Using the properties of f, we can conclude that:

$$\bar{f}(S_1 + I) - \bar{f}(S_1) = f(\mathcal{N}_1 + u) - f(\mathcal{N}_1) \geq f(\mathcal{N}_2 + u) - f(\mathcal{N}_2) = \bar{f}(S_2 + I) - \bar{f}(S_2)$$

The last inequality requires some explanations. If $u \notin \mathcal{N}_2$, then the inequality follows from the submodularity of f. If $u \in \mathcal{N}_1$, then both sides of the inequality are equal to 0, and the inequality trivially holds. Finally, if $u \in \mathcal{N}_2$, but $u \notin \mathcal{N}_1$, then the right hand side of the inequality is 0, while the left hand side is non-negative due to the monotinicity of f.

Next, let us explain how to translate a feasible solution of the `Submodular Multiple Knapsacks` with identical knapsack sizes instance into a feasible solution of `Submodular k-Colorable Subgraph in Interval Graphs` without modifying the value of the objective. Let $S = (S_1, S_2, \ldots, S_k)$ be a feasible solution to the original instance of `Submodular Multiple Knapsacks` with identical knapsack sizes. Fix S_i, and let $u_{i_1}, u_{i_2}, \ldots, u_{i_m}$ be the elements of S_i. Let us construct a set $\bar{S}_i$ as following. For every $1 \leq i \leq m$, add the interval of u starting at time $\sum_{k=1}^{j-1} s_{u_{i,k}}$ to $\bar{S}_i$. Notice that this interval exists since the total size of the elements in S_i is at most B. Moreover, all the intervals in $\bar{S}_i$ are disjoint. We can now construct a solution $\bar{S}$ for `Submodular k-Colorable Subgraph in Interval Graphs` which is simply the union $\cup_{i=1}^{k} \bar{S}_i$. Since the union is over k sets only, there are at most k intervals in $\bar{S}$ that contain any point p. This completes the construction of $\bar{S}$, and the proof that $\bar{S}$ is a feasible solution. Clearly, $f(S) = \bar{f}(\bar{S})$.

We now prove the other direction. Let $\bar{S}$ be a solution to `Submodular k-Colorable Subgraph in Interval Graphs`. Since there are at most k intervals in $\bar{S}$ containing any point p, we can color all intervals in $\bar{S}$ by k colors such that any two intervals with the same color do not intersect. Recall that each interval is associated with an element $u \in \mathcal{N}$. Thus, we can define $S = (S_1, S_2, \ldots, S_k)$ by assigning all elements corresponding to intervals of color i to S_i. Note that S is a feasible solution of `Submodular Multiple Knapsacks` with identical knapsack sizes because all intervals of color i have total length of at most B (by the fact that any two intervals of the same color do not intersect). Clearly $\bar{f}(\bar{S}) = f(S)$, which completes the translation of the lemma. □

Corollary 6.2.13. *`Submodular Multiple Knapsacks` with identical knapsack sizes has $1 - e^{-1} - o(1)$ approximation algorithm with pseudo polynomial time complexity.*

The $o(1)$ term in Corollary 6.2.13 is with respect to $\min\{n, k\}$, *i.e.*, it diminishes when both n and k are large. Interestingly, the guarantee of Corollary 6.2.13 for many identical

knapsacks is identical to the one achieved by [77] for a single knapsack, which is known to be tight (*e.g.*, it can be easily proved using the symmetry gap technique of [81]).

In order to get an approximation that uses polynomial time regardless of the sizes of the knapsacks, additional techniques has to be applied. As usual, the $o(1)$ term in Lemma 6.2.14 is with regard to $\min\{n, k\}$.

Lemma 6.2.14. *There is a polynomial time* $((e-1)/(3e-1) - o(1))$*-approximation algorithm for* `Submodular Multiple Knapsacks` *with identical knapsack sizes and a normalized, monotone and submodular objective function.*

In the following proof we abuse notation, and unify a schedule with the set of elements within this schedule.

Proof. We suggest an algorithm that constructs three feasible solutions, and output the single solution with the maximum value. We assume there are no elements larger than B, otherwise, any such element can be safely removed. Let X be the set of all elements smaller than B/n. Notice that all the elements of X can be packed into a single knapsack. Hence, X is a feasible solution. Since f is monotone, the value of X is at least: $f(X) \geq f(X \cap OPT)$, where OPT is the optimal solution.

From now on we consider only the elements of $\mathcal{N} \setminus X$. We set the size of the knapsacks to $B' = n^2$, and scale the size of every element $u \in \mathcal{N} \setminus X$ accordingly to $s'_u = \lfloor s_u \cdot B'/B \rfloor$. Notice that the set of feasible solutions can only increase by this scaling, *i.e.*, $OPT \setminus X$ is still a feasible solution. We now apply the algorithm from Corollary 6.2.13 to the instance (with the ground set $\mathcal{N} \setminus X$). This algorithm runs in polynomial time on this instance because all numbers in the representation of the instance are smaller or equal to n^2. Let S be the output of the algorithm. Since $OPT \setminus X$ is still a feasible solution, we are guaranteed that $f(S) \geq [1 - e^{-1} - o(1)] \cdot f(OPT \setminus X)$.

Consider the set of elements S_i packed by S to some knapsack. The total *original* size of the elements in S_i is $B(1 + |S_i|/n^2) \leq B(1 + 1/n)$. Hence, by removing any single element from S_i, we get a set of elements of size at most B. In other words, the set of elements in S_i can be split into two disjoint sets Y_i and Z_i, each having a total size of at most B. By repeating this argument for all knapsacks, we get two disjoint feasible solutions Y and Z whose union is S. From submodularity and non-negativity, we get: $f(Y) + f(Z) \geq f(S) \geq [1 - e^{-1} - o(1)] \cdot f(OPT \setminus X)$.

