Arithmetic Combinations of Submodular and Supermodular Optimization and Submodular Generalized Matching for Peptide Identification in Tandem Mass Spectrometry

Wenruo Bai

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2023

Reading Committee:

Jeff Bilmes, Chair

Maryam Fazel

Kevin Jamieson

Program Authorized to Offer Degree:
Department of Electrical and Computer Engineering

University of Washington

Abstract

Arithmetic Combinations of Submodular and Supermodular Optimization and Submodular Generalized Matching for Peptide Identification in Tandem Mass Spectrometry

Wenruo Bai

Chair of the Supervisory Committee:
Jeff Bilmes
Department of Electrical and Computer Engineering

Submodular functions have recently shown utility for a number of machine learning applications such as information gathering, document summarization, image segmentation, and string alignment, since they are natural for modeling concepts such as diversity, information, and representativeness. Submodular optimization problems are widely studied under different scenarios, such as submodular minimization without constraints or submodular maximization under a cardinality constraint. However, in real-world applications, the objective function is usually not a simple submodular function (or supermodular function) but is naturally written as an arithmetic combination of submodular and/or supermodular functions. For the **first part** of this thesis work, we study the property of the widest arithmetic combinations of submodular and supermodular functions and how we can optimize them. The content includes *sums* $f_1 + g_1$, *divisions* f_1/f_2, f_1/g_1, g_1/f_1, g_1/g_2, *products* $f_1 * f_2$ and *p-norms* $f_1^p + f_2^p$, where f and g donates submodular and supermodular respectively. We study the novel optimization problems on these non-submodular functions and propose algorithms to achieve tight approximation guarantees under various constraints. This greatly expands the study on submodular and non-submodular optimizations and draws a rather complete picture of optimizing combinations of submodular and supermodular functions.

For the **second part** of this thesis work, we focus on a biological application on identi-

fication of spectra produced by a shotgun proteomics mass spectrometry experiment using submodular generalized matchings. This is commonly performed by searching the observed spectra against a peptide database. The heart of this search procedure is a score function that evaluates the quality of a hypothesized match between an observed spectrum and a theoretical spectrum corresponding to a particular peptide sequence. Accordingly, the success of a spectrum analysis pipeline depends critically upon this peptide-spectrum score function. We develop peptide-spectrum score functions that compute the maximum value of a submodular function under m matroid constraints. We call this procedure a *submodular generalized matching* (SGM) since it generalizes bipartite matching. We use a greedy algorithm to compute maximization, which can achieve a solution whose objective is guaranteed to be at least $\frac{1}{1+m}$ of the true optimum. The advantage of the SGM framework is that known long-range properties of experimental spectra can be modeled by designing suitable submodular functions and matroid constraints. Experiments on four data sets from various organisms and mass spectrometry platforms show that the SGM approach leads to significantly improved performance compared to several state-of-the-art methods.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to all those who have supported me throughout this journey of pursuing my PhD degree.

First, I would like to express my sincerest appreciation to my advisor, Prof. Jeff Bilmes, for his support, guidance, and encouragement throughout my research journey. Jeff led me to the world of submodular and machine learning. He taught me classical knowledge in class, introduced me to frontier research problems, and showed me his amazing ideas in the realm of submodular and machine learning. Whenever I got any questions or got lost in my research project, Jeff can always answer my questions and point me in the right direction. I can not count how many times Jeff's mathematical intuition and data sensitivity inspired me during my journey toward the Ph.D. study. Jeff also helped me build the skills to be a professional researcher. He taught me how to find and read classical or latest research papers, how to be strict in mathematical proofs, and how to do machine learning experiments properly. He also taught me how to make good slides, write research papers using latex and write quality codes in C++ and python. Jeff is strict in details and, at enormous times, he corrected the bugs and typos, and told me how to improve in my proofs, codes, writing, or slides. But that 'strictness' is just how a researcher becomes 'professional'. This is an endless wealth that I can benefit from in my future career as a researcher. Thank you, Jeff!

I am grateful to the members of my thesis committee, Prof. Maryam Fazel, and Prof. Kevin Jamieson and Prof. Emilija Perkovic, for their critical feedback, insightful comments, and valuable suggestions that have significantly improved the quality of my thesis.

I would also like to extend my gratitude to my collaborators, Shengjie Wang, Tianyi Zhou, Chandrashekhar Lavania, Wei Kai, Rishabh Iyer, John Halloran, Sunil Thulasidasan,

Lily Kumari, Arnav Das, Gantavya Bhatt, Megh Bhalerao, Harshil Dadlani and Armagan Er in MELODI lab. I could still remember the discussions on whiteboards, group meetings, and zooms, how they shared their knowledge in submodular and machine learning with me, and how their words encouraged me throughout my studies. Also a huge thanks to Prof. William Noble, Alex Hu, Yang Lu, and Haoran Cai in UW Noble research lab, who taught and helped me explore the amazing world of peptides and mass spectrometry. Without everyone's helps, it is impossible for me to complete this long journey.

My sincere thanks go to my family: my love Zhaokang and my parents Chenggang and Yujie, who have always been my source of inspiration and motivation. Their unwavering support and understanding throughout this journey have been instrumental in helping me to stay focused and motivated.

Thank you all for your support and guidance, without which this achievement would not have been possible.

DEDICATION

to Zhaokang Li and my parents Chenggang Bai and Yujie Sun.

Part I

OPTIMIZATION OF ARITHMETIC COMBINATIONS OF SUBMODULAR AND SUPERMODULAR FUNCTIONS

Part I focuses on the theoretical study of the optimization problems of arithmetic combinations of submodular and supermodular functions. In the first chapter, we introduce the concept of submodular and supermodular functions and maximization and minimization problems. We further discuss the motivations for studying the arithmetic combinations of submodular and supermodular functions.

In chapter 2, we discuss the sum of submodular and supermodular functions. The work is published in **Wenruo Bai** *and Jeff Bilmes. Greed is Still Good: Maximizing Monotone Submodular+Supermodular (BP) Functions. In International Conference on Machine Learning (ICML), Stockholm, Sweden, July 2018.* and **Wenruo Bai** *and Jeffrey Bilmes. Greed is Still Good: Maximizing Monotone Submodular+Supermodular Functions. Arxiv, abs/1801.07413, Jan 2018.*

In chapter 3, we discuss the ratio of submodular and supermodular functions. The work is published in **Wenruo Bai**, *Rishabh Iyer, Kai Wei, and Jeff Bilmes. Algorithms for Optimizing the Ratio of Submodular Functions. In International Conference on Machine Learning (ICML), New York, NY, July 2016.*

In chapter 4, we discuss the product and generalized mean of submodular functions. The work is from an unpublished paper **Wenruo Bai**, *Shengjie Wang, Tianyi Zhou, and Jeff Bilmes, Optimizing Means of Submodular Functions.*

All these optimization problems are discussed with applications, algorithms, guarantees, hardness, and synthetic data experiments.

Chapter 1

PRELIMINARIES: SUBMODULAR OPTIMIZATION

1.1 Submodular and Supermodular Function

Defined over an underlying ground set V, a set function $f : 2^V \to \mathbb{R}$ is said to be submodular if and only if for all subsets $X, Y \subseteq V$,

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y) \tag{1.1}$$

The ground set V is not empty and has a finite number of elements, and throughout the thesis work, we use $n = |V|$ unless otherwise stated. An equivalent definition for submodularity is for all subsets $X \subseteq Y \subset V$ and for all $v \in V \setminus Y$, $f(v|X) \geq f(v|Y)$, where $f(v|X) = f(\{v\} \cup X) - f(X)$ is a gain notation. This definition is also called *diminishing return*, which is the most fundamental property of submodular. A set function f is called monotonic non-decreasing or non-increasing if and only if $f(v|X) \geq 0$ or $f(v|X) \leq 0$ for all $X \subset V$ and $v \in V \setminus X$, respectively. f is called normalized if and only if $f(\emptyset) = 0$. A normalized monotonic non-decreasing submodular function has the *subadditive* property, i.e. $f(X) + f(Y) \geq f(X \cup Y)$ for all $X, Y \subseteq V$. Note that subadditive is different from submodular.

For the counterpart of submodularity, a set function $g : 2^V \to \mathbb{R}$ is called supermodular if $-g$ is submodular. For all subsets $X, Y \subseteq V$,

$$g(X) + g(Y) \leq g(X \cup Y) + g(X \cap Y) \tag{1.2}$$

And similarly, for all subsets $X \subseteq Y \subset V$ and for all $v \in V \setminus Y$, $g(v|X) \leq g(v|Y)$.

A set function $m : 2^V \to V$ is called a modular function if m is both submodular and supermodular. A normalized modular function can be seen as a dot product $m(X) = \mathbf{1}_X \cdot \mathbf{m}$, where $\mathbf{1}_X, \mathbf{m} \in \mathbb{R}^n$ and $\mathbf{1}_{X,i} = 1$ for index i if $i \in X$ and $\mathbf{1}_{X,i} = 0$ otherwise.

1.2 Set Function Optimization

Given a set function $h(S)$, we study the optimization problems

Problem 1. Set function maximization $\max_{S\subseteq V, S\in\mathcal{C}} h(X)$, *where* $\mathcal{C}$ *is a constraint.*

Problem 2. Set function minimization $\min_{S\subseteq V, S\in\mathcal{C}} h(X)$, *where* $\mathcal{C}$ *is a constraint.*

Usually, we refer $S^* \in \operatorname{argmax}_{S\subseteq V, S\in\mathcal{C}} h(X)$ as the optimal solution for Problem 1; or $S^* \in \operatorname{argmin}_{S\subseteq V, S\in\mathcal{C}} h(X)$ for Problem 2 respectively. Let $\mathrm{OPT} = h(S^*)$. Note that there is a trivial algorithm to search all possible subsets of V, and then find S^* and OPT. However, this requires an exponential number of queries to the size of the ground set and thus is not feasible in practice.

In many applications, it is acceptable to find an approximation solution $\hat{S}$, such as $h(\hat{S}) \geq \gamma \cdot \mathrm{OPT}$ for maximization or $h(\hat{S}) \leq \gamma \cdot \mathrm{OPT}$ for minimization. If an algorithm can always find such a solution, then γ is called a *guarantee* of this algorithm. Note that there is an assumption that $\mathrm{OPT} \geq 0$, otherwise the *guarantee* is not well defined. For non-deterministic algorithms, a guarantee can be defined as $\mathbf{E}h(\hat{S}) \geq \gamma h(\mathrm{OPT})$ for maximization, or $\mathbf{E}h(\hat{S}) \leq \gamma h(\mathrm{OPT})$ for minimization. There are other ways to define a *guarantee*, for example, $h(\hat{S}) \geq \mathrm{OPT} - \gamma$, but those are beyond the discussion in this thesis work.

Another important assumption is that we can only access the oracle call of function values of the objective function (or separate components which we will discuss more in chapter 2). For example, if we know the exact mathematical formula of the set function such as $h(S) = |S| + 1/|S|$, then it might be trivial to solve the optimization problem without calling the function even once. So we assume the set function is a black box and we can analyze it using a selected input set and get an output real number. Later in chapter 2, we may expand the definition, such as $h(S) = f(S) + g(S)$, where we get oracle values of f and g separately.

In general, a set function $h : 2^V \to \mathbb{R}$ can not be approximated to any positive factor in either maximization or minimization problem by polynomial oracle calls. An example is

$h(S) = 0$ for all $S \neq T$ and $h(T) = 1$ for some random set $T \subseteq V$. Note we only have oracle access, so we are not 100% sure to find T unless we search all 2^n subsets. Any polynomial algorithm can only guarantee to find $\hat{S}$ such as $h(\hat{S}) = 0$, so *guarantee* is 0. Moreover, the guarantee of any polynomial non-deterministic algorithm is at most $\frac{O(n^m)}{2^n}$, which can be arbitrarily close to 0.

However, given the properties of modularity, submodularity, or supermodularity, it is possible to solve or approximate the optimization problems. We will first summarize the known results of submodular/supermodular optimization problems and then discuss how we can generalize them.

1.3 Known Results of Submodular and Supermodular Optimization

Note that $-f$ is supermodular when f is submodular and $\max f$ can be written as $-\min -f$. So every supermodular optimization problem can be written as an equivalent submodular optimization problem and vice versa. In general, we won't need to study supermodular if we fully understand submodular. However, as discussed in section 1.2, we want to study the case when OPT ≥ 0 so that the concept of *guarantee* is well defined. In this case, submodular and supermodular optimizations are distinct nontrivial problems.

In this thesis work, unless otherwise specified, we focus on non-negative set functions. In this perspective, we will not treat supermodular functions simply as negative submodular functions, but also study the unique properties of these classes of functions, as well as an arithmetic combination of submodular and supermodular functions.

Note in this section, we only introduce the most famous and fundamental results for submodular and supermodular optimizations and aim to present the overall picture to the readers, such as which problems are solvable or approximatable. Later in the main part of this thesis work, we will introduce more recent work on different optimizations problem and compare it with our novel results.

1.3.1 maximization and minimization problem

$$\max_{S\in V, S\in\mathcal{C}} h(S) \textbf{ or } \min_{S\in V, S\in\mathcal{C}} h(S) \tag{1.3}$$

where h is modular, submodular or supermodular, and $\mathcal{C}$ is a certain kind of constraint. As a notation, let $h = m, f, g$ if h is modular, submodular, or supermodular, respectively. The interesting constraints are

- unconstrained, i.e. $\mathcal{C} = \{S \subseteq V\}$.
- cardinality constraints, i.e. $\mathcal{C} = \big\{S \subseteq V \big| |S| \leq k\big\}$.
- matroid constraints. A *matroid* $M = (V, \mathcal{I})$ is a pair consisting of a ground set V and a set of subsets $\mathcal{I} = (I_1, I_2, \dots)$ where $I_i \subseteq V$ for all i. The subsets are said to be "independent" and to be a matroid if the subsets satisfy certain properties. Specifically, the pair $M = (V, \mathcal{I})$ is a matroid if it satisfies (i) $\emptyset \in \mathcal{I}$; (ii) $A \subset B \in \mathcal{I}$ implies that $A \in \mathcal{I}$; and (iii) given $A, B \in \mathcal{I}$ with $|A| > |B|$ then there exists $x \in A \setminus B$ such that $B \cup \{x\} \in \mathcal{I}$. Matroids are extremely powerful combinatorial objects, despite their simple definition, and have undergone years of mathematical study [99].
- $\mathcal{C} = \{S \subseteq V | f'(S) \leq k\}$, where f' is another submodular function.

Unconstrained supermodular function maximization can be exactly solved. This is equivalent to the well-known **unconstrained submodular function minimization** problem [19], since minus submodular function $-f$ is supermodular. If we further assume all the function values are integers, then the problem can be exactly solved in strong or weak polynomial time.

Cardinality constrained submodular function maximization can be approximated to $1/2$ [12]. A more famous result is to further assume f is monotonic non-decreasing, then **cardinality constrained monotonic non-decreasing submodular function maximization** problem can be approximated to a factor of $1 - 1/e$ or $\kappa_f \left(1 - e^{-\kappa_f}\right)$ using a

simple **greedy algorithm** [95, 38], where $\kappa_f = 1 - \min_{v \in V} \frac{f(v|V\setminus\{v\})}{f(v|\emptyset)}$ is called the submodular curvature. The greedy algorithm is also one of the key topics of this thesis, and we will discuss the cases that a greedy or a slightly modified greedy algorithm can still work for other non-submodular optimization problems and the cases that greedy won't work.

Matroid constrained submodular function maximization can be approximated to $1 - 1/e$ for one matroid case using a randomized algorithm [14]. In the case of p matroids, The greedy algorithm can obtain a $1/p+1$ approximation gurantee [37].

Cardinality constrained supermodular function maximization is hard to solve, even we assume g is monotonic non-decreasing. In fact, the problem can not be approximated to any positive factor in polynomial time to the size of the ground set V. We can demonstrate it in a simple example, $g(S) = 0$ when $|S| < k$ and $g(S) = 2|S| - 2k$ when $|S| \geq k$ and $S \neq R$ and $g(R) = 1$, where R is an arbitrary set with size equal to k. We can show g is supermodular and $\max_{|S|\leq k} g(S) = 1$ but it is not possible to find the arbitrary set R in any polynomial oracle call to the function value. Any such algorithm can only find a set with value 0. Similarly, **matroid constrained supermodular function maximization** is inapproximable. However, in a later section, we discuss the concept of supermodular curvature and show the problem, and a more general submodular + supermodular maximization problem can be approximated if supermodular curvature is less than 1.

1.4 Arithmetic Combinations of Submodular and Supermodular Functions

Arithmetic combinations of submodular and supermodular function includes *sums* $f_1 + g_1$, *divisions* f_1/f_2, f_1/g_1, g_1/f_1, g_1/g_2, *products* $f_1 * f_2$ or *p-norms* $f_1^p + f_2^p$, where f and g donates submodular and supermodular respectively. In the first part of this thesis work, we focus on the optimization problem of these forms. Note that, we always assume we have oracle calls of the submodular or supermodular functions individually.