Our algorithm picks the best set among X, Y and Z. The value of this solution is at least:

$$\max\{f(OPT \cap X), [1 - e^{-1} - o(1)]/2 \cdot f(OPT \setminus X)\} \ .$$

By the submodularity and non-negativity of f, we know that $f(OPT \cap X) + f(OPT \setminus X) \geq f(OPT)$. Hence, $f(OPT \cap X) \geq f(OPT) - f(OPT \setminus X)$. Plugging this into the previous expression, and denoting $f(OPT \setminus X)$ by x, we get that the algorithm outputs a solution of value at least:

$$\max\{f(OPT) - x, [1 - e^{-1} - o(1)]/2 \cdot x\} \ . \tag{6.1}$$

Clearly this expression is minimized when the two arguments of the max are equal, *i.e.*, when:

$$f(OPT) - x = [1 - e^{-1} - o(1)]/2 \cdot x \Rightarrow f(OPT) = [3 - e^{-1} - o(1)]/2 \cdot x \Rightarrow x = \frac{3 \cdot f(OPT)}{2 - e^{-1} - o(1)} \ .$$

Plugging this x into Equation 6.1 gives the following lower bound on the value of the algorithm's output:

$$\max\{f(OPT) - x, [1 - e^{-1} - o(1)]/2 \cdot x\} \geq f(OPT) - \frac{2 \cdot f(OPT)}{3 - e^{-1} - o(1)}$$
$$= \frac{1 - e^{-1} - o(1)}{3 - e^{-1} - o(1)} \geq \frac{e-1}{3e-1} - o(1) \ . \qquad \square$$

If the knapsacks have different sizes, the problem can still be reduced to `Submodular Independent Set in Interval Graphs`. This leads to somewhat weaker results, as can be seen from the following claims.

Lemma 6.2.15. *For normalized, monotone and submodular objectives, There is an approximation preserving reduction from `Submodular Multiple Knapsacks` to `Submodular Job Interval Selection`. The time complexity of this reduction is pseudo-polynomial.*

The main reason why the proof of the Lemma 6.2.12 fails for the general problem is that there is no way to convert a solution of `Submodular` k`-Colorable Subgraph in Interval Graphs` back to a solution `Submodular Multiple Knapsacks` when there are multiple knapsacks. The source of the failure is that intervals corresponding to different knapsacks appear in the solution together. To by pass that, the following proof has a disjoint range for intervals of each knapsack.

Proof. Given an instance of `Submodular Multiple Knapsacks`, we define an instance of `Submodular Job Interval Selection` as following. Let $\mathcal{B}$ be the size of the largest knapsack. Every element u of `Submodular Multiple Knapsacks` translates into a job u which has the following intervals: $\mathcal{J}_u = \{[i\mathcal{B}+j, i\mathcal{B}+j+s_u) \mid 1 \leq i \leq k, 0 \leq j \leq B_i - s_u\}$. The submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$ is common to both instances.

Let $S = (S_1, S_2, \ldots, S_k)$ be a feasible solution to the original instance of `Submodular Multiple Knapsacks`. Fix S_i, and let $u_{i_1}, u_{i_2}, \ldots, u_{i_m}$ be all the elements of S_i. Let us construct a set $\bar{S}_i$ as following. For every $1 \leq i \leq m$, add the interval of u starting at time $\sum_{k=1}^{j-1} s_{u_{i,k}}$ to $\bar{S}_i$. This interval exists since the total size of the elements in S_i is at most B_i. Moreover, all the intervals in $\bar{S}_i$ are disjoint, and are within the range $[i\mathcal{B}, (i+1)\mathcal{B})$. We can now construct a solution $\bar{S}$ for `Submodular Job Interval Selection` which is simply the union $\cup_{i=1}^{k} \bar{S}_i$. Since each one of the sets $\bar{S}_i$ contains intervals included in a different range $[i\mathcal{B}, (i+1)\mathcal{B})$, clearly $\bar{S}$ contains no two intersecting intervals. This completes the construction of $\bar{S}$, and the proof that $\bar{S}$ is a feasible solution. Clearly, $f(S) = \bar{f}(\bar{S})$ because the jobs corresponding to the intervals of $\bar{S}$ are exactly the elements of S.

Consider now the reverse direction. Let $\bar{S}$ be a feasible solution to the instance of `Submodular Job Interval Selection`. For every $1 \leq i \leq k$, we construct a set S_i containing all jobs for which $\bar{S}$ contains an interval in the range $[i\mathcal{B}, (i+1)\mathcal{B})$. By the construction of the `Submodular Job Interval Selection` instance, and the fact that no two intervals of $\bar{S}$ intersect, the total size of the elements of S_i must be at most B_i. Hence, $S = (S_1, S_2, \ldots, S_k)$ is a feasible solution for the original `Submodular Multiple Knapsacks` instance. Clearly, $f(S) = \bar{f}(\bar{S})$ because the jobs corresponding to the intervals of $\bar{S}$ are exactly the elements of S. $\square$

Corollary 6.2.16. *`Submodular Multiple Knapsacks` has* $1/4$ *approximation algorithm with pseudo polynomial time complexity.*

Again, it is possible to get an approximation that uses only polynomial time regardless of the sizes of the knapsacks, but at the cost of a somewhat inferior approximation ratio.