A natural question is **why do we want to study these functions**? The answers are:

- **The problems themselves are theoretically interesting**. Imaging we just learned

$\sin(x)$ and $\cos(x)$, the next question is always, what is the property of $\sin(x) + \cos(x)$, $\sin(x) \cdot \cos(x)$, $\sin(x)/\cos(x)$ or $\sin^2(x) + \cos^2(x)$? These questions happen in the realm of submodularity and supermodularity as well. Studying the arithmetic combinations of submodular and supermodular functions will greatly expand our understanding of submodularity and supermodularity. We do show that most of the introduced optimization problems are solvable or approximatable to a constant guarantee (some are related to curvature), using algorithms with polynomial running time. We also showed the hardness and tightness of the problems. Major results are summarized in Table 2.1 and Table 4.1.1.

- **Arithmetic combinations of submodular and supermodular functions naturally occur in many real-world applications.** For example, when choosing a subset of training data in a machine learning system, the submodular function has the ability to measure *redundancies*, but it can only diminish, rather than enhance. There are cases where the full collective utility of these elements is seen only when utilized together; and this *complementary* property can be measured by supermodular set functions. When we want to optimize both properties, it is natural to introduce an objective function that is a sum of a submodular function and a supermodular function (BP function). Another example is F-measure, which is the harmonic mean of the precision and recall and can be written as a ratio of submodular and a modular function (see Section 3.1.1). This introduces the optimization problem of a Ratio function. A third example is the robust maximization of two submodular functions, like $\max_S \min(f_1(S), f_2(S))$. But in real applications, the objective is fully dominated by either $f_1(S)$ or $f_2(S)$, and the incremental gain of the larger one does not add to the final function value. We find that the product function $f_1(S) \cdot f_2(S)$ is a better objective function where it can measure robustness while avoiding saturation. Robustness means the optimal solution should perform well on both $f_1(S)$ and $f_2(S)$; e.g. S_1 s.t $f_1(S_1) = 2$ and $f_2(S_1) = 2$ gets higher product value than S_2 s.t. $f_1(S_2) = 0.1$ and

$f_2(S_2) = 10$. On the other hand, saturation can be prevented, that even in the case of S_2, an element $v \notin S_2$ s.t. $f_1(v|S_2) = 0$ and $f_2(v|S_2) = 10$ gets positive gain on $f_1 \cdot f_2$, while the gain is 0 for $\min(f_1, f_2)$. These three examples lead to the three chapters in the first part of this thesis work.

1.5 Greedy Algorithm

The *Greedy algorithm* [7, 16] is a technique in combinatorial optimization that makes a locally optimal choice at each stage in the hope of finding a good global solution. It is one of the simplest, most widely applied, and most successful algorithms in practice [69, 139, 66, 106, 136]. Due to its simplicity, and low time and memory complexities, it is used empirically even when no guarantees are known to exist although, being inherently myopic, the greedy algorithm's final solution can be arbitrarily far from the optimum solution [6].

On the other hand, there are results going back many years showing where the greedy algorithm is, or almost is, optimal, including Huffman coding [51], linear programming [25, 22], minimum spanning trees [76, 104], partially ordered sets [30, 22], matroids [27, 24], greedoids [72], and so on, perhaps culminating in the association between the greedy algorithm and submodular functions [26, 96, 15, 39].

Algorithm 1: GreedMax for submodular maximization

Input: f, g and constraint set $\mathcal{C}$.
Output: An approximation solution $\hat{X}$.
Initialize: $X_0 \leftarrow \emptyset$, $i \leftarrow 0$ and $R \leftarrow V$
while $\exists v \in R$ s.t. $X_i \cup v \in \mathcal{C}$ **do**
    $v \in \text{argmax}_{v \in R, X_i \cup v \in \mathcal{C}} f(v|X_i)$.
    $X_{i+1} \leftarrow X_i \cup v$.
    $R \leftarrow R \setminus v$.
    $i \leftarrow i + 1$.
end while
Return $\hat{X} \leftarrow X_i$.

The greedy algorithm is the major weapon for optimization problems on the arithmetic combinations of submodular and supermodular functions. In the following chapters, we proposed the greedy algorithm or modified versions of the greedy algorithm to produce constant guarantees on various optimization problems.

Chapter 2

MAXIMIZING MONOTONE SUBMODULAR+SUPERMODULAR FUNCTIONS

We analyze the performance of the greedy algorithm, and also a discrete semi-gradient based algorithm, for maximizing the sum of a suBmodular and a suPermodular (BP) function (both of which are non-negative monotone non-decreasing) under two types of constraints, either a cardinality constraint or $p \geq 1$ matroid independence constraints. These problems occur naturally in several real-world applications in data science, machine learning, and artificial intelligence. The problems are ordinarily inapproximable to any factor (as we show). Using the curvature κ_f of the submodular term, and introducing κ^g for the supermodular term (a natural dual curvature for supermodular functions), however, both of which are computable in linear time, we show that BP maximization can be efficiently approximated by both the greedy and the semi-gradient based algorithm. The algorithms yield multiplicative guarantees of $\frac{1}{\kappa_f}\left[1 - e^{-(1-\kappa^g)\kappa_f}\right]$ and $\frac{1-\kappa^g}{(1-\kappa^g)\kappa_f+p}$ for the two types of constraints respectively. For pure monotone supermodular constrained maximization, these yield $1-\kappa^g$ and $(1-\kappa^g)/p$ for the two types of constraints respectively. We also analyze the hardness of BP maximization and show that our guarantees match hardness by a constant factor and by $O(\ln(p))$ respectively. Computational experiments are also provided to support our analysis. This work is published in

Wenruo Bai *and Jeff Bilmes. Greed is Still Good: Maximizing Monotone Submodular+Supermodular (BP) Functions. In International Conference on Machine Learning (ICML), Stockholm, Sweden, July 2018.*

Wenruo Bai *and Jeffrey Bilmes. Greed is Still Good: Maximizing Monotone Submodular+Supermodular Functions. Arxiv, abs/1801.07413, Jan 2018.*

2.1 Introduction

Certain subset selection problems in data science are not purely submodular. For example, when choosing a subset of training data in a machine learning system [132], there might be not only redundancies but also complementarities amongst certain subsets of elements, where the full collective utility of these elements is seen only when utilized together. Submodular functions can only diminish, rather than enhance, the utility of a data item in the presence of other data items. Supermodular set functions can model such phenomena, and are widely utilized in economics and social sciences, where the notion of complementary [120] is naturally needed, but are studied and utilized less frequently in machine learning.

In this chapter, we advance the state of the art in understanding when the greedy (and the semigradient) algorithm offers a guarantee, in particular for approximating the constrained maximization of an objective that may be decomposed into the sum of a submodular and a supermodular function (applications are given in Section 2.1.1). That is, we consider the following problem

Problem 3. BP function maximization $max_{X \in \mathcal{C}} h(X) := f(X) + g(X)$, *where* $\mathcal{C} \subseteq 2^V$ *is a family of feasible sets,* f *and* g *are normalized (*$f(\emptyset) = 0$*), monotonic non-decreasing (*$f(\{s\}|S) \geq 0$ *for any* $s \in V$ *and* $S \subseteq V$*) submodular and supermodular functions respectively*[1] *and hence are non-negative.*

We call this problem *suBmodular-suPermodular* (BP) maximization, and $f + g$ a *BP function*, and we say h admits a BP decomposition if $\exists f, g$ such that $h = f + g$ where f and g are defined as above. In the chapter, the set $\mathcal{C}$ may correspond either to a cardinality constraint (i.e., $\mathcal{C} = \{A \subseteq V \mid |A| \leq k\}$ for some $k \geq 0$), or alternatively, a more general case where $\mathcal{C}$ is defined as the intersection of p matroids. Hence, we may have $\mathcal{C} = \{X \subseteq V \mid X \in \mathcal{I}_1 \cap \mathcal{I}_2 \cap \cdots \cap \mathcal{I}_p\}$, where $\mathcal{I}_i$ is the set of independent sets for the ith matroid $\mathcal{M}_i = (V, \mathcal{I}_i)$.

[1] Throughout, f & g are assumed monotonic non-decreasing submodular/submodular functions respectively.

The performance of the greedy algorithm for some special cases of BP maximization has been studied before. For example, when $g(X)$ is modular, the problem reduces to submodular maximization where, if f and g are also monotone, the greedy algorithm is guaranteed to obtain an $1 - 1/e$ approximate solution under a cardinality constraint [95] and $1/p+1$ for p matroids [38, 15]. The greedy algorithm often does much better than this in practice. Correspondingly, the bounds can be significantly improved if we also make further assumptions on the submodular function. One such assumption is the (total) *curvature*, defined as $\kappa_f = 1 - \min_{v \in V} \frac{f(v|V\setminus\{v\})}{f(v)}$ — the greedy algorithm has a $\frac{1}{\kappa_f}(1 - e^{-\kappa_f})$ and a $\frac{1}{\kappa_f + p}$ guarantee [15] for a cardinality and for p matroid constraints, respectively. Curvature is also attractive since it is linear time computable with only oracle function access. Liu et al. [86] shows that κ_f can be replaced by a similar quantity, i.e., $b = 1 - \min_{v \in A \in \mathcal{I}} \frac{f(\{v\}|A\setminus\{v\})}{f(\{v\})}$ for a single matroid $\mathcal{M} = (V, \mathcal{I})$, a quantity defined only on the independent sets of the matroid, thereby improving the bounds further. In the chapter, however, we utilize the traditional definition of curvature. The current best guarantee is $1 - \kappa_f/e$ for a cardinality constraint using modifications of the continuous greedy algorithm [117] and $\frac{1}{\epsilon + p}$ for multiple matroid constraints based on a local search algorithm [79]. In another relevant result, Sarpatwar et al. [107] gives a bound of $(1-e^{-(p+1)})/(p+1)$ for submodular maximization with a single knapsack and the intersection of p matroid constraints.

When $g(X)$ is not modular, the problem is much harder and is NP-hard to approximate to any factor (Lemma 1). In this chapter, we show that bounds are obtainable if we make analogous further assumptions on the supermodular function g. That is, we introduce a natural curvature notion to monotone non-decreasing nonnegative supermodular functions, defining the **supermodular curvature** as $\kappa^g = \kappa_{g(V)-g(V\setminus X)} = 1 - \min_{v \in V} \frac{g(v)}{g(v|V\setminus\{v\})}$. We note that κ^g is distinct from the steepness [52, 117] of a nonincreasing supermodular function (see Section 2.3.1). The function $g(V) - g(V \setminus X)$ is a normalized monotonic non-decreasing submodular function, known as the submodular function dual to the supermodular function g [39]. *Supermodular curvature* is a natural dual to *submodular curvature* and, like submodular curvature, is computationally feasible to compute, requiring only linear time in the oracle

	bound	hardness
cardinality constraint	$\frac{1}{\kappa_f}\left[1-e^{-(1-\kappa^g)\kappa_f}\right]$	$1-\kappa^g+\epsilon$
p matroid constraints	$\frac{1-\kappa^g}{(1-\kappa^g)\kappa_f+p}$	$(1-\kappa^g)O(\frac{\ln p}{p})$

Table 2.1: Lower bounds for GreedMax (Alg. 6)/SemiGrad (Alg. 3) and BP maximization hardness.

model, unlike other measures of non-submodularity (Section 2.1.2). Hence, given a BP decomposition of $h = f + g$, it is possible, as we show below, to derive practical and useful quality assurances based on the curvature of each component of the decomposition.

We examine two algorithms, GreedMax (Alg. 6) and SemiGrad (Alg. 3) and show that, despite the two algorithms being different, both of them have a worst-case guarantee of $\frac{1}{\kappa_f}\left[1-e^{-(1-\kappa^g)\kappa_f}\right]$ for a cardinality constraint (Theorem 1) and $\frac{1-\kappa^g}{(1-\kappa^g)\kappa_f+p}$ for p matroid constraints (Theorem 2). If $\kappa^g = 0$ (i.e., g is modular), the bounds reduce to $\frac{1}{\kappa_f}(1-e^{-\kappa_f})$ and $\frac{1}{\kappa_f+p}$, which recovers the aforementioned bounds. If $\kappa^g = 1$ (i.e., g is fully curved) the bounds are 0 since, in general, the problem is NP-hard to approximate (Lemma 1). For pure monotone supermodular function maximization, the bounds yield $1-\kappa^g$ and $(1-\kappa^g)/p$ respectively. We also show that no polynomial algorithm can do better than $1-\kappa^g+\epsilon$ or $(1-\kappa^g)O(\frac{\ln p}{p})$ for cardinality or multiple matroid constraints respectively unless P=NP. Therefore, no polynomial algorithm can beat GreedMax by a factor of $\frac{1+\epsilon}{1-e^{-1}}$ or $O(\ln(p))$ for the two constraints unless P=NP.

2.1.1 Applications

Problem 3 naturally applies to a number of machine learning and data science applications.

Summarization with Complementarity Submodular functions are an expressive set of models for summarization tasks where they capture how data elements are mutually redundant. In some cases, however, certain subsets might be usefully chosen together, i.e., when

their elements have a complementary relationship. For example, when choosing a subset of training data samples for a supervised machine learning system [132], nearby points on opposite sides of a decision boundary would be more useful to characterize this boundary if chosen together. Also, for the problem of document summarization [83, 81], where a subset of sentences is chosen to represent a document, there are some cases where a single sentence makes sense only in the context of other sentences, an instance of complementarity. In such cases, it is reasonable to allow these relationships to be expressed via a monotone supermodular function. One such complementarity family takes g to be a weighted sum of monotone convex functions composed with non-negative modular functions, as in $g(A) = \sum_i w_i \psi_i(m_i(A))$. A still more expressive family includes the "deep supermodular functions" [9] which consist of multiple nested layers of such transformations. A natural formulation of the summarization with complementary problem is to maximize an objective that is the weighted sum of a monotone submodular utility function and one of the above complementarity functions. Hence, such a formulation is an instance of Problem 3. In either case, the supermodular curvature is easy to compute, and for many instances is less than unity leading to a quality assurance based on the results of this chapter.

Generalized Bipartite Matching Submodularity has been used to generalize bipartite matching. For example, a generalized bipartite matching [84] procedure starts with a non-negative weighted bipartite graph (V, U, E), where V is a set of left vertices, U is a set of right vertices, $E \subseteq V \times U$ is a set of edges, and $h : 2^E \to \mathbb{R}_+$ is a score function on the edges. Note that a matching constraint is an intersection of two partition matroid constraints, so a matching can be generalized to the intersection of multiple matroid constraints. Word alignment between two sentences of different languages [89] can be viewed as a matching problem, where each word pair is associated with a score reflecting the desirability of aligning that pair, and an alignment is formed as the highest scored matching under some constraints. Lin and Bilmes [84] use a submodular objective functions that can represent complex interactions among alignment decisions. Also in [4], similar bipartite matching generalizations are

used for the task of peptide identification in tandem mass spectrometry. By utilizing a BP function in Problem 3, our approach can extend this to allow also for complementarity to be represented amongst sets of matched vertices. Further discussion of **Generalized Bipartite Matching** is followed in Part II

2.1.2 Approach, and Related Studies

An arbitrary set function can always be expressed as a difference of submodular (DS) functions [93, 54]. Although finding such a decomposition itself can be hard [54], the decomposition allows for additional optimization strategies based on discrete semi-gradients (Equation (2.1)) that do not offer guarantees, even in the unconstrained case [54]. Our problem is a special case of constrained DS optimization since a negative submodular function is supermodular. Our problem also asks for a BP decomposition of h which is not always possible even for monotone functions (Lemma 2). Constrainedly optimizing an arbitrary monotonic non-decreasing set function is impossible in polynomial time and not even approximable to any positive factor (Lemma 1). In general, there are two ways to approach such a problem: one is to offer polynomial time heuristics without any theoretical guarantee (and hence possibly performing arbitrarily poorly in worst case); another is to analyze (using possibly exponential time itself, e.g., see below starting with the submodularity ratio) the set function in order to provide theoretical guarantees. In our framework, as we will see, the BP decomposition not only allows for additional optimization strategies as does a DS decomposition, but also, given additional information about the curvature of the two components (computable easily in linear time), allows us to show how the set function can be approximately maximized in polynomial time with guarantees. With a curvature analysis, not only the greedy algorithm but also a semi-gradient optimization strategy (Alg. 3) attains a guarantee even in the constrained setting. We also argued, in Section 2.1.1, that BP functions, even considering their loss of expressivity relative to DS functions, are still quite natural in applications.

Submodularity Ratio and Curvature Bian et al. [8] offered a bound based on both the submodularity ratio and a newly introduced form of generalized curvature. The submodularity ratio [20] of a non-negative set function h is defined as the largest scalar γ s.t. $\sum_{\omega \in \Omega \setminus S} h(\omega|S) \geq \gamma h(\Omega|S), \forall \Omega, S \subseteq V$ and is equal to one if and only if h is submodular[8]. It is often defined as $\gamma_{U,k}(h) = \min_{L \subseteq U, S:|S| \leq k, S \cap L = \emptyset} \frac{\sum_{x \in S} h(x|L)}{h(S|L)}$ for $U \subseteq V$ and $1 \leq k \leq |V|$, and then $\gamma = \gamma_{V,|V|}(h)$. The generalized curvature [8] of a non-negative set function h is defined as the smallest scalar α s.t. $h(i|S \setminus \{i\} \cup \Omega) \geq (1-\alpha)h(i|S \setminus \{i\}), \forall \Omega, S \subseteq V, i \in S \setminus \Omega$. [8] offers a lower bound of $\frac{1}{\alpha}(1-e^{-\alpha\gamma})$ for the greedy algorithm. Computing this bound is not computationally feasible in general because both the submodularity ratio and the generalized curvature are information theoretically hard to compute under the oracle model, as we show in Section 2.I.2. This is unlike curvatures κ_f, κ^g which are both computable in linear time given only oracle access to both f and g. We make further comparisons between the pair κ_f, κ^g with the submodularity ratio in Section 2.I.

Approximately Submodular Functions A function h is said to be ϵ-approximately submodular if there exists a submodular function f such that $(1-\epsilon)f(S) \leq h(S) \leq (1+\epsilon)f(S)$ for all subsets S. Horel and Singer [49] show that the greedy algorithm achieves a $(1 - 1/e - O(\delta))$ approximation ratio when $\epsilon = \frac{\delta}{k}$. Furthermore, this bound is tight: given a $1/k^{1-\beta}$-approximately submodular function, the greedy algorithm no longer provides a constant factor approximation guarantee.

Elemental Curvature and Total Primal Curvature Wang et al. [128] analyze the approximation ratio of the greedy algorithm on maximizing non-submodular functions under cardinality constraints. Their bound is $1 - \left(1 - \left(\sum_{i=1}^{k-1} \alpha^i\right)^{-1}\right)^k$ based on the *elemental curvature* with $\alpha = \max_{S \subseteq X, i,j \in X} \frac{f(i|S \cup \{j\})}{f(i|S)}$, and α^i the i^{th} power of α. Smith and Thai [112] generalize this definition to *total primal curvature*, $\Gamma(x|B,A) = \frac{f(x|A \cup B)}{f(x|A)}$ and define an estimator $\hat{\Gamma}(i,S)$ satisfying $\forall |T| \leq k, S \subset T, i = |T \setminus S|, x \notin T \cup S : \Gamma(x|T,S) \leq \hat{\Gamma}(i,S) + \epsilon_i$. They claim a bound of $\left[1 + \left(\frac{f(S^+)}{f(S)} - 1\right) \sum_{t=0}^{k-1} (\hat{\Gamma}(t,S) + \epsilon_t)\right]^{-1} f(S^*) \leq f(S)$ where S is the

greedy solution, and S^+ is the greedy solution for an identical problem for $k+1$ cardinality constraints. They also claim that finding a deterministic strict estimator $\hat{\Gamma}$ is not feasible and therefore, they provide an algorithm for finding a probabilistic estimator based on Monte-Carlo simulation.

Supermodular Degree Feige and et al. [32] introduce a parameter, the supermodular degree, for solving the welfare maximization problem. Feldman and et al. [36, 35] use this concept to analyze monotone set function maximization under a p-extendable system constraint with guarantees. A supermodular degree of one element $u \in V$ by a set function h is defined as the cardinality of the set $\mathcal{D}_h^+(u) = \{v \in V | \exists_{S \subseteq V} h(u|S+v) > h(u|S)\}$, containing all elements whose existence in a set might increase the marginal contribution of u. The supermodular degree of h is $\mathcal{D}_h^+ = \max_{u \in V} |D_h^+(u)|$. A set system $(V, \mathcal{I})$ is called p-extendable [36, 35] if for every two subsets $T \subseteq S \in \mathcal{I}$ and element $u \notin T$ for which $T \cup u \in \mathcal{I}$, there exists a subset $Y \subseteq S \setminus T$ of cardinality at most p for which $S \setminus Y + u \in \mathcal{I}$, which is a generalization of the intersection of p matroids. They offer a greedy algorithm for maximizing a monotonic non-decreasing set function h subject to a p-extendable system with an guarantee of $\frac{1}{p(\mathcal{D}_h^+ + 1)+1}$ and time complexity polynomial in n and $2^{\mathcal{D}_h^+}$ [36, 35], where $n = |V|$. But again, $\mathcal{D}_h^+$ can not be calculated in polynomial time in general unlike our curvatures. Moreover, if we consider a simple supermodular function $g(X) = |X|^{1+\alpha}$ where α is a small positive number. Then $\mathcal{D}_h^+ = n - 1$ since all elements have supermodular interactions. Therefore, the time complexity of their algorithm is polynomial in 2^{n-1} and their bound is $\frac{1}{pn+1}$, while our algorithm requires at most n^2 quires with a performance guarantee of $\frac{1-\log(n)\kappa^g}{p}$ where $\kappa^g = 1 - \frac{1}{n^{1+\alpha} - (n-1)^{1+\alpha}}$. When α is small, our bound is around n times betters than theirs; e.g., $n = 10$, $p = 5$, $\alpha = 0.05$, ours is around $\frac{1}{7.61}$ while theirs is $\frac{1}{51}$.

Proportional Submodularity Borodin et al. [10] define the notion of proportionally submodular functions defined as those set functions h satisfying $|X|h(Y) + |Y|h(X) \geq |X \cap Y|h(X \cup Y) + |X \cup Y|h(X \cap Y)$ for all $X, Y \subseteq V$. The class of proportionally submod-

ular functions includes both submodular functions and also some supermodular functions, although there are instances of BP functions, e.g., $h(X) = |X|^4$, that are not proportionally submodular ([10] proposition 3.12).

Discussion The above results are both useful and complementary with our analyses below for BP-decomposable functions, thus broadening our understanding of settings where the greedy and semi-gradient algorithms offer a guarantee. We say our analysis is complementary in a sense the following example demonstrates. Should a given function h have a BP decomposition $h = f + g$, then it is easy, given oracle access to both f and g, to compute curvatures and establish bounds. On the other hand, if we do not know h's BP decomposition, or if h does not admit a BP decomposition (Lemma 2), then we would need to resort, for example, to the submodularity ratio and generalized curvature bounds of Bian et al. [8].

2.1.3 Recent studies on the sum of submodular and supermodular functions

There are recent studies following this work after it got published in 2018, studying the optimization of the sum of submodular and supermodular functions in different scenarios. Narang et al. [92] study **online suBmodular + suPermodular (BP) maximization** with bandit feedback, and propose algorithm achieves sublinear α-regret for $\alpha = \frac{1}{\kappa_f}\left(1 - e^{-\kappa_f(1-\kappa^g)}\right)$, where κ_f and κ^g use the same definition in this work. This choice of α reflects the approximation ratio of the greedy algorithm for BP functions in the offline setting. Ji et al. [63] propose a **stochastic greedy algorithm** that gets a lower performance guarantee $1 - e^{-(1-\epsilon)(1-\kappa^g)}$, but with a faster running time $O(n \log(1/\epsilon))$.

2.2 Approximation Algorithms for BP Maximization

Algorithm 2: GreedMax for BP maximization

Input: f, g and constraint set $\mathcal{C}$.
Output: An approximation solution $\hat{X}$.
Initialize: $X_0 \leftarrow \emptyset$, $i \leftarrow 0$ and $R \leftarrow V$
while $\exists v \in R$ s.t. $X_i \cup v \in \mathcal{C}$ **do**
  $v \in \text{argmax}_{v \in R, X_i \cup v \in \mathcal{C}} f(v|X_i) + g(v|X_i)$.
  $X_{i+1} \leftarrow X_i \cup v$.
  $R \leftarrow R \setminus v$.
  $i \leftarrow i + 1$.
end while
Return $\hat{X} \leftarrow X_i$.

Algorithm 3: SemiGrad for BP maximization

Input: f, g, constraint set $\mathcal{C}$ and an initial set X_0
Output: An approximation solution $\hat{X}$.
Initialize: $i \leftarrow 0$.
repeat
  pick a semigradient g_i at X_i of g
  $X_{i+1} \in \text{argmax}_{X \in \mathcal{C}} f(X) + g_i(X) \backslash\backslash \frac{1}{\kappa_{f+g_i}}(1 - e^{-\kappa_{f+g_i}})$−Approximately solved by Algorithm 6
  $i \leftarrow i + 1$
until we have converged ($X_i = X_{i-1}$)
Return $\hat{X} \leftarrow X_i$

GreedMax (Alg. 6) The simplest and most well known algorithm for approximate constrained non-monotone submodular maximization is the greedy algorithm [95]. We show that this also works boundedly well for BP maximization when the functions are not both fully curved ($\kappa_f \leq 1, \kappa^g < 1$). At each step, a feasible element with highest gain with respect

to the current set is chosen and added to the set. Finally, if no more elements are feasible, the algorithm returns the greedy set.

SemiGrad (Alg. 3) Akin to convex functions, supermodular functions have tight modular lower bounds. These bounds are related to the subdifferential $\partial_g(Y)$ of the supermodular set function g at a set $Y \subseteq V$, which is defined [39][2] as:

$$\partial_g(Y) = \{y \in \mathbb{R}^n : g(X) - y(X) \geq g(Y) - y(Y) \text{ for all } X \subseteq V\} \tag{2.1}$$

It is possible, moreover, to provide specific subgradients [58, 60], computable in linear time, that define the following two modular lower bounds:

$$m_{g,X,1}(Y) \triangleq g(X) - \sum_{j \in X \setminus Y} g(j|X \setminus j) + \sum_{j \in Y \setminus X} g(j|\emptyset), \tag{2.2}$$

$$m_{g,X,2}(Y) \triangleq g(X) - \sum_{j \in X \setminus Y} g(j|V \setminus j) + \sum_{j \in Y \setminus X} g(j|X). \tag{2.3}$$

Then $m_{g,X,1}(Y), m_{g,X,2}(Y) \leq g(Y), \forall Y \subseteq V$ and $m_{g,X,1}(X) = m_{g,X,2}(X) = g(X)$. Removing constants yields normalized non-negative (since g is monotone) modular functions for g_i in Alg. 3.

Having formally defined the modular lower bound of g, we are ready to discuss how to apply this machinery to BP maximization. SemiGrad consists of two stages. In the first stage, it is initialized by an arbitrary set (e.g., $\emptyset$, V, or the solution of GreedMax). In the second stage, SemiGrad replaces g by its modular lower bound, and solves the resulting problem using GreedyMax. The algorithm repeatedly updates the set and calculates an updated modular lower bound until convergence.

Since SemiGrad does no worse than the arbitrary initial set, we may start with the solution of GreedMax and show that SemiGrad is always no worse than GreedMax.

[2] [39] defines the subdifferential of a submodular set function. The definition of the subdifferential for a supermodular set function takes the same form (i.e., Equation 2.1) as the definition of the subdifferential for a submodular function. Specific instances of supermodular subgradients, yielding Equations (2.2)-(2.3) and as found in [58, 60], however, take a different form than instances of submodular subdifferentials.

Interestingly, we obtain the same bounds for SemiGrad even if we start with the empty set (Theorems 3 and 4) despite that they may behave quite differently empirically and yield different solutions (Section 3.6).

2.3 Analysis of Approximation Algorithms for BP Maximization

We next analyze the performance of two algorithms GreedMax (Alg. 6) and SemiGrad(Alg. 3) under a cardinality constraint and under p matroid constraints. First, we claim that BP maximization is hard and can not be approximately solved to any factor in polynomial time in general.

Lemma 1. *[123] There exists an instance of a BP maximization problem that can not be approximately solved to any positive factor in polynomial time.*

Proof. For completeness, Appendix 2.A offers a detailed proof based on [123]. □

It is also important to realize that not all monotone functions are BP-decomposable, as the following demonstrates.

Lemma 2. *There exists a monotonic non-decreasing set function h that is not BP decomposable.*

Proof. Let $h(X) = \min(\max(|X|, 1), 3) - 1$. This function is monotonic, and we wish to show it is not BP decomposable. Let $A \subset B$ be subsets of V with $|A| = 1$ and $|B| = 3$. Let $v \in V \setminus B$. We calculate that $h(v|\emptyset) = 0$, $h(v|A) = 1$, $h(v|B) = 0$. So $h(v|\emptyset) + h(v|B) < h(v|A)$.

Assume $h(X) = f(X) + g(X)$ where f is submodular, g is supermodular and both are monotonic non-decreasing. We have $f(v|\emptyset)+f(v|B) \geq f(v|\emptyset) \geq f(v|A)$ and $g(v|\emptyset)+g(v|B) \geq g(v|B) \geq g(v|A)$. Therefore $h(v|\emptyset)+h(v|B) \geq h(v|A)$ by summing the two inequalities, which is a contradiction. We thus have that h is not BP decomposable.

□

2.3.1 Supermodular Curvature

Although BP maximization is therefore not possible in general, we show next that we can get worst-case lower bounds using curvature whenever the functions in question indeed have limited curvature.

The (total) curvature of a submodular function f is defined as $\kappa_f = 1 - \min_{v \in V} \frac{f(v|V \setminus \{v\})}{f(v)}$ [15]. Note that $0 \leq \kappa_f \leq 1$ since $0 \leq f(v|V \setminus \{v\}) \leq f(v)$ and if $\kappa_f = 0$ then f is modular. We observed that for any monotonically non-decreasing supermodular function $g(X)$, the dual submodular function [39] $g(V) - g(V \setminus X)$ is always monotonically non-decreasing and submodular. Hence, the definition of submodular curvature can be naturally extended to supermodular functions g:

Definition 1. T*he* supermodular curvature *of a non-negative monotone nondecreasing supermodular function is defined as* $\kappa^g = \kappa_{g(V)-g(V\setminus X)} = 1 - \min_{v \in V} \frac{g(v)}{g(v|V\setminus\{v\})}$.

For clarity of notation, we use a superscript for supermodular curvature and a subscript for submodular curvature, which also indicates the duality between the two. In fact, for supermodular curvature, we can recover the submodular curvature.

Corollary 1. $\kappa_f = \kappa^{f(V)-f(V\setminus X)}$.

The dual form also implies similar properties, e.g., we have that $0 \leq \kappa^g \leq 1$ and if $\kappa^g = 0$ then g is modular. In both cases, a form of curvature indicates the degree of submodularity or supermodularity. If $\kappa_f = 1$ (or $\kappa^g = 1$), we say that f (or g) is fully curved. Intuitively, a submodular function is very (or fully) curved if there is a context B and element v at which the gain is close to (or equal to) zero ($f(v|B) \approx 0$), whereas a supermodular function is very (or fully) curved if there is an element v whose valuation is close to (or equal to) zero ($g(v) \approx 0$). We can calculate both submodular and supermodular curvature easily in linear time. Hence, given a BP decomposition of $h = f + g$, we can easily calculate both curvatures, and the corresponding bounds, with only oracle access to f and g.

Proposition 1. *Calculating* κ_f *or* κ^g *requires at most* $2|V| + 1$ *oracle queries of* f *or* g.

The steepness [52, 117] of a monotone nonincreasing supermodular function g' is defined as $s = 1 - \min_{v \in V} \frac{g'(v|V\setminus\{v\})}{g'(v|\emptyset)}$. Here, the numerator and denominator are both negative and g need not be normalized. Steepness has a similar mathematical form to the submodular curvature of a nondecreasing submodular function f, i.e., $\kappa_f = 1 - \min_{v \in V} \frac{f(v|V\setminus\{v\})}{f(v|\emptyset)}$, but is distinct from the supermodular curvature. Steepness may be used to offer a bound for the minimization of such nonincreasing supermodular functions [117], whereas we in the present work are interested in maximizing nondecreasing BP (and, hence, which also includes supermodular) functions.

2.3.2 Theoretical Guarantees for GreedMax

Before analyzing specific constraints, we first analyze each step of GreedMax based on submodular and supermodular curvature.

The following holds for any chain of sets, not just those produced by the greedy algorithm.

Lemma 3. *For any chain of solutions $\emptyset = S_0 \subset S_1 \subset \ldots \subset S_k$, where $|S_i| = i$, the following holds for all $i = 0 \ldots k-1$,*

$$h(X^*) \leq \kappa_f \sum_{j:s_j \in S_i \setminus X^*} a_j + \sum_{j:s_j \in S_i \cap X^*} a_j + h(X^* \setminus S_i | S_i) \tag{2.4}$$

where $\{s_i\} = S_i \setminus S_{i-1}$, $a_i = h(s_i|S_{i-1})$ and X^* is the optimal set.