Lemma 6.2.17. *`Submodular Multiple Knapsacks` with a normalized, monotone and submodular objective function has a polynomial time* $(1/9 - o(1))$*-approximation algorithm.*

Proof. The proof of Lemma 6.2.14 works for this lemma also, with a few modifications.

- The set of elements in X contains all elements smaller than $B_{\max}/n$, where $B_{\max}$ is the largest knapsack. Notice that all the elements of X can still be packed into a single knapsack.
- We can assume all knapsacks are of size at least $B_{\max}/n$, otherwise, no remaining element can be packed into them.
- The size of a knapsack of size B is scaled to $\lceil B \cdot n^3/B_{\max} \rceil$, and the size each element u is scaled accordingly to $s'_u = \lfloor s_u \cdot n^3/B_{\max} \rfloor$. Notice that, as before, the set of feasible solutions only increases in the process, and all numbers become polynomial in n.
- We apply the algorithm of Corollary 6.2.16 instead of the algorithm given by Corollary 6.2.13. However, this time the algorithm cannot be applied in a black box fashion. The problem is that it is now possible that there is an element which could not fit into a given knapsack before the scaling, but fits into it after the scaling. To solve this problem, we require that the algorithm of Corollary 6.2.16 will not generate segments for a given combination of element and knapsack, unless the element fits into the knapsack before the scaling.
- Consider some knapsack, and the set S_i of elements S pack into this knapsack. The total *original* size of the elements of S_i can be at most:

$$\begin{aligned} \lceil B \cdot n^2/B_{\max} \rceil \cdot B_{\max}/n^3 + n \cdot (B_{\max}/n^3) &= B + B_{\max}/n^3 + B_{\max}/n^2 \\ &\leq B(1 + 1/n + 1/n^2) \ . \end{aligned}$$

  If the total *original* size exceeds B, we need to describe how to split S_i into Y_i and Z_i, and guarantee that both Y_i and Z_i are feasible. There are two case. If there is an element of size at least $B(1/n + 1/n^2)$ in the knapsack, then this element is placed into Y_i, and the remaining elements, are placed into Z_i. Clearly both sets are feasible. If no such element exists, we pick arbitrary two elements and place them in Y_i. The rest of the elements go to Z_i. The elements in Z_i are feasible because their total size is less than B. The two elements in Y_i also form a feasible set because their total size is at most $B(2/n + 2/n^2) \leq B$ (assuming $n \geq 3$). □

6.2.4 The `Submodular Broadcast Scheduling` Problem

The `Submodular Broadcast Scheduling` problem is defined as follows. We are given a set $\mathcal{K}$ of n pages, and a set $\mathcal{N}$ of requests. Each page $P \in \mathcal{K}$ is associated with a set $\mathcal{J}_P$ of intervals, and each request $u \in \mathcal{N}$ is associated with a page P_u and a subset $\mathcal{J}_u \subseteq \mathcal{J}_{P_u}$ of intervals of P_u. Two intervals of two pages are always considered to be different from each other, even if they cover exactly the same range (*i.e.*, they can appear together in one set). Additionally, we are given a normalized monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$. A feasible schedule is a set S of intervals, such that no two intervals intersect. A request u is *fulfilled* by schedule S if S contains an interval from $\mathcal{J}_u$. The goal is to find a feasible schedule S maximizing the value of f over the set of requests fulfilled by S.

One can assume without loss of generality that there is only one page $\bar{P}$ in any instance of `Submodular Broadcast Scheduling`. The reduction goes as following. Replace all pages with a new page $\bar{P}$. The set of intervals of $\bar{P}$ is $\mathcal{J}_{\bar{P}} = \cup_{P \in \mathcal{K}} \mathcal{J}_P$ (notice that $\mathcal{J}_{\bar{P}}$ can contain multiple intervals that cover the same range if they originate from different pages. Such intervals are considered different from each other). All requests $u \in \mathcal{N}$ are now associated with $\bar{P}$, but keep the same intervals set $\mathcal{J}_u \subseteq \mathcal{J}_{P_u} \subseteq \mathcal{J}_{\bar{P}}$. Neither the definition of feasible schedules, nor the objective of the problem, are changed by the introduction of $\bar{P}$. The following lemma is proved using this reduction.

Lemma 6.2.18. *There exists an approximation preserving reduction from* ***Submodular Broadcast Scheduling*** *to* ***Submodular Independent Set in Interval Graphs*** *for normalized, monotone and submodular objectives.*

Proof. Following the above reduction, we assume there is a single page $\bar{P}$ in the instance of `Submodular Broadcast Scheduling`. Given an instance of `Submodular Broadcast Scheduling`, we construct an instance of `Submodular Independent Set in Interval Graphs` as following. The set of intervals in the new instance is $\mathcal{J}_{\bar{P}}$, and its objective function $\bar{f}$ is defined as following:

$$\bar{f}(S) = f(\{u \in \mathcal{N} \mid S \cap \mathcal{J}_u \neq \varnothing\}) \ .$$

Notice that a set S of intervals is feasible under both instances if and only if it contains no intersecting intervals. Moreover, under both instances the objective value associated with S is the value of f over the set of requests u which have some request of $\mathcal{J}_u$ in S. Hence, we are only left to show that $\bar{f}$ is normalized, monotone and submodular.