Proof. See Appendix 2.B. □

Cardinality constraints

In this section, we provide a lower bound for Greedy maximization of a BP function under a cardinality constraint, inspired by the proof in [15] where they focus only on submodular functions.

Lemma 4. GreedMax *is guaranteed to obtain a solution* $\hat{X}$ *such that*

$$h(\hat{X}) \geq \frac{1}{\kappa_f}\left[1 - \left(1 - \frac{(1-\kappa^g)\kappa_f}{k}\right)^k\right] h(X^*) \tag{2.5}$$

where $X^* \in \operatorname{argmax}_{|X|\leq k} h(X)$, $h(X) = f(X) + g(X)$, κ_f *is the curvature of submodular* f *and* κ^g *is the curvature of supermodular* g.

Proof. See Appendix 2.C. □

Theorem 1. Theoretical guarantee in the cardinality constrained case. GreedMax *is guaranteed to obtain a solution* $\hat{X}$ *such that*

$$h(\hat{X}) \geq \frac{1}{\kappa_f}\left[1 - e^{-(1-\kappa^g)\kappa_f}\right] h(X^*) \tag{2.6}$$

where $X^* \in \operatorname{argmax}_{|X|\leq k} h(X)$, $h(X) = f(X) + g(X)$, κ_f *is the curvature of submodular* f *and* κ^g *is the curvature of supermodular* g.

Proof. This follows Lemma 4 and uses the inequality $(1 - \frac{a}{k})^k \leq e^{-a}$ for all $a \geq 0$ and $k \geq 1$. □

Theorem 1 gives a lower bound of GreedMax in terms of the submodular curvature κ_f and the supermodular curvature κ^g. We notice that this bound immediately generalizes known results and provides one new one.

1. $\kappa_f = 0$, $\kappa^g = 0$, $h(\hat{X}) = h(X^*)$. In this case, the BP problem reduces to modular maximization under a cardinality constraint, which is solved exactly by the greedy algorithm.

2. $\kappa_f > 0$, $\kappa^g = 0$, $h(\hat{X}) \geq \frac{1}{\kappa_f}\left[1 - e^{-\kappa_f}\right] h(X^*)$. In this case, BP problem reduces to submodular maximization under a cardinality constraint, and with the same $\frac{1}{\kappa_f}\left[1 - e^{-\kappa_f}\right]$ guarantee for the greedy algorithm [15].

3. If we take $\kappa_f \to 0$, we get $1 - \kappa^g$, which is a new curvature-based bound for monotone supermodular maximization subject to a cardinality constraint.

4. $\kappa^g = 1$, $h(\hat{X}) \geq 0$ which means, in the general fully curved case for g, this offers no theoretical guarantee for constrained BP or supermodular maximization, consistent with [123] and Lemma 1.

Weaker Bound in the Cardinality Constrained Case via Simple Analysis

The bound in Equation (2.6) is one of the major contributions of this chapter. Another bound can be easily achieved using a surrogate objective $h'(X) = f(X) + \sum_{v \in X} g(v)$, similar to an approach used in [56]. We have that $h'(X) \leq h(X)$ thanks to the supermodularity of g, and we can apply GreedMax directly to h', the solution of which has a guarantee w.r.t. the original objective h. The proof of this bound is quite a bit simpler, so we first offer it here immediately. On the other hand, we also show that the bound obtained by this method is worse than Equation (2.6) for all $0 < \kappa_f, \kappa^g < 1$, sometimes appreciably.

Lemma 5. Weak bound in cardinality constrained case. GreedMax *maximizing* $h'(X) = f(X) + \sum_{v \in X} g(v)$ *is guaranteed to obtain a solution* $\hat{X}$ *such that*

$$h(\hat{X}) \geq \frac{1 - \kappa^g}{\kappa_f} \left[1 - e^{-\kappa_f}\right] h(X^*) \tag{2.7}$$

where $X^* \in \operatorname{argmax}_{|X| \leq k} h(X)$, $h(X) = f(X) + g(X)$, κ_f *is the curvature of submodular* f *and* κ^g *is the curvature of supermodular* g.

Proof. According to Lemma 9 (iv), $(1 - \kappa^g)h(X) \leq h'(X)$ for all $X \subseteq V$. Also we have $h'(X) \leq h(X)$. And h' is a monotone submodular function with $\kappa_{h'} = 1 - \min_{v \in V} \frac{h'(v|V \setminus \{v\})}{h'(v)} = 1 - \min_{v \in V} \frac{f(v|V \setminus \{v\}) + g(v)}{f(v) + g(v)} \leq 1 - \min_{v \in V} \frac{f(v|V \setminus \{v\})}{f(v)} = \kappa_f$ since $0 \leq f(v|V \setminus \{v\}) \leq f(v)$.

Using the traditional curvature bound for submodular maximization [15], the greedy algorithm to maximize h' provides a solution $\hat{X}$ s.t. $h'(\hat{X}) \geq \frac{1}{\kappa_{h'}} \left[1 - e^{-\kappa_{h'}}\right] h'(X^*)$ where $X^* \in \operatorname{argmax}_{|X| \leq k} h(X)$. Thus, we have

$$h(\hat{X}) \geq h'(\hat{X}) \geq \frac{1}{\kappa_{h'}} \left[1 - e^{-\kappa_{h'}}\right] h'(X^*) \geq \frac{1}{\kappa_f} \left[1 - e^{-\kappa_f}\right] h'(X^*) \tag{2.8}$$

$$\geq \frac{1 - \kappa^g}{\kappa_f} \left[1 - e^{-\kappa_f}\right] h(X^*) \tag{2.9}$$

□

Next, we show that this bound is almost everywhere worse than Equation (2.6).

Lemma 6. $\frac{1}{\kappa_f}\left[1-e^{-(1-\kappa^g)\kappa_f}\right] \geq \frac{1-\kappa^g}{\kappa_f}\left[1-e^{-\kappa_f}\right]$ *for all* $0 \leq \kappa_f, \kappa^g \leq 1$ *where equality holds if and only if* $\kappa_f = 0$ *or* $\kappa^g = 0$ *or* $\kappa^g = 1$. *For simplicity, dividing by* 0 *is defined using limits, e.g.,* $\frac{1}{\kappa_f}\left[1-e^{-(1-\kappa^g)\kappa_f}\right] = \lim_{\kappa_f \to 0^+} \frac{1}{\kappa_f}\left[1-e^{-(1-\kappa^g)\kappa_f}\right] = 1-\kappa^g$ *when* $\kappa_f = 0$.

Proof. Let $\phi(\kappa_f, \kappa^g) = \frac{1}{\kappa_f}\left[1-e^{-(1-\kappa^g)\kappa_f}\right]$ and $\psi(\kappa_f, \kappa^g) = \frac{1-\kappa^g}{\kappa_f}\left[1-e^{-\kappa_f}\right]$. Specifically, $\phi(0, \kappa^g) = \lim_{\kappa_f \to 0^+} \phi(\kappa_f, \kappa^g) = 1-\kappa^g$ and $\psi(0, \kappa^g) = \lim_{\kappa_f \to 0^+} \psi(\kappa_f, \kappa^g) = 1-\kappa^g$. So if $\kappa_f = 0$, $\phi(\kappa_f, \kappa^g) = \psi(\kappa_f, \kappa^g)$.

When $0 < \kappa_f \leq 1$, we notice that $\phi(\kappa_f, \kappa^g) = \psi(\kappa_f, \kappa^g)$ when $\kappa^g = 0$ or $\kappa^g = 1$. When $0 < \kappa^g < 1$, we have $\phi(\kappa_f, \kappa^g) > \psi(\kappa_f, \kappa^g)$ since $\phi(\kappa_f, \kappa^g)$ is a strictly concave function in κ^g and $\psi(\kappa_f, \kappa^g)$ is linear in κ^g. □

A simple computation shows the maximum ratio of these two bounds is $1/(1-e^{-1}) \approx 1.5820$ when $\kappa_f = 1$ and $\kappa^g \to 1$. As another example, with $\kappa_f = 1$ and $\kappa^g = \ln(e-1) \approx 0.541$, the ratio is ≈ 1.2688. It should be noted that the modular approximation of g leading to $h'(X) = f(X) + \sum_{v \in X} g(v)$ is identical to one iteration of the semigradient approach (Algorithm 3) using a subgradient of g at $\emptyset$. A more complex analysis leads to the improved bound, as shown in Theorem 3.

Multiple matroid constraints

Matroids are useful combinatorial objects for expressing constraints in discrete problems, and which are made more useful when taking the intersection of the independent sets of $p > 1$ matroids defined on the same ground set [95]. In this section, we show that the greedy algorithm on a BP function subject to p matroid independent constraints has a guarantee if g is not fully curved.

Theorem 2. Theoretical guarantee in the p matroids case. GreedMax *is guaranteed to obtain a solution* $\hat{X}$ *such that*

$$h(\hat{X}) \geq \frac{1-\kappa^g}{(1-\kappa^g)\kappa_f + p} h(X^*) \tag{2.10}$$

where $X^* \in \text{argmax}_{X \in \mathcal{M}_1 \cap \ldots \cap \mathcal{M}_p} h(X)$, $h(X) = f(X) + g(X)$, κ_f *is the curvature of submodular* f *and* κ^g *is the curvature of supermodular* g.

Proof. See Appendix 2.D. □

Theorem 2 gives a theoretical lower bound of GreedMax in terms of submodular curvature κ_f and supermodular curvature κ^g for the p matroid constraints case. Like in the cardinality case, this bound also generalizes known results and yields a new one.

1. $\kappa_f = 0$, $\kappa^g = 0$, $h(\hat{X}) \geq \frac{1}{p} h(X^*)$. In this case, the BP problem reduces to modular maximization under p matroid constraints [15].

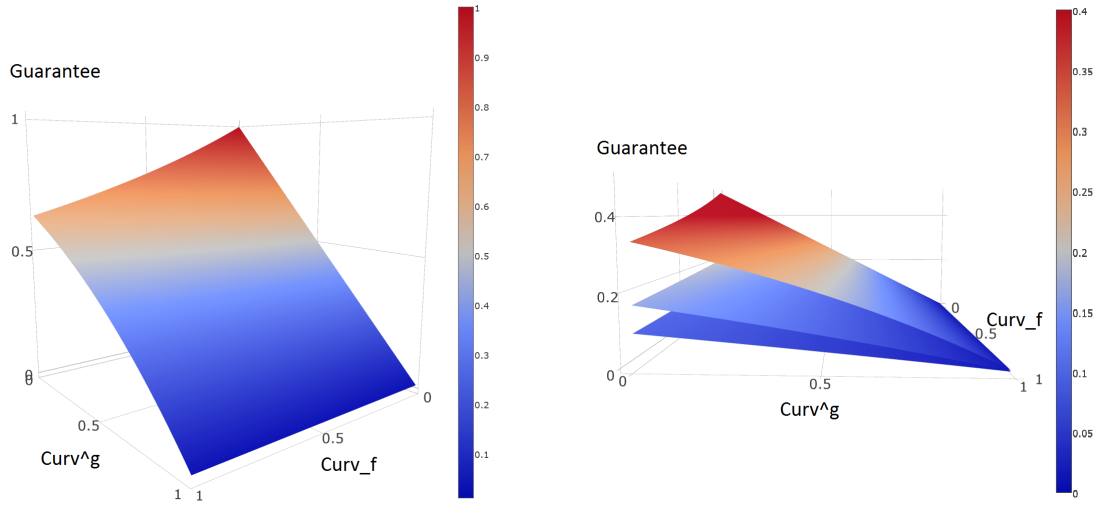
2. $\kappa_f > 0$, $\kappa^g = 0$, $h(\hat{X}) \geq \frac{1}{p+\kappa_f} h(X^*)$. In this case, the BP problem reduces to submodular maximization under p matroid constraints [15].

3. If we take $\kappa_f \to 0$, we get $(1-\kappa^g)/p$, which is a new curvature-based bound for monotone supermodular maximization subject to a p matroid constraints.

4. $\kappa^g = 1$, $h(\hat{X}) \geq 0$ which means that, in general, there is no theoretical guarantee for constrained BP or supermodular maximization.

2.3.3 Theoretical guarantee of SemiGrad

In this section, we show a perhaps interesting result that SemiGrad achieves the same bounds as GreedMax even if we initialize SemiGrad with $\emptyset$ and even though the two algorithms can produce quite different solutions (as demonstrated in Section 3.6).



(a) Cardinality constraint

(b) Multiple matroid constraints

Figure 2.1: Guarantees of GreedMax for two constraint types. The x and y axes are κ_f and κ^g, respectively, and the z axis is the guarantee. In (b), from top to bottom, the surfaces represent $p = 2, 5, 10$.

Theorem 3. SemiGrad *initialized with the empty set is guaranteed to obtain a solution* $\hat{X}$ *for the cardinality constrained case such that*

$$h(\hat{X}) \geq \frac{1}{\kappa_f}\left[1 - e^{-(1-\kappa^g)\kappa_f}\right] h(X^*) \tag{2.11}$$

where $X^* \in \operatorname{argmax}_{|X|\leq k} h(X)$, $h(X) = f(X) + g(X)$, *&* κ_f *(resp.* κ^g*) is the curvature of* f *(resp.* g*).*

Proof. See Appendix 2.E. □

Theorem 4. SemiGrad *initialized with the empty set is guaranteed to obtain a solution* $\hat{X}$*, feasible for the* p *matroid constraints, such that*

$$h(\hat{X}) \geq \frac{1 - \kappa^g}{(1-\kappa^g)\kappa_f + p} h(X^*) \tag{2.12}$$

where $X^* \in \operatorname{argmax}_{X\in\mathcal{M}_1\cap\ldots\cap\mathcal{M}_p} h(X)$, $h = f + g$, *&* κ_f *(resp.* κ^g*) is the curvature of* f *(resp.* g*).*

Proof. See Appendix 2.F. □

All the above guarantees are plotted in Figure 2.1 (in the matroid case for $p = 2$, 5, or 10 matroids).

2.4 Better Bound Than Theorem 1 by Splitting the Modular Part

We showed GreedMax can maximize BP functions with approximate guarantee $\frac{1}{\kappa_f}\left[1 - e^{-(1-\kappa^g)\kappa_f}\right]$. However, a BP split is not unique, and the bound can be improved if we use a 'smarter' split. In this section, $h(X) = f(X) + g(X)$ and we assume $f(X)$ is not a modular function, i.e. $f(v) > f(V \setminus \{v\})$ for at least one $v \in V$; otherwise the best split is trivial, $h(X) = 0 + (f(X) + g(X))$, and BP bound is $1 - \kappa^{f+g}$.

Let $\bar{f}(X) = f(X) - \sum_{v\in X} f(v|V \setminus \{v\})$ and $\bar{g}(X) = g(X) - \sum_{v\in X} g(v)$. It is straightforward to show that $\bar{f}$ and $\bar{g}$ are fully-curved monotonic non-decreasing submodular and supermodular functions, respectively. Let $m(X) = h(X) - \bar{f}(X) - \bar{g}(X) = \sum_{v\in X} [f(v|V \setminus \{v\}) + g(v)]$.

It is possible to split m into $m_{f,\lambda}$ and $m_{g,\lambda}$ using a set of parameters λ, where $\lambda \in \Lambda = [0,1]^n$, i.e. $0 \leq \lambda_v \leq 1$ for all $v \in V$. We have $m_{f,\lambda}(X) = \sum_{v \in X} \lambda_v m(v)$ and $m_{g,\lambda}(X) = \sum_{v \in X} \bar{\lambda}_v m(v)$, where $\bar{\lambda}_v = 1 - \lambda_v$.

So for each $\lambda \in \Lambda$, we can get a different BP split of $h(X) = f_\lambda(X) + g_\lambda(X)$, where $f_\lambda(X) = \bar{f}(X) + m_{f,\lambda}(X)$ and $g_\lambda(X) = \bar{g}(X) + m_{g,\lambda}(X)$. Then the problem is to find the best $\lambda \in \Lambda$ that can maximize $\frac{1}{\kappa_{f_\lambda}} \left[1 - e^{-(1-\kappa^{g_\lambda})\kappa_{f_\lambda}}\right]$.

Problem 4. *Find* $\max_{\lambda \in \Lambda} \gamma_\lambda = \frac{1}{\kappa_{f_\lambda}} \left[1 - e^{-(1-\kappa^{g_\lambda})\kappa_{f_\lambda}}\right]$, *given the values of* $f(v), f(v|V \setminus \{v\}), g(v), g(v|V \setminus \{v\})$ *for all* $v \in V$.

Note that we only need $4n$ function values to calculate the best bounds since κ_{f_λ} and κ^{g_λ} only depend on these function values and λ. In Problem 4, we assume that the $4n$ function values are fixed, so κ_{f_λ} and κ^{g_λ} are functions of λ; i.e. $\kappa_f(\lambda) : \mathbf{R}^n \to \mathbf{R} = \kappa_{f_\lambda}$, $\kappa^g(\lambda) : \mathbf{R}^n \to \mathbf{R} = \kappa^{g_\lambda}$. We write the curvature as $\kappa_f(\lambda)$ just to emphasize the fact it is a function of λ, but notation-wise, $\kappa_f(\lambda) = \kappa_{f_\lambda}$ is always true, and similar for g.

Let $\kappa_{f,v}(\lambda_v) : \mathbf{R} \to \mathbf{R} = 1 - \frac{f_\lambda(v|V\setminus\{v\})}{f_\lambda(v)} = \frac{\bar{f}(v)}{\bar{f}(v)+\lambda_v m(v)}$, since $\bar{f}(v|V \setminus \{v\}) = 0$. So, by definition, $\kappa_f(\lambda) = \max_{v \in V} \kappa_{f,v}(\lambda_v)$. Similarly, Let $\kappa^g_v(\lambda_v) : \mathbf{R} \to \mathbf{R} = 1 - \frac{g_\lambda(v)}{g_\lambda(v|V\setminus\{v\})} = \frac{\bar{g}(v|V\setminus\{v\})}{\bar{g}(v|V\setminus\{v\})+\lambda_v m(v)}$, since $\bar{g}(v) = 0$. And, by definition, $\kappa^g(\lambda) = \max_{v \in V} \kappa^g_v(\lambda_v)$.

Lemma 7. *In Problem 4, for any* $\lambda^* \in \operatorname{argmax}_{\lambda \in \Lambda} \gamma_\lambda = \frac{1}{\kappa_{f_\lambda}} \left[1 - e^{-(1-\kappa^{g_\lambda})\kappa_{f_\lambda}}\right]$, *we can construct a new* $\hat{\lambda}$ *from* λ^*, *such that* $\hat{\lambda} \in \operatorname{argmax}_{\lambda \in \Lambda} \gamma_\lambda$, *and* $\kappa_f(\hat{\lambda}) = \kappa_{f,v}(\hat{\lambda}_v)$ *for all* $v \in V$.

Proof. Let $K = \kappa_f(\lambda^*) = \max_{v \in V} \kappa_{f,v}(\lambda^*_v)$. Note that $0 < K \leq 1$ because we assume f is not purely modular,

we construct $\hat{\lambda}$ by

$$\hat{\lambda}_v = \frac{\bar{f}(v)}{m(v)} \left[\frac{1}{K} - 1\right] \tag{2.13}$$

Then we can show

1. $\hat{\lambda}_v \leq \lambda^*_v$ for all $v \in V$. This is because $\lambda^*_v = \frac{\bar{f}(v)}{m(v)} \left[\frac{1}{\kappa_{f,v}(\lambda^*_v)} - 1\right]$, by reversing the definition of $\kappa_{f,v}(\lambda^*_v) = \frac{\bar{f}(v)}{\bar{f}(v)+\lambda^*_v m(v)}$, and $K \geq \kappa_{f,v}(\lambda^*_v)$. This also implies $\hat{\lambda}_v \leq 1$.

2. $\hat{\lambda}_v \geq 0$ for all $v \in V$. This is because $K \leq 1$. Therefore, we know $\hat{\lambda} \in \Lambda$.

3. $\kappa_{f,v}(\hat{\lambda}_v) = K$ for all $v \in V$. Therefore, $\kappa_f(\hat{\lambda}) = K$.

4. $\kappa_v^g(\hat{\lambda}_v) \leq \kappa_v^g(\lambda_v^*)$, since $\kappa_v^g(\cdot)$ is an non-decreasing function and $\hat{\lambda}_v \leq \lambda_v^*$. This implies $\kappa^g(\hat{\lambda}) \leq \kappa^g(\lambda^*)$.

γ_λ depends on the submodular curvature $\kappa_f(\lambda)$ and supermodular curvature $\kappa^g(\lambda)$, and is monotonic non-increasing of these two curvatures. So $\gamma_{\hat{\lambda}} \geq \gamma_{\lambda^*}$ and we already know $\lambda^* \in \text{argmax}_{\lambda\in\Lambda}\, \gamma_\lambda$, then we must have $\hat{\lambda} \in \text{argmax}_{\lambda\in\Lambda}\, \gamma_\lambda$

3. is the second property we need for $\hat{\lambda}$ in the statement of this lemma. □

Corollary 2. $\max_{\lambda\in\Lambda} \gamma_\lambda = \max_{\lambda\in\Lambda'} \gamma_\lambda$, *where* $\Lambda' = \{\lambda \in \Lambda | \kappa_{f,v}(\lambda_v) = \kappa_f(\lambda), \forall v \in V\}$.

Therefore, in Problem 4, we only need to consider the solution set Λ'; in this set, $\kappa_{f,v}(\lambda_v)$ is the same for all $v \in V$.

Let $x = \frac{1}{\kappa_f(\lambda)}$, and we have $\kappa_{f,v}(\lambda_v) = \frac{1}{x}$.[3]

$$\frac{\bar{f}(v)}{\bar{f}(v) + \lambda_v m(v)} = \frac{1}{x} \tag{2.14}$$

$$1 + \frac{\lambda_v m(v)}{\bar{f}(v)} = x \tag{2.15}$$

$$\lambda_v = \frac{\bar{f}(v)}{m(v)}(x-1) \tag{2.16}$$

Therefore, we can treat λ_v as a function of x, since $\bar{f}(v)$ and $m(v)$ are fixed. The range of x bounded by $0 \leq \lambda_v \leq 1$. So $1 \leq x \leq \min_{v\in V} \frac{m(v)}{\bar{f}(v)} + 1$.

$\kappa_v^g(\lambda_v)$ is a function of λ_v and can also be treated as a function of x, i,e. $\kappa_v^g(x)$. So $\kappa^g(\lambda) = \max_{v\in V} \kappa_v^g(x)$.

$$\gamma_\lambda = \frac{1}{\kappa_{f_\lambda}} \left[1 - e^{-(1-\kappa^{g_\lambda})\kappa_{f_\lambda}}\right] \tag{2.17}$$

---

[3]The intuition of using $x = \frac{1}{\kappa_f(\lambda)}$ is $\frac{\bar{f}(v)}{\kappa_{f,v}(\lambda_v)} + \frac{\bar{g}(v|V\setminus\{v\})}{\kappa_v^g(\lambda_v)} = \bar{f}(v) + \bar{g}(v|V \setminus \{v\}) + m(v)$. And we have $\kappa_f(\lambda)$ always greater than 0.

$$= \frac{1}{\frac{1}{x}} \left[1 - e^{-(1-\max_{v\in V} \kappa_v^g(x))\frac{1}{x}}\right] \tag{2.18}$$

$$= \min_{v\in V} x \left[1 - e^{\frac{\kappa_v^g(x)-1}{x}}\right] \tag{2.19}$$

$$\tag{2.20}$$

where $\kappa_v^g(x) = \kappa_v^g(\frac{\bar{f}(v)}{m(v)}(x-1)) = \frac{\bar{g}(v|V\setminus\{v\})}{\bar{g}(v|V\setminus\{v\})+\left[1-\left(\frac{\bar{f}(v)}{m(v)}(x-1)\right)\right]m(v)}$.

Therefore, we can reconstruct Problem 4 as a one variable optimization problem on x, given the fixed $4n$ function values $f(v), f(v|V\setminus\{v\}), g(v), g(v|V\setminus\{v\})$ for all $v \in V$. Furthermore, we can show this function is actually concave.

Lemma 8. *$\gamma(x) = \min_{v\in V} x \left[1 - e^{\frac{\kappa_v^g(x)-1}{x}}\right]$ is a concave function of x.*

Proof. Let $a = \frac{\bar{f}(v)}{\bar{f}(v)+\bar{g}(v|V\setminus\{v\}+m(v)}$, $b = \frac{\bar{g}(v|V\setminus\{v\}}{\bar{f}(v)+\bar{g}(v|V\setminus\{v\})+m(v)}$, we have

$$\gamma_v(x) = x\left[1 - e^{\frac{\frac{b}{1-ax}-1}{x}}\right] \tag{2.21}$$

$$\frac{d^2}{dx^2}\gamma_v(x) = -e^{\frac{1-ax-b}{ax^2-x}} \cdot \frac{(2abx - b + (ax-1)^2)^2 - 2b(ax-1)a^2x^3}{x^3(ax-1)^4} \tag{2.22}$$

$$\leq 0 \tag{2.23}$$

since $ax \leq a\left(\frac{m(v)}{\bar{f}(v)} + 1\right) = \frac{m(v)+\bar{f}(v)}{m(v)+\bar{f}(v)+\bar{g}(v|V\setminus\{v\})} \leq 1$.

Then $\gamma(x)$ is the minimum of multiple concave functions, thus concave. □

Corollary 3. *Problem 4 can be solved by $\max_{\lambda\in\Lambda}\gamma_\lambda = \max_{1\leq x\leq \min_{v\in V}\frac{m(v)}{\bar{f}(v)}+1}\gamma(x)$, where $\gamma(x) = \min_{v\in V} x\left[1 - e^{\frac{\bar{g}(v|V\setminus\{v\})}{(\bar{f}(v)+\bar{g}(v|V\setminus\{v\}+m(v)-\bar{f}(v)x)x}-\frac{1}{x}}\right]$ is a concave function of x. Optimal solution $\lambda_v^* = \frac{\bar{f}(v)}{m(v)}(x^*-1)$.*

Some insights

1. Recall $\bar{g}(v|V\setminus\{v\}) = g(v|V\setminus\{v\}) - g(v)$, $\bar{f}(v) = f(v) - f(v|V\setminus\{v\})$ and $m(v) = f(v|V\setminus\{v\}) + g(v)$. $\gamma(x)$ contains only the initial and final gain of f and g, and is one variable concave function, thus easy to maximize.

2. The worst split gives us BP bound 0. This can be achieved by $\lambda_v = 1$ for all $v \in V$, and the supermodular part is fully curved. This process can improve the bound from 0 to a positive number.

3. If we break the assumption that f is not purely modular, then $\gamma(x)$ is a monotonic non-decreasing function, then the max bound is $\lim_{x\to+\infty} \min_{v\in V} x\left[1 - e^{\frac{\bar{g}(v|V\setminus\{v\})}{\bar{g}(v|V\setminus\{v\})+m(v)}^{-1}}\right] = \min_{v\in V} \frac{m(v)}{\bar{g}(v|V\setminus\{v\})+m(v)} = 1 - \kappa^h$, aligned with what we discussed at the beginning of this section.

Compared with trivial splits

1. The worst split, $\lambda_v = 1$ for all $v \in V$. In this case, BP bound is 0.

2. $\lambda_v = 0$ for all $v \in V$. In this case, f_λ is fully curved, and BP bound is $\min_{v\in V} 1 - e^{-\frac{m(v)}{\bar{g}(v|V\setminus\{v\})+m(v)}} = \min_{v\in V} 1 - e^{-\frac{f(v|V\setminus\{v\})+g(v)}{f(v|V\setminus\{v\})+g(v|V\setminus\{v\})}}$. This bound is not 0 if $f(v|V\setminus\{v\}) + g(v) > 0$ for all $v \in V$. However, this trivial split may not give the best bound. For example, if f is almost modular and g is modular, i.e. $\kappa_f = \epsilon$, $\kappa_g = 0$. The best BP bound is greater or equal to $\frac{1}{\epsilon}(1 - e^{-\epsilon}) \approx 1 - \epsilon$, where this trivial split gives us $1 - e^{-1}$.

3. $\lambda_v = \frac{1}{2}$ for all $v \in V$. We also use one example to show this split is not optimal. Let $f(X) = |X|$ and $g(X) = \max(0, |X| - 1)$. $lamba_v = \frac{1}{2}$ gives us BP bound $\frac{1}{3}$ while the best split gives us BP bound $\frac{1}{2}$.

4. Random $\lambda \in \Lambda$. Usually, these λ will give us a bound close to 0, since the $\kappa^{g_\lambda} = \max_{v\in V} \kappa_v^g(\lambda_v)$. It is likely to get one λ_v that is close to 0, thus the $\kappa_v^g(\lambda_v)$ is close to 1, and dominates the other terms. For example, if $f(X) = |X|$ is purely modular, and $g = \max(0, |X| - 1)$ is fully curved, and $f(v) = g(v|V\setminus\{v\})$. $P(\min_v \lambda_v \le \epsilon) = 1 - (1-\epsilon)^n$ since all λ_v are i.i.d., so $P(\max_v \kappa_v^g(\lambda_v) \ge \frac{1}{1+\epsilon}) = 1 - (1-\epsilon)^n$. Therefore, $E_{\lambda\in\Lambda}\gamma_\lambda \le [1 - (1-\epsilon)^n] \cdot \left[1 - \frac{1}{1+\epsilon}\right] + (1-\epsilon)^n \cdot 1 = \frac{\epsilon}{1+\epsilon} + \frac{(1-\epsilon)^n}{1+\epsilon}$ can be arbitrarily close to 0 when $n \to +\infty$, while the best split give us BP bound $\frac{1}{2}$.

2.5 Hardness

We next show that the curvature κ^g limits the polynomial time approximability of BP maximization.

Theorem 5. Hardness for cardinality constrained case. *For all $0 \leq \beta \leq 1$, there exists an instance of a BP function $h = f + g$ with supermodular curvature $\kappa^g = \beta$ such that no poly-time algorithm solving Problem 1 with a cardinality constraint can achieve an approximation factor better than $1 - \kappa^g + \epsilon$, for any $\epsilon > 0$.*

Proof. See Appendix 2.G. □

For the p matroid constraints case, Hazan et al. [47] studied the complexity of approximating p-set packing which is defined as follows: given a family of sets over a certain domain, find the maximum number of disjoint sets, which is actually a special case of finding the maximum intersection of p matroids. They claim that this problem cannot be efficiently approximated to a factor better than $O(\ln p/p)$ unless P = NP. We generalize their result to BP maximization.

Theorem 6. Hardness for p matroids constraint case. *For all $0 \leq \beta \leq 1$, there exists an instance of a BP function $h = f + g$ with supermodular curvature $\kappa^g = \beta$ such that no poly-time algorithm can achieve an approximation factor better than $(1 - \kappa^g)O(\frac{\ln p}{p})$ unless P=NP.*

Proof. See Appendix 2.H. □

Corollary 4. *No polynomial algorithm can beat* GreedMax *or* SemiGrad *by a factor of $\frac{1+\epsilon}{1-e^{-1}}$ for cardinality, or $O(\ln(p))$ for p matroid constraints, unless P=NP.*

2.6 Computational Experiments

We empirically test our guarantees for BP maximization subject to a cardinality constraint on contrived functions using GreedMax and SemiGrad. For the first experiment, we let $|V| = 20$ set the cardinality constraint to $k = 10$, and partition the ground set into

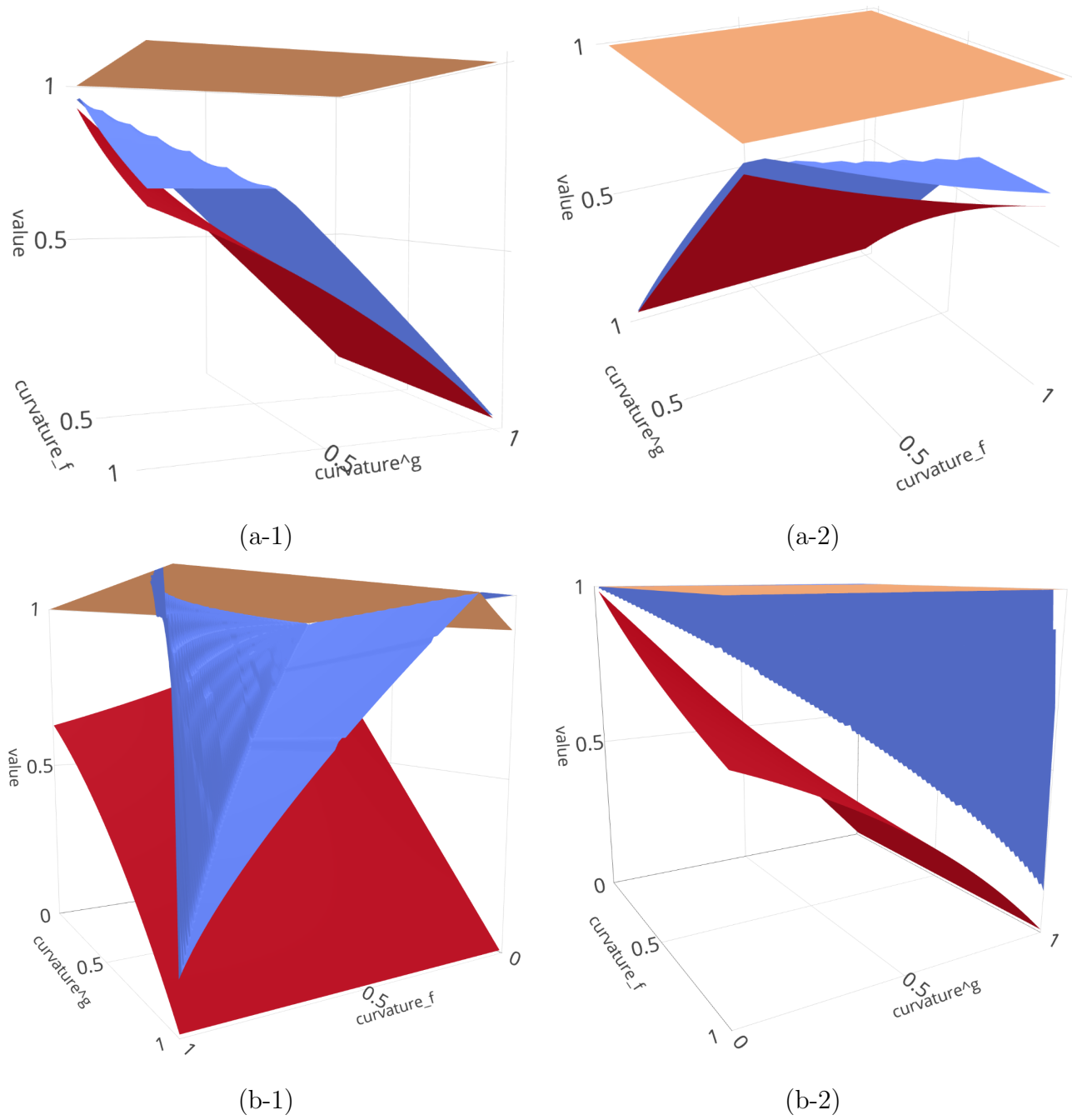


Figure 2.2: Empirical test of our guarantee. The upper and middle surface indicate the performance of SemiGrad and GreedMax respectively, and the lower surface is the theoretical worst case guarantee. (a) and (b) are two sets of experiments. (a-1) and (a-2) are two rotations of the same 3d plot; (b-1) and (b-2) are two rotations of the same 3d plot. 3d and interactive version of plot (a) and (b) are available in `https://plotly.com/~Wenruo/14/` and `https://plotly.com/~Wenruo/12/`.