Let us start with the normalization: $\bar{f}(\varnothing) = f(\varnothing) = 0$. Next, we need to show the monotonicity of $\bar{f}$. Let $S_1 \subseteq S_2$ be two sets of intervals. Let $\mathcal{N}_1$ ($\mathcal{N}_2$) be the set of requests u for which there is some interval of $\mathcal{J}_u$ in S_1 (respectively, S_2). Clearly, $\mathcal{N}_1 \subseteq \mathcal{N}_2$. Hence, $\bar{f}(S_1) = f(\mathcal{N}_1) \leq f(\mathcal{N}_2) = \bar{f}(S_2)$. Finally, we should prove that $\bar{f}$ is submodular. Let $S_1, S_2, \mathcal{N}_1$ and $\mathcal{N}_2$ be defined as before, let I be an interval that does not appear in S_2 and let S_I be the set of requests u for which $\mathcal{J}_u \cap S_i \neq \varnothing$. Then:

$$\begin{aligned}
\bar{f}(S_1 + I) - \bar{f}(S_1) &= f(\mathcal{N}_1 \cup S_I) - f(\mathcal{N}_1) \\
&= [f(\mathcal{N}_1 \cup S_I) - f(\mathcal{N}_1 \cup (S_I \cap \mathcal{N}_2))] + [f(\mathcal{N}_1 \cup (S_I \cap \mathcal{N}_2)) - f(\mathcal{N}_1)] \\
&\geq f(\mathcal{N}_1 \cup S_I) - f(\mathcal{N}_1 \cup (S_I \cap \mathcal{N}_2)) \geq f(\mathcal{N}_2 \cup S_I) - f(\mathcal{N}_2) \\
&= \bar{f}(S_2 + I) - \bar{f}(S_2) \ ,
\end{aligned}$$

where the first inequality follows from the monotonicity of f, and the second one from its submodularity. □

Corollary 6.2.19. ***Submodular Broadcast Scheduling*** *has a* $1/4$ *approximation algorithm.*

We would like to emphasis that the approximtion ratio given by Corollary 6.2.19 is the best approximation ratio known also for the linear variant of `Submodular Broadcast Scheduling` [8].

6.2.5 The `Submodular Matching Scheduling` Problem

The `Submodular Matching Scheduling` problem is defined as follows. We are given a multigraph $G = (V, E)$ of edge degree at most k and a set $\mathcal{J}_e$ of intervals *tuples* for every edge $e \in E$, where the number of intervals in each tuple of $\mathcal{J}_e$ is equal to the degree of

e. Given an intervals tuple $J \in \mathcal{J}_e$, one interval of J is associated with each end point of e. Let $E(J)$ be the edge with which the tuple J is associated, and let $J(u)$ denote the interval in J associated with node u (assuming $u \in E(J)$). A feasible matching schedule $S \subseteq \cup_{e \in E} \mathcal{J}_e$ is a set of intervals tuples such that for every two intervals tuples $J_1, J_2 \in S$, if $E(J_1)$ and $E(J_2)$ share a common vertex u, then $J_1(u)$ and $J_2(u)$ do not intersect. An edge e is *covered* by a matching schedule if $\mathcal{J}_e \cap S \neq \emptyset$. The objective is to find a matching schedule S maximizing a normalized monotone submodular function $f : 2^E \to \mathbb{R}^+$ over the set of edges that are covered by S.

Informally, a matching schedule is a matching in which every edge is also assigned a tuple of intervals, one for each end point, and we allow multiple edges to hit the same node as long as their associated intervals do not intersect. This problem has several interesting special cases:

1. The multigraph is a graph, *i.e.*, $k = 2$, and every intervals tuple contains two identical intervals. In this case, we can think of each edge e has having a given list $\mathcal{J}_e$ of allowed intervals. A feasible schedule chooses a subset E' of edges, and assigns an interval from $\mathcal{J}_e$ to every edge $e \in E'$ in such a way that the intervals assigned to edges intersecting a common vertex are do not intersect. This problem models a situation where nodes wish to transmit a set of messages among themselves, however, a node can participate at every point in time in one transmission only. The goal is to maximize a normalized monotone submodular function over the transmitted messages.

2. The input is a graph in which every node $v \in V$ has a budget $B(v) \in \mathbb{N}$, and every edge $e \in E$ has a weight w_e. A feasible schedule is a subset of edges such that for every vertex v, the total weight of the schedule's edges hitting v does not exceed $B(v)$. The goal is to maximize a normalized monotone submodular function over the schedule's edges.

   This problem has an approximation preserving pseudo polynomial time reduction to `Submodular Matching Scheduling`. This reduction uses the same ideas presented in the proofs of Section 6.2.3.

Lemma 6.2.20. *One can assume without loss of generality that for every edge $e \in E$, $|\mathcal{J}_e| = 1$.*

Proof. Replace every edge e with $|\mathcal{J}_e|$ parallel edges: $e_1, e_2, \ldots, e_{|\mathcal{J}_e|}$, and assign a unique intervals tuple from $\mathcal{J}_e$ to every edge e_i. The new objective function $\bar{f}$ is defined as following: $\bar{f}(S) = f(\{e \in E \mid \exists_i J_{e_i} \subseteq S\})$. Clearly $\bar{f}$ is also a normalized monotone submodular function. Observe that any feasible matching schedule before the reduction is also a feasible matching schedule after the reduction, and vice versa. Moreover, it is easy to see that the reduction preserves the value of the objective function because $\bar{f}$ is not effected by the size of $\mathcal{J}_e \cap S$ as long as this quantity is positive. □

After the reduction presented by Lemma 6.2.20 is applied, an instance of `Submodular Matching Scheduling` can be viewed as the intersection of n `Submodular Independent Set in Interval Graphs` instances, one for each node. Thus, we can apply the contention resolution scheme of `Submodular Independent Set in Interval Graphs` to each node separately, and get a scheme for `Submodular Matching Scheduling`. By Lemma 1.5 of [20], this yields a (b, e^{-bk})-monotone balanced contention resolution scheme for `Submodular Matching Scheduling`.