$|V_1| = |V_2| = k$, $V_1 \cup V_2 = V$ where $V_1 = \{v_1, v_2, \ldots, v_k\}$. Let $w_i = \frac{1}{\alpha}\left[\left(1 - \frac{\alpha}{k}\right)^i - \left(1 - \frac{\alpha}{k}\right)^{i+1}\right]$ for $i = 1, 2, \ldots, k$. Then we define the submodular and supermodular functions as follows, $f(X) = \left[\frac{k - \alpha|X \cap V_2|}{k}\right] \sum_{\{i: v_i \in X\}} w_i + \frac{|X \cap V_2|}{k}$, $g(X) = |X| - \beta \min(1 + |X \cap V_1|, |X|, k) + \epsilon \max(|X|, |X| + \frac{\beta}{1-\beta}(|X \cap V_2| - k + 1))$ and $h(X) = \lambda f(X) + (1 - \lambda) g(X)$ for $0 \leq \alpha, \beta, \lambda \leq 1$ and $\epsilon = 1 \times 10^{-5}$. Immediately, we notice that $\kappa_f = \alpha$ and $\kappa^g = \beta$. In particular, we choose $\alpha, \beta, \lambda = 0, 0.01, 0.02, \ldots, 1$ and for all cases, we normalize $h(X)$ using either exhaustive search so that $\text{OPT} = h(X^*) = 1$. Since we are doing a proof-of-concept experiment to verify the guarantee, we are interested in the worst case performance at curvatures κ_f and κ^g. In Figure 2.2(a), we see that both methods are always above the theoretical worst case guarantee, as expected. Interestingly, SemiGrad is doing significantly better than GreedMax demonstrating the different behavior of the algorithms, despite their identical guarantee. Moreover, the gap between GreedMax and the bound layer is small (the maximum difference is 0.1852), which suggests the guarantee for greedy may be almost tight in this case.

The above example is designed to show the tightness of GreedMax and the better potential performance of SemiGrad. For a next experiment, we again let $|V| = 20$ and $k = 10$, partition the ground set into $|V_1| = |V_2| = k$, $V_1 \cup V_2 = V$. Let $f(X) = |X \cap V_1|^\alpha$ and $g(X) = \max(0, \frac{|X \cap V_2| - \beta}{1 - \beta})$ $0 \leq \alpha, \beta \leq 1$, and normalize h (by exhaustive search) to ensure $\text{OPT} = h(X^*) = 1$. Immediately, we notice that the curvature of f is $\kappa_f = 1 - k^\alpha + (k-1)^\alpha$ and the curvature of g is $\kappa^g = \beta$. The objective BP function is $h(X) = f(X) + g(X)$. We see that SemiGrad is again doing better than GreedMax in most but not all cases (Figure 2.2(b)) and both are above their bounds, as they should be.

Appendices

2.A Proof of Lemma 1

Lemma 1. *[123] There exists an instance of a BP maximization problem that can not be approximately solved to any positive factor in polynomial time.*

Proof. We consider the BP problem with ground set n and a cardinality constraint $|X| \leq$

$k = n/2$. Let $R \subseteq V$ be an arbitrary set with $|R| = k$. Let $f = 0$ and $g'(X) = \max(|X|-k, 0)$ so that $g'(X) = 0$ for all $|X| = k$. $g'(X)$ is clearly supermodular.

Let $g(X) = g'(X)$ for all $X \neq R$ but $g(R) = 0.5$. We notice that for $X \subset V$ and $v \notin X$, $g(v|X) = 0$ if $|X| \leq k-2$, $g(v|X) = 0$ or 0.5 if $|X| = k-1$, $g(v|X) = 0.5$ or 1 if $|X| = k$, and $g(v|X) = 1$ if $|X| \geq k+1$. Immediately, we have for all $X \subset Y \subset V$ and $v \notin Y$, $g(v|X) \leq g(v|Y)$. Therefore, $g(X)$ is also supermodular.

Next, we use a proof technique similar to [118]. Note that $g'(X) = g(X)$ if and only if $X \neq R$. So for any algorithm maximizing $g(X)$, before it evaluates $g(R)$, all function evaluations are the same with maximizing $g'(X)$. Additionally, since $g'(X) = \max(|X|-k, 0)$, it is permutation symmetric. Therefore, the algorithm can only do random search to find R. If the algorithm acquires a polynomial number $O(n^m)$ of sets of size k, the probability of finding R is $\frac{O(n^m)}{\binom{n}{k}} \leq \frac{O(n^m)}{(n/k)^k} = \frac{O(n^m)}{2^{n/2}} \leq O(2^{-n/2+\epsilon n})$ for all $\epsilon > 0$. Therefore, no polynomial time algorithm can distinguish g and g' with probability greater than $1 - O(2^{-n/2+\epsilon n})$ and will return 0 in almost all cases.

Hence, we have $\max_{|X|\leq k} f(X)+g(X) = 0.5 > 0$ so no polynomial algorithm can do better than $\max_{|X|\leq k} f(X) + g'(X) = 0$ with high probability, or has any positive guarantee. □

2.B Proof of Lemma 3

We begin with the following four-part lemma,

Lemma 9. *For a BP function $h(X) = f(X) + g(X)$, we have*

(i) $h(v|Y) \geq (1-\kappa_f)h(v|X)$ *for all* $X \subseteq Y \subset V$ *and* $v \notin Y$

(ii) $h(v|Y) \leq \frac{1}{1-\kappa^g}h(v|X)$ *for all* $X \subseteq Y \subset V$ *and* $v \notin Y$

(iii) $h(X|Y) \geq (1-\kappa_f)\sum_{v\in X\setminus Y} h(v|Y)$ *for all* $X, Y \subseteq V$

(iv) $h(X|Y) \leq \frac{1}{1-\kappa^g}\sum_{v\in X\setminus Y} h(v|Y)$ *for all* $X, Y \subseteq V$

Proof. (i) $\kappa_f = 1 - \min_{v \in V} \frac{f(v|V \setminus \{v\})}{f(v)}$, therefore, $f(v|V \setminus \{v\}) \geq (1 - \kappa_f) f(v)$ for all v.

So we have $f(v|Y) \geq f(v|V \setminus \{v\}) \geq (1 - \kappa_f) f(v) \geq (1 - \kappa_f) f(v|X)$ and $g(v|Y) \geq g(v|X) \geq (1 - \kappa_f) g(v|X)$ for all $X \subseteq Y \subset V$ and $v \notin Y$. Therefore, $h(v|Y) \geq (1 - \kappa_f) h(v|X)$ for all $X \subset Y \subseteq V$ and $v \notin Y$.

(ii) $\kappa^g = 1 - \min_{v \in V} \frac{g(v)}{g(v|V \setminus \{v\})}$, therefore, $g(v|V \setminus \{v\}) \leq \frac{1}{1-\kappa^g} g(v)$ for all v.

So we have $g(v|Y) \leq g(v|V \setminus \{v\}) \leq \frac{1}{1-\kappa^g} g(v) \leq \frac{1}{1-\kappa^g} g(v|X)$ and $f(v|Y) \leq f(v|X) \leq \frac{1}{1-\kappa^g} f(v|X)$ for all $X \subseteq Y \subset V$ and $v \notin Y$. Therefore, $h(v|Y) \leq \frac{1}{1-\kappa^g} h(v|X)$ for all $X \subset Y \subseteq V$ and $v \notin Y$.

(iii) Let $X \setminus Y$ be $\{v_1, \ldots, v_m\}$, $h(X|Y) = \sum_{i=1,2,\ldots,m} h(v_i|Y \cup \{v_1\} \cup \{v_2\} \cup \ldots \cup \{v_{i-1}\}) \geq (1 - \kappa_f) \sum_{i=1,2,\ldots,m} h(v_i|Y) = (1 - \kappa_f) \sum_{v \in X \setminus Y} h(v|Y)$, according to Lemma (i).

(iv) Let $X \setminus Y$ be $\{v_1, \ldots, v_m\}$, $h(X|Y) = \sum_{i=1,2,\ldots,m} h(v_i|Y \cup \{v_1\} \cup \{v_2\} \cup \ldots \cup \{v_{i-1}\}) \leq \frac{1}{1-\kappa^g} \sum_{i=1,2,\ldots,m} h(v_i|Y) = \frac{1}{1-\kappa^g} \sum_{v \in X \setminus Y} h(v|Y)$, according to Lemma (ii).

□

Lemma 3. *For any chain of solutions $\emptyset = S_0 \subset S_1 \subset \ldots \subset S_k$, where $|S_i| = i$, the following holds for all $i = 0 \ldots k-1$,*

$$h(X^*) \leq \kappa_f \sum_{j: s_j \in S_i \setminus X^*} a_j + \sum_{j: s_j \in S_i \cap X^*} a_j + h(X^* \setminus S_i | S_i) \tag{2.4}$$

where $\{s_i\} = S_i \setminus S_{i-1}$, $a_i = h(s_i|S_{i-1})$ and X^* is the optimal set.

Proof. For any $i = 0, \ldots, k-1$, we focus on the term $h(X^* \cup S_i)$.

According to basic set operations,

$$h(X^* \cup S_i) = h(S_i) + h(X^*|S_i) = \tag{2.24}$$

$$= \sum_{j: s_j \in S_i \setminus X^*} a_j + \sum_{j: s_j \in S_i \cap X^*} a_j + h(X^* \setminus S_i | S_i). \tag{2.25}$$

We can also express $h(X^* \cup S_i)$ the other way around, $h(X^* \cup S_i) = h(X^*) + h(S_i \setminus X^* | X^*)$. Since we already have an order of element in S_i, we can expand $h(S_i \setminus X^* | X^*)$. When adding s_j to the context $S_{j-1} \cup X^*$ we do not need add elements that are not in $S_i \setminus X^*$ since $h(s_j | X^* \cup S_{j-1}) = 0$ if $s_j \in X^*$. Thus, using Lemma 9 (i), we get $h(X^* \cup S_i) = h(X^*) + \sum_{j:s_j \in S_i \setminus X^*} h(s_j | X^* \cup S_{j-1}) \geq h(X^*) + (1 - \kappa_f) \sum_{j:s_j \in S_i \setminus X^*} h(s_j | S_{j-1})$.

Therefore, we have inequalities on both sides of $h(X^* \cup S_i)$ and we can join them together to get:

$$h(X^*) + (1 - \kappa_f) \sum_{j:s_j \in S_i \setminus X^*} a_j \leq \kappa_f \sum_{j:s_j \in S_i \setminus X^*} a_j + \sum_{j:s_j \in S_i \cap X^*} a_j + h(X^* \setminus S_i | S_i), \tag{2.26}$$

or

$$h(X^*) \leq \kappa_f \sum_{j:s_j \in S_i \setminus X^*} a_j + \sum_{j:s_j \in S_i \cap X^*} a_j + h(X^* \setminus S_i | S_i). \tag{2.27}$$

□

2.C Proof of Lemma 4

We begin with an important lemma that is used both for the proof of Lemma 4 and also for the proof of Theorem 3.

Lemma 10. *Given any chain of solutions $\emptyset = S_0 \subset S_1 \subset \ldots \subset S_k$ such that $|S_i| = i$, if the following holds for all $i = 0 \ldots k-1$:*

$$h(X^*) \leq \alpha \sum_{j:s_j \in S_i \setminus X^*} a_j + \sum_{j:s_j \in S_i \cap X^*} a_j + \frac{k - |X^* \cap S_i|}{1 - \beta} a_{i+1} \tag{2.28}$$

where $0 \leq \alpha, \beta \leq 1$ and $s_i = S_i \setminus S_{i-1}$, and $a_i = h(s_i | S_{i-1})$, then we have

$$h(S_k) \geq \frac{1}{\alpha} \left[1 - \left(1 - \frac{(1-\beta)\alpha}{k} \right)^k \right] h(X^*). \tag{2.29}$$

Proof. Assume $\beta < 1$ as otherwise the bound is immediate. This lemma aims to show one inequality (Equation (2.29)) based on k other inequalities (Equation (2.28)) with k variables $a_1, \ldots, a_k$. In the inequalities, $s_j \in S_k \cap X^*$ and $s_j \in S_k \setminus X^*$ are not treated identically.

We will, in fact, correspondingly treat the indices of the elements in $S_k \cap X^*$ as parameters. Recall, $S_k = \{s_1, s_2, \ldots, s_k\}$ is an ordered set and S_k has index set $\{1, 2, \ldots, k\} = [k]$. Let $B = \{b_1, \ldots, b_p\} \subseteq [k]$ be the set of indices of $S_k \cap X^*$ where b_i's are in increasing order (so $b_i < b_{i+1}$) and $p = |S_k \cap X^*|$. Thus, $i \in B$ means $s_i \in S_k \cap X^*$, and $i \in [k] \setminus B$ means $s_i \in S_k \setminus X^*$.

Our next step is to view this problem as a set of parameterized (by B) linear programming problems. Each linear programming problem is characterized as finding:

$$T(B) = T(b_1, b_2, \ldots, b_p) = \min_{a_1, a_2, \ldots, a_k} \sum_{i=1}^{k} a_i \tag{2.30}$$

subject to

$$h(X^*) \leq \alpha \sum_{j \in [i-1] \setminus B_{i-1}} a_j + \sum_{j \in B_{i-1}} a_j + \frac{k - |B_{i-1}|}{1 - \beta} a_i, \text{ for } i = 1, \ldots, k. \tag{2.31}$$

where $B_i = \{b \in B | b \leq i\}$. In this LP problem, $a_1, \ldots, a_k$ are non-negative variables, and k, α, β and $h(X^*)$ are fixed values. Different indices $B = \{b_1, b_2, \ldots, b_p\}$ define different LP problems, and our immediate goal is to show that $T(\emptyset) \leq T(b_1, b_2, \ldots, b_p)$ for all $b_1, b_2, \ldots, b_p$ and $p \geq 0$. In the below, we will use $\Upsilon(B, a, i)$ to refer to the right hand side of Equation (2.31) for a given set B, vector a, and index $i = 1, \ldots, k$, and hence Equation (2.31) becomes $h(X^*) \leq \Upsilon(B, a, i)$ for $i = 1, \ldots, k$. Note that $\Upsilon(B, a, i)$ is linear in a with non-negative coefficients.

First, we show that there exists an optimal solution[4] $a_1, a_2, \ldots, a_k$ s.t. for all $r \leq k - 1$ with $r \in B$, $a_r \leq a_{r+1}$. Let r_a be the largest r s.t. $r \leq k - 1$, $r \in B$ and $a_r > a_{r+1}$; if such an r does not exist, let $r_a = 0$. Our goal here is equivalent to showing, for any feasible solution $\{a_i\}_{i=1}^k$ with $r_a > 0$, we can create another feasible solution $\{a'_i\}_{i=1}^k$ with $r_{a'} = 0$ and the objective $\sum_{i=1}^k a'_i \leq \sum_{i=1}^k a_i$. We do this iteratively, by in each step showing that for any feasible solution $\{a_i\}_{i=1}^k$ with $r_a > 0$, we can create another feasible solution $\{a'_i\}_{i=1}^k$ with $r_{a'} \leq r_a - 1$ and with objective having $\sum_{i=1}^k a'_i \leq \sum_{i=1}^k a_i$. Repeating this argument leads ultimately to $r_{a'} = 0$.

[4]Optimal in this case means for the LP, distinct from the optimal BP maximization solution X^*.

Let $r = r_a$ for notational simplicity. Consider the r^{th} and $(r+1)^{\text{th}}$ inequalities:

$$h(X^*) \leq \alpha \sum_{j \leq [r-1] \setminus B_{r-1}} a_j + \sum_{j \in B_{r-1}} a_j + \frac{k - |B_{r-1}|}{1-\beta} a_r \tag{2.32}$$

and

$$h(X^*) \leq \alpha \sum_{j \leq [r-1] \setminus B_{r-1}} a_j + \sum_{j \in B_{r-1}} a_j + a_r + \frac{k - |B_{r-1}| - 1}{1-\beta} a_{r+1}. \tag{2.33}$$

Since $a_r > a_{r+1}$ and $\beta < 1$, $\frac{k-|B_{r-1}|}{1-\beta} a_r > \frac{k-|B_{r-1}|-1}{1-\beta} a_{r+1} + a_r$ and thus the r.h.s. of Eq. (2.32) is always strictly larger than the r.h.s. of Eq. (2.33).

Therefore, Eq. (2.32) is not tight and it is possible to decrease a_r a little bit. Let $\{a'_i\}$ be another set of solutions with $a'_i = a_i$ for all $i = 1, 2, \ldots, r-1$; $a'_r = a_r - \epsilon$; $a'_i = a_i + \epsilon/(k - |B_r|)$ for $i = r+1, r+2, \ldots, k$ and $\epsilon = \left[1 - \frac{1-\beta}{k-|B_{r-1}|}\right][a_r - a_{r+1}]$. It is easy to see that $\epsilon > 0$ since $|B_{r-1}| \leq r - 1 \leq k - 2$.

Below, we show that $a'_r \leq a'_{r+1}$. First, we notice $\sum_{i=1}^k a'_i \leq \sum_{i=1}^k a_i$ since $|B_r| \leq r$ and $-\epsilon + \frac{k-r}{k-|B_r|}\epsilon \leq 0$. Next, we want to show that $a'_1, a'_2, \ldots, a'_k$ is still feasible. As mentioned above, define $\Upsilon(B, a, i) = \alpha \sum_{j \in [i-1] \setminus B_{i-1}} a_j + \sum_{j \in B_{i-1}} a_j + \frac{k-|B_{i-1}|}{1-\beta} a_i$.

We examine if $h(X^*) \leq \Upsilon(B, a', i)$ or not for each i.

1. For $i = 1, 2, \ldots, r-1$, $\Upsilon(B, a', i) = \Upsilon(B, a, i) \geq h(X^*)$.

2. For $i = r$, $\Upsilon(B, a', r) - \Upsilon(B, a, r+1) = \frac{k-|B_{r-1}|}{1-\beta}[a_r - \epsilon] - a_r - \frac{k-|B_{r-1}|-1}{1-\beta} a_{r+1} \geq \frac{k-|B_{r-1}|}{1-\beta}[a_r - a_{r+1}] + a_{r+1} - a_r - \frac{k-|B_{r-1}|}{1-\beta}\epsilon = \left[\frac{k-|B_{r-1}|}{1-\beta} - 1\right][a_r - a_{r+1}] - \frac{k-|B_{r-1}|}{1-\beta}\left[1 - \frac{1-\beta}{k-|B_{r-1}|}\right][a_r - a_{r+1}]$ 0. So $\Upsilon(B, a', r) \geq \Upsilon(B, a, r+1) \geq h(X^*)$.

3. For $i = r+1, r+2, \ldots, k$, we compare $\Upsilon(B, a', i)$ with $\Upsilon(B, a, i)$. Note that $\Upsilon(B, a, i) = \alpha \sum_{j \in [i-1] \setminus B_{i-1}} a_j + \sum_{j \in B_{i-1}} a_j + \frac{k-|B_{i-1}|}{1-\beta} a_i$ and it has three terms, that we consider individually.

   3.1. The first term is not decreasing since $a'_i < a_i$ only if $i = r$, but $r \notin [i-1] \setminus B_{i-1}$. The increment therefore is at least 0.

3.2. a_r appears in the second term once, and when changing to a'_r, will decreases the value by ϵ. However, $a'_j = a_j + \epsilon/(k - |B_r|)$ for all $j = r+1, r+2, \ldots, k$. Immediately, we notice the number of such a_j in the second term is $\sum_{j \in B_{i-1}, j \geq r+1} 1 = \sum_{j \in B_{i-1}, j \notin B_r} 1 = |B_{i-1}| - |B_r|$. So the increment of the second term is $\frac{|B_{i-1}|-|B_r|}{k-|B_r|}\epsilon - \epsilon$.

3.3. The third term is increased by $\frac{k-|B_{i-1}|}{(1-\beta)(k-|B_r|)}\epsilon \geq \frac{k-|B_{i-1}|}{k-|B_r|}\epsilon$.

So overall, the increment is greater than or equal to $\frac{|B_{i-1}|-|B_r|}{k-|B_r|}\epsilon - \epsilon + \frac{k-|B_{i-1}|}{k-|B_r|}\epsilon \geq 0$, which means $\Upsilon(B, a', i) \geq \Upsilon(B, a, i) = h(X^*)$.

Therefore, $\{a'_i\}_{i=1}^k$ still satisfies all the constraints but $\sum_{i=1}^k a'_k \leq \sum_{i=1}^k a_k$. Note that $r = r_a = \max(\{r' \in B | r' \leq k-1, a_{r'} > a_{r'+1}\})$ by definition. And we have $a'_i = a_i + \frac{\epsilon}{k-|B_r|}$ for $i = r+1, r+2, \ldots, k$. Therefore, $a'_{r'} \leq a'_{r'+1}$ for any $r' \in B \cap [r+1, k-1]$. Next we calculate $a'_r - a'_{r+1} = a_r - a_{r+1} - \epsilon - \frac{\epsilon}{k-|B_r|} = \left[1 - \left(1 + \frac{1}{k-|B_r|}\right)\left(1 - \frac{1-\beta}{k-|B_{r-1}|}\right)\right](a_r - a_{r+1}) \leq 0$. Therefore, $a'_{r'} \leq a'_{r'+1}$ for all $r' \in B \cap [r, k-1]$ which implies $r_{a'} \leq r_a - 1$.

By repeating the above steps, we can get a feasible solution $\{a''\}$ s.t. $r_{a''} = 0$ and $\sum_{i=1}^k a''_k \leq \sum_{i=1}^k a_k$. Therefore, from any optimal solution $\{a_i\}_{i=1}^k$, we can also create another optimal solution $\{a''_i\}$ s.t. for all $r \in B$ and $r \leq k-1$, we have $a''_r \leq a''_{r+1}$. W.l.o.g, we henceforth consider only the optimal solutions $\{a_i\}_{i=1}^k$ with $r_a = 0$.

Second, we assume $r \in B$ but $r+1 \notin B$ for some $r \leq k-1$. We can create $B' = B \cup \{r+1\} \setminus \{r\}$ and show for all $\{a_i\}_{i=1}^k$ that satisfies the constraints of B, $\{a_i\}_{i=1}^k$ will also still satisfy the constraints of B' by showing that $\Upsilon(B', a, i) \geq \Upsilon(B, a, i)$ for $i = 1, \ldots, k$. We consider each i in turn.

1. For $i = 1, 2, \ldots, r$, $\Upsilon(B, a, i) = \Upsilon(B', a, i)$.

2. If $i = r+1$, we notice a_r moves from the second term to the first, and the third term is changed from $\frac{k-|B_r|}{1-\beta}a_{r+1}$ to $\frac{k-|B'_r|}{1-\beta}a_{r+1}$ and $|B'_r| = |B_r| - 1$. So the overall value is increased by $\Upsilon(B', a, i) - \Upsilon(B, a, i) = \frac{1}{1-\beta}a_{r+1} - (1-\alpha)a_r \geq 0$ since $a_r \leq a_{r+1}$.

3. For $i = r+2, r+3, \ldots, k$, we notice that the third term does not change but a_r moves from the second term to the first and a_{r+1} moves from the first term to the second. Thus, the value is increased by $\Upsilon(B', a, i) - \Upsilon(B, a, i) = (1-\alpha)(a_{r+1} - a_r) \geq 0$ since $a_r \leq a_{r+1}$.

Since $\Upsilon(B', a, i) \geq \Upsilon(B, a, i)$ for $i = 1, \ldots, k$, we have that $T(B') \leq T(B)$. Therefore, if we see two indexes in B differ by at least 2, we can increase the first index by 1. Repeating this process, we get

$$T(B) \geq T(k-p+1, k-p+2, \ldots, k). \tag{2.34}$$

Third, if $\{a_i\}_{i=1}^k$ satisfies the constraints for $B = \{k-p+1, k-p+2, \ldots, k\}$and $a_{k-p+1} \leq \ldots \leq a_k$, then $\{a_i\}_{i=1}^k$ also must satisfy the constraints for $B' = \{k-p+2, k-p+3, \ldots, k\}$. We show that $\Upsilon(B', a, i) \geq \Upsilon(B, a, i)$ for $i = 1, \ldots, k$ and again consider each i in turn.

1. For $i = 1, 2, \ldots, k-p+1$, $\Upsilon(B', a, i) = \Upsilon(B, a, i)$.

2. For $i = k-p+2, k-p+3, \ldots, k$, the change of the value is $\Upsilon(B', a, i) - \Upsilon(B, a, i) = (\alpha-1)a_{k-p+1} + \frac{1}{1-\beta}a_i$. We notice that $a_i \geq a_{k-p+1}$ since $k-p+1, k-p+2, \ldots, i-1 \in B$. Thus, we have $\Upsilon(B', a, i) - \Upsilon(B, a, i) \geq 0$ and correspondingly $T(B) \geq T(B')$.

Repeating this process, therefore, we have that

$$T(B) \geq T(\emptyset) \tag{2.35}$$

Finally, we calculate $T(\emptyset)$. For $B = \emptyset$ and any feasible (for Equation (2.31)) $a_1, a_2, \ldots, a_k$, let T_i be the partial sum $T_i = \sum_{j=1}^{i} a_j$ for $i = 0, \ldots, k$ with $T_0 = 0$. We get, for $i = 1, \ldots, k$ that $h(X^*) \leq \Upsilon(\emptyset, a, i)$ which takes the form

$$h(X^*) \leq \alpha \sum_{j \in [i-1]} a_j + \frac{k}{1-\beta} a_i, \tag{2.36}$$

which is the same as

$$h(X^*) \leq \alpha T_{i-1} + \frac{k}{1-\beta}(T_i - T_{i-1}), \tag{2.37}$$

and also, after multiplying both sides by $(1-\beta)/k$ and then adding $(1/\alpha)h(X^*)$ to both sides, the same as

$$\frac{1}{\alpha}h(X^*) - T_i \leq \left(1 - \frac{(1-\beta)\alpha}{k}\right)\left(\frac{1}{\alpha}h(X^*) - T_{i-1}\right). \tag{2.38}$$

$$\tag{2.39}$$

We then repeatedly apply all k inequalities from $i = k, \ldots, 1$, to get

$$\frac{1}{\alpha}h(X^*) - T_k \leq \left(1 - \frac{(1-\beta)\alpha}{k}\right)^k \left(\frac{1}{\alpha}h(X^*) - T_0)\right) \tag{2.40}$$

yielding

$$T_k \geq \frac{1}{\alpha}\left[1 - \left(1 - \frac{(1-\beta)\alpha}{k}\right)^k\right] h(X^*). \tag{2.41}$$

Let $\gamma = \frac{1}{\alpha}\left[1 - \left(1 - \frac{(1-\beta)\alpha}{k}\right)^k\right]$. So, for $B = \emptyset$ and any feasible $a_1, a_2, \ldots, a_k$, we have $\sum_{j=1}^k a_j = T_k \geq \gamma h(X^*)$. Therefore $T(\emptyset) = \min_{a_1,a_2,\ldots,a_k} \sum_{i=1}^k a_i \geq \gamma h(X^*)$.

Recall that $T(B) \geq T(\emptyset)$ for all B. We thus have, with $a_i = h(s_i|\{s_1, \ldots, s_{i-1}\})$ (which are also feasible for Equation (2.31) with B again the indices of $S_k \cap X^*$, which follows from Equation 2.43), $h(S_k) = \sum_i^k a_i \geq T(B) \geq T(\emptyset) \geq \gamma h(X^*)$. □

Lemma 4. GreedMax *is guaranteed to obtain a solution* $\hat{X}$ *such that*

$$h(\hat{X}) \geq \frac{1}{\kappa_f}\left[1 - \left(1 - \frac{(1-\kappa^g)\kappa_f}{k}\right)^k\right] h(X^*) \tag{2.5}$$

where $X^* \in \operatorname{argmax}_{|X|\leq k} h(X)$, $h(X) = f(X) + g(X)$, κ_f *is the curvature of submodular* f *and* κ^g *is the curvature of supermodular* g.

Proof. According to Lemma 3, for all $i = 0, \ldots, k-1$,

$$h(X^*) \leq \kappa_f \sum_{j:s_j \in S_i \setminus X^*} a_j + \sum_{j:s_j \in S_i \cap X^*} a_j + h(X^* \setminus S_i | S_i) \tag{2.42}$$

Since GreedMax is choosing the feasible element with the largest gain, we have $h(v|S_i) \leq h(s_{i+1}|S_i)$ for all feasible $v \in X^*$. In fact, all elements in $X^* \setminus S_j$ are feasible since we are

considering a cardinality constraint and $|S_j| \leq k-1$. Also, $|X^* \setminus S_j| = |X^*| - |X^* \cap S_j| = k - |X^* \cap S_j|$, and therefore from Lemma 3 and Lemma 9(iv), we have that:

$$h(X^*) \leq \kappa_f \sum_{j:s_j \in S_i \setminus X^*} a_j + \sum_{j:s_j \in S_i \cap X^*} a_j + \frac{k - |X^* \cap S_i|}{1-\kappa^g} a_{i+1} \tag{2.43}$$

Lemma 10 then yields Equation (2.5) which shows the result for Lemma 4. □

2.D Proof of Theorem 2

Theorem 2. Theoretical guarantee in the p matroids case. GREEDMAX *is guaranteed to obtain a solution $\hat{X}$ such that*

$$h(\hat{X}) \geq \frac{1-\kappa^g}{(1-\kappa^g)\kappa_f + p} h(X^*) \tag{2.10}$$

where $X^* \in \operatorname{argmax}_{X \in \mathcal{M}_1 \cap \ldots \cap \mathcal{M}_p} h(X)$, $h(X) = f(X) + g(X)$, κ_f is the curvature of submodular f and κ^g is the curvature of supermodular g.

Proof. The greedy procedure produces a chain of solutions $S_0, S_1, \ldots, S_k$ such that $|S_i| = i$, $S_i \subset S_{i+1}$, where k is the iteration after which any addition to S_k is infeasible in at least one matroid, and hence[5] $|\hat{X}| = k$. Immediately, we notice all S_i and X^* are independent sets for all p matroids.

For $j = 0, \ldots, k$ and $l = 1, \ldots, p$, there exist at least $\max(|X^*| - j, 0)$ elements $v \in X^* \setminus S_j$ s.t. $v \notin S_j$ and $S_j + v \in \mathcal{I}(M_l)$, which follows from the third property in the matroid definition. Therefore, for $j = 0, \ldots, k-1$, $l = 1, \ldots, p$, there are at most j elements of X^* that can not be added to S_j.

We next consider the intersection of all p matroids. For $j = 0, \ldots, k$, since in each matroid, there are at most j elements of X^* that cannot be added to S_j, the total possible number of elements for which there exists at least one matroid preventing us from adding to S_j is jp (the case that the p sets of at most j elements are disjoint). In other words, there are at least $\max(|X^*| - pj, 0)$ different $v \in |X^*|$ s.t. $v \notin S_j$, $S_j \cup \{v\} \in \mathcal{M}_1 \cap \ldots \cap \mathcal{M}_p$.