Corollary 6.2.21. *`Submodular Matching Scheduling` has a $ke^{-1}(k+1)^{-2}$ approximation algorithm.*

Proof. Using the contention resolution scheme described above, Theorem 6.0.3 and $b = \ln(1+1/k)$, we get an algorithm for `Submodular Matching Scheduling` with an approximation ratio of:

$$(1-e^{-b})\cdot e^{-bk} = \left(1-\frac{k}{k+1}\right)\cdot\left(\frac{k}{k+1}\right)^k = \frac{1}{k+1}\cdot\left(1-\frac{1}{k+1}\right)^k \geq \frac{k}{e(k+1)^2} \quad . \quad \square$$

Bibliography

[1] A. A. Ageev and M. I. Sviridenko. An 0.828 approximation algorithm for the uncapacitated facility location problem. *Discrete Appl. Math.*, 93:149–156, July 1999.

[2] A. A. Ageev and M. I. Sviridenko. Pipage rounding: a new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8(3):307–328, 2004.

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

[4] Noga Alon and Joel H. Spencer. *The Probabilistic Method*. Wiley Interscience Series in Discrete Mathematics and Optimization. John Wiley and Sons, Inc., second edition, 2000.

[5] Yossi Azar, Iftah Gamzu, and Ran Roth. Submodular max-sat. In *ESA*, pages 323–334, 2011.

[6] Nikhil Bansal, Nitish Korula, Viswanath Nagarajan, and Aravind Srinivasan. On k-column sparse packing programs. In Friedrich Eisenbrand and F. Shepherd, editors, *Integer Programming and Combinatorial Optimization*, volume 6080 of *Lecture Notes in Computer Science*, pages 369–382. Springer Berlin / Heidelberg, 2010.

[7] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph (Seffi) Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *J. ACM*, 48(5):1069–1090, 2001.

[8] Amotz Bar-Noy, Sudipto Guha, Yoav Katz, Joseph (Seffi) Naor, Baruch Schieber, and Hadas Shachnai. Throughput maximization of real-time scheduling with batching. *ACM Trans. Algorithms*, 5(2):1–17, 2009.

[9] Amotz Bar-Noy, Sudipto Guha, Joseph (Seffi) Naor, and Baruch Schieber. Approximating the throughput of multiple machines under real-time scheduling. *SIAM Journal on Computing (SICOMP)*, 31(2):331–352, September 2001.

[10] Richard A. Brualdi. Comments on bases in dependence structures. *Bull. of the Australian Math. Soc.*, 1(02):161–167, 1969.

[11] Richard A. Brualdi. Common transversals and strong exchange systems. *J. of Combinatorial Theory*, 8(3):307–329, April 1970.

[12] Richard A. Brualdi. Induced matroids. *Proc. of the American Math. Soc.*, 29:213–221, 1971.

[13] Richard A. Brualdi and Edward B. Scrimger. Exchange systems, matchings, and transversals. *J. of Combinatorial Theory*, 5(3):244–257, November 1968.

[14] Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *FOCS*, pages 649–658, 2012.

[15] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint. In *IPCO*, pages 182–196, 2007.

[16] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

[17] Deeparnab Chakrabarty and Gagan Goel. On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and gap. *SIAM J. Comput.*, 39(6):2189–2211, 2010.

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

[19] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Multi-budgeted matchings and matroid intersection via dependent rounding. In *SODA*, pages 1080–1097, 2011.

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

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

[22] M. Conforti and G. Cornuèjols. Submodular set functions, matroids and the greedy algorithm. tight worstcase bounds and some generalizations of the radoedmonds theorem. *Disc. Appl. Math.*, 7(3):251–274, 1984.

[23] G. Cornuejols, M. L. Fisher, and G. L. Nemhauser. Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. *Management Sciences*, 23:789–810, 1977.

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

[25] Shahar Dobzinski, Noam Nisan, and Michael Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. *Math. Oper. Res.*, 31(1):1–13, 2010.

[26] Shahar Dobzinski and Michael Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In *SODA*, pages 1064–1073, 2006.

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

[28] Shaddin Dughmi, Tim Roughgarden, and Mukund Sundararajan. Revenue submodularity. *Theory of Computing*, 8(1):95–119, 2012.

[29] Uriel Feige. On maximizing welfare when utility functions are subadditive. *SIAM J. Comput.*, 39(1):122–142, 2009.

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

[31] Uriel Feige and Jan Vondrák. Approximation algorithms for allocation problems: Improving the factor of $1 - 1/e$. In *FOCS*, pages 667–676, 2006.

[32] Uriel Feige and Jan Vondrák. The submodular welfare problem with demand queries. *Theory of Computing*, 6(1):247–290, 2010.

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

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

[35] Moran Feldman, Joseph (Seffi) Naor, Roy Schwartz, and Justin Ward. Improved approximations for k-exchange systems. In *ESA*, pages 784–798, 2011.

[36] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions – ii. In *Polyhedral Combinatorics*, volume 8 of *Mathematical Programming Studies*, pages 73–87. Springer Berlin Heidelberg, 1978.

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

[38] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

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

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

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

[42] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: offline and secretary algorithms. In *WINE*, pages 246–257. Springer-Verlag, 2010.

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

[44] Jason Hartline, Vahab Mirrokni, and Mukund Sundararajan. Optimal marketing strategies over social networks. In *WWW*, pages 189–198, 2008.

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

[46] D. Hausmann and B. Korte. K-greedy algorithms for independence systems. *Oper. Res. Ser. A-B*, 22(1):219–228, 1978.

[47] D. Hausmann, B. Korte, and T. Jenkyns. Worst case analysis of greedy type algorithms for independence systems. *Math. Prog. Study*, 12:120–131, 1980.

[48] Dorit S. Hochbaum. *Approximation algorithms for NP-hard problems*, chapter Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems, pages 94–143. PWS Publishing Co., Boston, MA, USA, 1997.

[49] C. A. J. Hurkens and A. Schrijver. On the size of systems of sets every t of which have an sdr, with an application to the worst case ratio of heuristics for packing problems. *SIAM J. Disc. Math.*, 2(1):68–72, 1989.

[50] T. Jenkyns. The efficacy of the greedy algorithm. *Cong. Num.*, 17:341–350, 1976.

[51] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

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

[53] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM J. Comput.*, 37:319–357, April 2007.

[54] Subhash Khot, Richard J. Lipton, Evangelos Markakis, and Aranyak Mehta. Inapproximability results for combinatorial auctions with submodular utility functions. *Algorithmica*, 52(1):3–18, 2008.

[55] S. Khuller, A. Moss, , and J. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

[56] B. Korte and D. Hausmann. An analysis of the greedy heuristic for independence systems. *Annals of Discrete Math.*, 2:65–74, 1978.

[57] Ariel Kulik, Hadas Shachnai, and Tami Tamir. Approximations for monotone and non-monotone submodular maximization with knapsack constraints. Manuscript, 2011.

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

[59] Eugene Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, New York, NY, USA, 1976.

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

[61] Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Maximizing non-monotone submodular functions under matroid or knapsack constraints. *SIAM Journal on Discrete Mathematics*, 23(4):2053–2078, 2010.

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

[63] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55(2):270–296, 2006.

[64] 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.

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

[66] Julián Mestre. Greedy in approximation algorithms. In *ESA*, pages 528–539, 2006.

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

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

[69] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[70] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge University Press, New York, NY, USA, 1995.

[71] 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.

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

[73] Noam Nisan and Ilya Segal. The communication requirements of efficient allocations and supporting prices. *Journal of Economic Theory*, 129:192–224, 2006.

[74] Alexander Schrijver. *Combinatorial Optimization, Polyhedra and Efficiency*. Springer, 2004.

[75] A. S. Schulz and N. A. Uhan. Approximating the least core and least core value of cooperative games with supermodular costs. To appear in Discrete Optimization, 2013.

[76] Hans Simon. Approximation algorithms for channel assignment in cellular radio networks. In J. Csirik, J. Demetrovics, and F. Gcseg, editors, *Fundamentals of Computation Theory*, volume 380 of *Lecture Notes in Computer Science*, pages 405–415. Springer Berlin / Heidelberg, 1989.

[77] Maxim Sviridenko. A note on maximizing a submodular set function subject to knapsack constraint. *Operations Research Letters*, 32:41–43, 2004.

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

[79] Jan Vondrák. personal communication.

[80] Jan Vondrák. *Submodularity in combinatorial optimization.* PhD thesis, Charles University, 2007.

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

[82] Justin Ward. A (k+3)/2-approximation algorithm for monotone submodular k-set packing and general k-exchange systems. In *STACS*, pages 42–53, 2012.

הקירוב הטובות ביותר הידועות לבעיות מיקסום תת-מודולריות שונות. התוצאה המפורסמת ביותר מבין תוצאות אלה היא הקירוב ההדוק אסימפטוטית של Calinescu et al. עבור מיקסום פונקציה תת-מודולרית מונוטונית תחת האילוץ שהפתרון חייב להיות קבוצה בלתי תלויה של מטרויד נתון (ללא הגבלות על סוג המטרואיד). השימוש בשיטות מבוססות הקלה מחייב פתרון שתי סוגיות מרכזיות. ראשית יש להגדיר הקלה של הבעיה אותה ניתן לפתור או לקרב בצורה יעילה. מכיוון שפונקצית המטרה אינה לינארית בבעיות אופטימיזציה תת-מודולרית, פתרון סוגיה זו אינו טריוואילי והוא נעשה בדרכים שונות בבעיות שונות. שנית, בהינתן פתרון שבור להקלה של הבעיה, יש למצוא שגרה המעגלת אותו בלי לאבד יותר מדי ערך בפונקצית המטרה. שני פרקים בתיזה זו עוסקים בסוגיות אלה. הפרק הראשון מבין השנים מתאר את האלגוריתם measured continuous greedy המשמש למציאת פתרון שבור עבור הקלות מולטילינאריות (multilinear relaxations) שהן הסוג הנפוץ של הקלות לבעיות מקסימיזציה תת-מודלרית. האלגוריתם משיג יחס קירוב משופר עבור הקלות של בעיות רבות. מעניין לציין שאלגוריתם זה הוא האלגוריתם הטוב ביותר הידוע הן עבור הקלות של בעיות בעלות פונקצית מטרה מונוטונית והן עבור הקלות של בעיות עם פונקצית מטרה לא מונוטונית. הפרק השני עוסק במספר הרחבות של שיטה כללית לעיגול פתרונות של הקלות הנקראת "עיגול על-ידי פתרון עימותים" (contention resolution scheme). הרחבה אחת מראה כיצד ניתן להתמודד במסגרת השיטה עם סוגים נוספים של אילוצים (כדוגמת אילוצי תזמון – scheduling). ההרחבה השניה מראה כי ניתן לשלב את השיטה באופן אורגאני עם האלגוריתם measured continuous greedy ולקבל תוצאות טובות יותר מאלו המתקבלות משילוב השיטה באופן נאיבי עם אלגוריתם כללי לפתרון הקלות בעל ביצועים דומים.

האינפורמציה (כלומר, כל אלגוריתם מדויק לבעיות אלה יחייב מספר אקספוננציאלי של פניות לאוב). במקום זאת, אנו מתארים אלגוריתמים עבור בעיות אלה המשיגים את יחס הקירוב הטוב ביותר הידוע. עבור מקצת הבעיות ניתן להראות שיחס הקירוב שאנו משיגים הוא הטוב ביותר שניתן להשיג.

אלגוריתמי קירוב לבעיות מקסימיזציה תת-מודולרית משתמשים בשיטות הנחלקות לשני סוגים עיקריים. הסוג הראשון כולל שיטות קומבינטוריות שהעיקריות מביניהן הן : חיפוש מקומי (local search) וכללים חמדניים (greedy rules). השימוש בסוג זה של שיטות עבור בעיות מקסימיזציה תת-מודולרית החל כבר בתחילת שנות ה-70 עבור בעיות המבקשות למקסם פונקצית מטרה מונוטונית ותת-מודולרית תחת האילוץ שהפתרון חייב להיות קבוצה בלתי תלויה של מטרואיד נתון. הדוגמא הפשוטה ביותר לשימוש כזה היא ההוכחה של Fisher et al. המראה כי האלגוריתם החמדן הנאיבי המוסיף לפתרון בכל צעד את האיבר עם התרומה השולית הגדולה ביותר הוא אופטימאלי עבור מטרואידים אחידים (במילים אחרות, הפתרון יכול להכיל עד k איברים, כאשר k הוא חלק מהקלט). לאחרונה השימוש בסוג זה של שיטות הורחב לפונקצית מטרה שאינן מונוטוניות ולסוגים נוספים של אילוצים (למשל הפתרון חייב להיות קבוצה בלתי תלויה בשני מטרואידים שונים).

שני פרקים בתיזה זו מתארים אלגוריתמים קומבינטוריים. הפרק הראשון מבין השניים מתאר אלגוריתם בעל יחס קירוב $1/2$ וזמן ריצה ליניארי עבור הבעיה של מיקסום פונקציה תת-מודולרית ללא אילוצים (במילים אחרות, בהינתן פונקציה תת-מודולרית f יש למצוא תת-קבוצה כלשהי של קבוצת הבסיס הממקסמת את $f(S)$). בשל השימושים הרבים של בעיה בסיסית זו היא זכתה לשורה ארוכה של מחקרים שתיארו אלגוריתמים בעלי יחסי קירוב הולכים ומשתפרים עבורה. האלגוריתם שלנו מסיים שורה זו של מחקרים מכיוון שהוא אופטימאלי הן מבחינת יחס הקירוב והן מבחינת זמן הריצה. בנוסף לאלגוריתם זה, הפרק מתאר אלגוריתמים אופטימאלים גם לבעיות Submodular SAT ו-Submodular Welfare עם שני שחקנים. הפרק השני מתאר אלגוריתם חיפוש מקומי עבור הבעיה של מיקסום פונקצית מטרה תת-מודולרית תחת האילוץ שהפתרון חייב להיות קבוצה בלתי תלויה של מערכת קבוצות (set system) מסוג k-exchange. מערכות קבוצות מסוג זה מכלילות מבנים כדוגמת שידוכים, b-שידוכים, שידוכים ברב-גרפים וחיתוכים של מטרואידים בעלי בסיסים הניתנים לסידור במובן החזק (strongly base orderable matroids). מעניין לציין כי Lee et al. הראו כי אלגוריתם זה משיג את אותן תוצאות קירוב גם עבור מערכות קבוצות מסוג k-intersection, אך ההוכחה לכך שונה מאוד ומסתמכת בצורה חזקה על תכונות של מטרואידים. מציאת הוכחה שתעבוד עבור שני סוגי מערכות הקבוצות היא שאלה פתוחה.

הסוג השני של שיטות המשמשות באלגוריתמי קירוב לבעיות מקסימיזציה תת-מודולריות דומה לשיטה הסטנדרטית לבנית אלגוריתמי קירוב באמצעות הקלות (relaxations). בכל שיטה מסוג זה ישנם שני שלבים. בשלב הראשון מחושב פיתרון שבור עבור הקלה של הבעיה. בשלב השני הפתרון השבור מעוגל לפתרון שלם תוך ירידה חסומה בלבד של ערך פונקצית המטרה. למרות האופי הקומבינטורי של פונקציות תת-מודולריות, סוג זה של שיטות שימש לגילוי רבות מתוצאות

תקציר

המחקר של בעיות קומבינטוריות עם פונקציות מטרה תת-מודולריות (submodular) מושך אליו תשומת לב מרובה בעת האחרונה. פונקציות תת-מודולריות מיצגות את העיקרון של "יתרון לגודל" (economy of scale) הנפוץ בבעיות שמקורן "בעולם האמיתי". יתר על כן, פונקציות תת-מודולריות משמשות לעיתים קרובות כפונקציות מטרה בכלכלה ובתורת המשחקים האלגוריתמית. מנקודת מבט תיאורטית, פונקציות תת-מודולריות ואופטימיזציה תת-מודלרית הן בעלות תפקיד חשוב בקומבינטוריקה, תורת הגרפים ואופטימיזציה קומבינטורית.

פונקצית קבוצות (set function) היא פונקציה המעניקה ערך מספרי לכל תת-קבוצה של קבוצת בסיס (ground set) נתונה. פונקציה תת-מודולרית היא פונקצית קבוצות המקיימת את התכונה הבאה: התרומה השולית של הוספת איבר לקבוצה אינה עולה כאשר נוספים איברים אחרים לקבוצה. לעיתים קרובות מתייחסים לפונקציות תת-מודולריות כמקבילה הבדידה של פונקציות קעורות. למרות זאת, פונקציות תת-מודולריות מגלות תכונות קמורות וקעורות גם יחד. להלן מספר דוגמאות ידועות לפונקציות תת-מודולריות: חתכים בגרפים מכוונים ובלתי מכוונים, פונקציות דרגה של מטרואידים (matroid), פונקציות כיסוי וחתכים בהיפרגרפים (hypergraph).

ישנן מספר תכונות שימושיות נוספות שפונקצית קבוצות עשויה לקיים. פונקציה מנורמלת (normalized) היא פונקציה הנותנת את הערך 0 לקבוצה הריקה. פונקציה מונוטונית (monotone) היא פונקציה הנותנת לקבוצה A ערך גדול לפחות כמו לקבוצה B אם A מכילה את B. לבסוף, פונקציה היא אי-שלילית אם היא מעניקה ערך אי-שלילי לכל קבוצה. ניתן לראות כי פונקציה מנורמלת מונוטונית היא תמיד אי-שלילית.

בבעיות אופטימיזציה תת-מודולרית המטרה היא למצוא קבוצה המביאה פונקציה תת-מודלרית נתונה למקסימום או למינימום. הבעיות נבדלות זו מזו באילוצים על הקבוצות המותרות כפתרונות לבעיה, ובתכונות שעל פונקצית המטרה לקיים בנוסף לתת-מודולריות (כדוגמת מונוטוניות). ניתן ליצג בעיות אופטימיזציה רבות באמצעות בעיות אופטימיזציה תת-מודולרית. לדוגמה הבעיות הבאות הן בעיות ידועות שנחקרו רבות וניתן ליצגן באמצעות בעיות אופטימיזציה תת-מודולרית: Max Cut, Max k-Cover, Generalized Assignment, מספר וריאנטים של Max SAT ומספר בעיות תזמון (scheduling) ורווחה (welfare). גילוי אלגוריתמים בעלי יחסי קירוב טובים לבעיות אופטימיזציה תת-מודולרית ישרה אלגוריתמים דומים לבעיות שהוצגו לעיל ולבעיות רבות נוספות. יתר על כן, מאחר שרבות מבעיות שהוצגו לעיל מופיעות לעיתים קרובות בעולם האמיתי, חשוב למצוא להן אלגוריתמים שהם גם בעלי יחס קירוב טוב וגם יעילים.

בתיזה זו אנו חוקרים מספר בעיות אופטימיזציה תת-מודלרית. לכל הבעיות שנחקור ישנו אותו מבנה כללי. בהינתן פונקצית מטרה f וקבוצת אילוצים (לעיתים ריקה), יש למצוא קבוצה S הממקסמת את $f(S)$ ומקיימת את האילוצים. יצוג מפורש של הפונקציה f עשוי להיות אקספוננציאלי בגודל קבוצת הבסיס. לכן, אנו מניחים כי הגישה ל-f מתבצעת באמצעות אוב. בהינתן תת-קבוצה S של קבוצת הבסיס, האוב מחשב את $f(S)$. כל הבעיות שנחקור בתיזה זו אינן ניתנות לפתרון מדויק בזמן פולינומי בשל תוצאות קושי הנובעות משיקולים של תורת

המחקר נעשה בהנחיית פרופסור ספי נאור מהפקולטה למדעי-המחשב.

ברצוני להודות לספי, המנחה שלי, שהוביל אותי בחוכמה והכווין אותי בנועם במשך כל שנות עבודתי. על הכוונה לנושאי מחקר מעניינים וחדשים. על עבודה משותפת לתוך הלילה כאשר תאריך יעד התקרב. על הצגתי בפני חוקרים מעניינים נוספים. ומעל הכל, על תמיכתך הרבה והעידוד, בעיקר כאשר מאמרים נדחו פעם אחר פעם. אני מעריך מאוד את ההזדמנות שנפלה בחלקי לעבוד איתך.

אני מודה למלגת גוגל האירופאית לחקר "אלגוריתמי שוק" ולטכניון על התמיכה הכספית הנדיבה בהשתלמותי.

רשימת פרסומים:

1. Moran Feldman, Joseph(Seffi) Naor, Roy Schwartz and Justin Ward, "Improved approximations for k-exchange systems"', In *Proceedings of the 19th Annual European Symposium on Algorithms, ESA11*, Saarbrücken, Germany, 2011.

2. Moran Feldman, Joseph(Seffi) Naor and Roy Schwartz, "A unified continuous greedy algorithm for submodular maximization.", In *Proceedings of the 52nd annual IEEE Symposium on Foundations of Computer Science, FOCS11*, Palm Springs, California, USA, 2011.

3. Niv Buchbinder, Moran Feldman, Joseph(Seffi) Naor and Roy Schwartz, "A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization."', In *Proceedings of the 53nd annual IEEE Symposium on Foundations of Computer Science, FOCS12*, New Brunswick, New Jersey, USA, 2012.

בעיות מקסימיזציה עם פונקציית מטרה תת-מודולרית

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
דוקטור לפילוסופיה

מורן פלדמן

הוגש לסנט הטכניון – מכון טכנולוגי לישראל

תמוז התשע״ג חיפה יוני 2013

בעיות מקסימיזציה עם פונקציית מטרה תת-מודולרית

מורן פלדמן