[5]There should be no confusion here that the k we refer to in this section is not any cardinality constraint, but rather the size of the greedy solution.

We claim $|X^*| \leq pk$ as otherwise, by setting $j = k$ above, there are still feasible elements in $X^* \setminus S_k$ in the context of S_k, which indicates that GreedMax has not ended at iteration k. Therefore, we are at liberty to create $pk - |X^*|$ dummy elements, that are always feasible (i.e., independent in all matroids) and that have $h(v|X) = 0$ for all $X \subset V$ for each dummy v. We add these dummy elements to X^* and henceforth assume, w.l.o.g., that $|X^*| = pk$.

We next form an ordered k-partition of $X^* = X_0 \cup X_1 \cup \ldots \cup X_{k-1}$. We show below that it is possible to form this partition so that it has the following properties for $j = 0, \ldots, k-1$:

1. $|X_j| = p$;

2. for all $v \in X_j$, we have $v \notin S_j$ and $S_j \cup \{v\} \in \mathcal{M}_1 \cap \ldots \cap \mathcal{M}_p$ (i.e., v can be added to S_j);

3. and for all j s.t. $s_{j+1} \in X^* \cap S_k$, we have $s_{j+1} \in X_j$.

Immediately, we notice that property 3 is compatible with property 2.

We construct this partition in an order reverse from that of the greedy procedure, that is we create X_j from $j = k-1$ to 0. Recall that, at each step with index $j = k-1, k-2, \ldots, 0$, there are at least $|X^*| - pj = p(k-j)$ elements in X^* can be added to S_j.

When $j = k-1$, there are at least p candidate elements[6] in X^* and we choose p of them to form X_{k-1}. The element s_k can be added to S_{k-1} because the greedy algorithm only adds feasible elements and hence, if also $s_k \in X^*$, then s_k can be one of the elements in X_{k-1}. Thus, abiding property 3 above, we place $s_k \in X_{k-1}$.

Continuing, for $j = k-2, k-3, \ldots, 0$, there are at least p candidate elements in $X^* \setminus [X_{k-1} \cup X_{k-2} \cup \ldots \cup X_{j+1}]$ since $|X_{k-1} \cup X_{k-2} \cup \ldots \cup X_{j+1}| = p(k-j-1)$ and we choose p of them for X_j. Moreover, if $s_{j+1} \in X^*$, we notice s_{j+1} may be one of those candidate elements because of the greedy properties and since $s_{j+1} \notin [X_{k-1} \cup X_{k-2} \cup \ldots \cup X_{j+1}]$ (this follows because $s_{j+1} \in S_{j'}$ for any $j' \geq j+1$, so s_{j+1} is not a candidate element at step

[6]Elements that can be added at the given step.

$j' = k-2, \ldots, j+1$). Similar to what was done in step $k-1$, we again choose p candidate elements to form X_j, and, if $s_{j+1} \in X^*$, we place $s_{j+1} \in X_j$.

We then arrive at partition $X^* = X_0 \cup X_1 \cup \ldots \cup X_{k-1}$ with the aforementioned three properties.

Next, we order the elements in $X^* = \{x_1, \ldots, x_{pk}\}$ where $\{x_{jp+1}, x_{jp+2}, \ldots, x_{(j+1)p}\} = X_j$ for $j = 0, 1, \ldots, k-1$. According to greedy, we have $h(x_{jp+t}|S_j) \leq h(s_{j+1}|S_j) = a_{j+1}$ for $t = 1, \ldots, p$. Recall that a_i is defined to be $h(s_i|S_{i-1})$. Moreover, if $x_{jp+t} \in X^* \cap S_k$, we have $x_{jp+t} = s_{j+1}$.

According to Lemma 3 above,

$$h(X^*) \leq \kappa_f \sum_{j:s_j \in S_k \setminus X^*} a_j + \sum_{j:s_j \in S_k \cap X^*} a_j + h(X^* \setminus S_k|S_k) \tag{2.44}$$

$$= \kappa_f \sum_{j:s_j \in S_k \setminus X^*} a_j + \sum_{j:s_j \in S_k \cap X^*} h(s_j|S_{j-1}) + \sum_{i=1}^{pk} h(x_i|S_k \cup \{x_1\} \ldots \cup \{x_{i-1}\}) \mathbf{1}_{\{x_i \in X^* \setminus S_k\}} \tag{2.45}$$

$$\leq \kappa_f \sum_{j:s_j \in S_k \setminus X^*} a_j + \frac{1}{1-\kappa^g} \sum_{j:s_j \in S_k \cap X^*} h(s_j|S_{j-1}) + \frac{1}{1-\kappa^g} \sum_{j=0}^{k-1} \sum_{t=1}^{p} h(x_{jp+t}|S_j) \mathbf{1}_{\{x_{jp+t} \in X^* \setminus S_k\}} \tag{2.46}$$

$$= \kappa_f \sum_{j:s_j \in S_k \setminus X^*} a_j + \frac{1}{1-\kappa^g} \left[\sum_{j:s_j \in S_k \cap X^*} h(s_j|S_{j-1}) + \sum_{j=0}^{k-1} \sum_{t=1}^{p} h(x_{jp+t}|S_j) - \sum_{j=0}^{k-1} \sum_{t=1}^{p} h(x_{jp+t}|S_j) \mathbf{1}_{\{x_{jp+t} \in X^* \cap S_k\}} \right] \tag{2.47}$$

$$= \kappa_f \sum_{j:s_j \in S_k \setminus X^*} a_j + \frac{1}{1-\kappa^g} \left[\sum_{j:s_j \in S_k \cap X^*} h(s_j|S_{j-1}) + \sum_{j=0}^{k-1} \sum_{t=1}^{p} h(x_{jp+t}|S_j) - \sum_{j:s_j \in S_k \cap X^*} h(s_j|S_{j-1}) \right] \tag{2.48}$$

$$\leq \kappa_f \sum_{j:s_j \in S_k \setminus X^*} a_j + \frac{1}{1-\kappa^g} \sum_{j=0}^{k-1} \sum_{t=1}^{p} a_{j+1} \tag{2.49}$$

$$\leq \left[\kappa_f + \frac{p}{1-\kappa^g}\right] \sum_{j=0}^{k-1} a_{j+1} = \left[\kappa_f + \frac{p}{1-\kappa^g}\right] h(\hat{X}) \tag{2.50}$$

where $\mathbf{1}_{\{\text{condition}\}}$ equals 1 if the condition is met and is 0 otherwise. Lines (2.45) to (2.46) hold because of Lemma 9 (ii). As for lines (2.47) to (2.48), we notice $x_{jp+t} = s_{j+1}$ if $x_{jp+t} \in X^* \cap S_k$. Lines 2.48 to 2.49 follows via the greedy procedure.

Therefore, we have our result which is

$$h(\hat{X}) \geq \frac{1-\kappa^g}{(1-\kappa^g)\kappa_f + p} h(X^*). \tag{2.51}$$

□

2.E Proof of Theorem 3

Theorem 3. SemiGrad *initialized with the empty set is guaranteed to obtain a solution* $\hat{X}$ *for the cardinality constrained case such that*

$$h(\hat{X}) \geq \frac{1}{\kappa_f} \left[1 - e^{-(1-\kappa^g)\kappa_f}\right] h(X^*) \tag{2.11}$$

where $X^* \in \operatorname{argmax}_{|X| \leq k} h(X)$, $h(X) = f(X) + g(X)$, *&* κ_f *(resp.* κ^g*) is the curvature of* f *(resp.* g*).*

Proof. If SemiGrad is initialized by empty set, we need to calculate the semigradient of g at $\emptyset$. By definition, we have

$$m_{g,\emptyset,1}(Y) = m_{g,\emptyset,2}(Y) = \sum_{v \in Y} g(j) \tag{2.52}$$

So in the first step of SemiGrad, we are optimizing $h'(X) = f(X) + m_g(X) = f(X) + \sum_{v \in X} g(v)$ by GreedMax. We will focus elusively on this step as later iterations can only improve the objective value.

According to Lemma 3, we have

$$h(X^*) \leq \kappa_f \sum_{j:s_j \in S_i \setminus X^*} h(s_j|S_{j-1}) + \sum_{j:s_j \in S_i \cap X^*} h(s_j|S_{j-1}) + h(X^* \setminus S_j|S_j) \tag{2.53}$$

Since GreedMax is choosing the feasible element with the largest gain, in the semi-gradient approximation we have $h'(v|S_i) \leq h'(s_{i+1}|S_i)$ instead of $h(v|S_i) \leq h(s_{i+1}|S_i)$. We get:

$$\begin{aligned}
h(X^* \setminus S_j|S_j) &= f(X^* \setminus S_j|S_j) + g(X^* \setminus S_j|S_j) && (2.54)\\
&\leq \sum_{v \in X^* \setminus S_j} f(v|S_j) + \frac{1}{1-\kappa^g} \sum_{v \in X^* \setminus S_j} g(v) && (2.55)\\
&\leq \frac{1}{1-\kappa^g} \sum_{v \in X^* \setminus S_j} h'(v|S_j) && (2.56)\\
&\leq \frac{1}{1-\kappa^g} \sum_{v \in X^* \setminus S_j} h'(s_{j+1}|S_j) && (2.57)\\
&= \frac{1}{1-\kappa^g} \sum_{v \in X^* \setminus S_j} f(s_{j+1}|S_j) + g(s_{j+1}) && (2.58)\\
&\leq \frac{1}{1-\kappa^g} \sum_{v \in X^* \setminus S_j} f(s_{j+1}|S_j) + g(s_{j+1}|S_j) && (2.59)\\
&= \frac{|X^* \setminus S_j|}{1-\kappa^g} h(s_{j+1}|S_j) && (2.60)
\end{aligned}$$

And hence,

$$h(X^*) \leq \kappa_f \sum_{j:s_j \in S_i \setminus X^*} a_i + \sum_{j:s_j \in S_i \cap X^*} a_i + \frac{k - |X^* \cap S_i|}{1-\kappa^g} s_{i+1}. \tag{2.61}$$

where $a_i = h(s_i|S_{i-1})$. We then use Lemma 10 to finish the proof. □

2.F Proof of Theorem 4

Theorem 4. SemiGrad *initialized with the empty set is guaranteed to obtain a solution* $\hat{X}$*, feasible for the* p *matroid constraints, such that*

$$h(\hat{X}) \geq \frac{1-\kappa^g}{(1-\kappa^g)\kappa_f + p} h(X^*) \tag{2.12}$$

where $X^* \in \text{argmax}_{X \in \mathcal{M}_1 \cap \ldots \cap \mathcal{M}_p} h(X)$, $h = f + g$, *&* κ_f *(resp.* κ^g*) is the curvature of* f *(resp.* g*).*

Proof. If SemiGrad is initialized by empty set, we need to calculate the semigradient of g at $\emptyset$. By definition, we have

$$m_{g,\emptyset,1}(Y) = m_{g,\emptyset,2}(Y) = \sum_{v \in Y} g(j) \tag{2.62}$$

So in the first step of SemiGrad, we are optimizing $h'(X) = f(X) + m_g(X) = f(X) + \sum_{v \in X} g(v)$ by GreedMax. We will focus on this step.

According to Lemma 3, we have

$$h(X^*) \leq \kappa_f \sum_{j: s_j \in S_i \setminus X^*} h(s_j|S_{j-1}) + \sum_{j: s_j \in S_i \cap X^*} h(s_j|S_{j-1}) + h(X^* \setminus S_j|S_j) \tag{2.63}$$

We then follow the proofs of Theorems 2 and 3. The only difference is that in Theorem 2 we have $h(v|S_i) \leq h(s_{i+1}|S_i)$ for all feasible v, but in this proof, we have $h'(v|S_i) \leq h'(s_{i+1}|S_i)$, which does not affect the proof as shown in the proof of Theorem 3. □

2.G Proof of Theorem 5

Lemma 11. *(lemma 4.1 from [118]) Let* R *be a random subset of* V *of size* $\alpha = \frac{x\sqrt{n}}{5}$*, let* $\beta = \frac{x^2}{5}$*, and let* x *be any parameter satisfying* $x^2 = \omega(\ln n)$ *and such that* α *and* β *are integer. Let* $f_1(X) = \min(|X|, \alpha)$ *and* $f_2(X) = \min(\beta + |X \cap \bar{R}|, |X|, \alpha)$*. Any algorithm that makes a polynomial number of oracle queries has probability* $n^{-\omega(1)}$ *of distinguishing the functions* f_1 *and* f_2*.*

Theorem 5. Hardness for cardinality constrained case. *For all* $0 \leq \beta \leq 1$*, there exists an instance of a BP function* $h = f + g$ *with supermodular curvature* $\kappa^g = \beta$ *such that no poly-time algorithm solving Problem 1 with a cardinality constraint can achieve an approximation factor better than* $1 - \kappa^g + \epsilon$*, for any* $\epsilon > 0$*.*

Proof. $\kappa^g = \alpha = 0$ is trivial since no algorithm can do better than 1.

The case when $\kappa^g = 1$ can be proven using the example in Lemma 1. $g(X) = \max\{|X| - k, 0\}$, except for a special set R where $g(R) = 0.5$ and $|R| = k$.

For the other case, we prove this result using the hardness construction from [44, 118]. The intuition is to construct two supermodular functions, g and g' both with curvature κ^g which are indistinguishable[7] with high probability in polynomial many function queries. Therefore, any polynomial time algorithm to maximize $g(X)$ can not find $\hat{X} \subseteq V$ with $|\hat{X}| \leq k$ s.t. $g(\hat{X}) > \max_{X \leq k} g'(X)$; otherwise we will have $g(\hat{X}) > \max_{X \leq k} g'(X) \geq g'(\hat{X})$ which contradicts the indistinguishability. In this case, the approximate ratio $\frac{g(\hat{X})}{OPT} \leq \frac{\text{OPT}'}{\text{OPT}}$ where $\text{OPT} = \max_{X \leq k} g(X)$ and $\text{OPT}' = \max_{X \leq k} g'(X)$. The guarantee, by definition, is the best case approximate ratio and, thus no greater than $\frac{\text{OPT}'}{\text{OPT}}$. If any polynomial algorithm has a guarantee greater than $\frac{\text{OPT}'}{\text{OPT}}$, then it contradicts the information theoretic hardness. This is meaningful if $\text{OPT}' < \text{OPT}$.

Let $g(X) = |X| - \beta \min\{\gamma + |X \cap \bar{R}|, |X|, \alpha\}$ and $g'(X) = |X| - \beta \min\{|X|, \alpha\}$, where $R \subseteq V$ is a random set of cardinality α. Let $\alpha = x\sqrt{n}/5$ and $\gamma = x^2/5$ and let x be any parameter satisfying $x^2 = \omega(\ln n)$ s.t. $\gamma < \alpha$ are positive integers and $\alpha \leq \frac{n}{2} - 1$.[8] g and g' are modular minus submodular functions, which implies supermodularity. Monotonicity follows from $g(v|X), g'(v|X) \geq 0$. Also, $\text{OPT} = \alpha - \beta\gamma > \text{OPT}' = \alpha(1 - \beta)$.

Next, we calculate the supermodular curvature. $g(\emptyset) = g'(\emptyset) = 0$. $g(v) = g'(v) = 1 - \beta$ for all $v \in V$ since $\alpha, \gamma \geq 1$. $g(V \setminus \{v\}) = g'(V \setminus \{v\}) = n - 1 - \beta\alpha$ and $g(V) = g'(V) = n - \beta\alpha$ for all $v \in V$ since $\alpha \leq \frac{n}{2} - 1$. Therefore, $\kappa^g = 1 - \min_{v \in V} \frac{g(v)}{g(v|V-v)} = \beta$. $\kappa^{g'} = 1 - \min_{v \in V} \frac{g'(v)}{g'(v|V-v)} = \beta$. So g and g' are monotone non-decreasing supermodular functions with curvature β. Let $f(X) = 0$ for all X and $h(X) = f(X) + g(X) = g(X)$ is the objective BP function.

Any algorithm that uses a polynomial number of queries can distinguish g and g' with

---

[7] Indistinguishable means for all sets X that the algorithm evaluates, $g(X) = g'(X)$.

[8] These examples and the specific parameters like 5 are adopted from [118].

probability only $n^{-\omega(1)}$ according to Lemma 11 [118]. More precisely, $g(X) > g'(X)$[9] if and only if $\gamma + |X \cap \bar{R}| < |X|$ and $\gamma + |X \cap \bar{R}| < \alpha$. It is equivalent with asking $|X \cap R| > \gamma$ and $|X \cap \bar{R}| < \alpha - \gamma$. Moreover, $\Pr(g(X) \neq g'(X))$, where randomness is over random subsets $R \subseteq V$ of size α, is maximized when $|X| = \alpha$ [118]. In this case, the two conditions become identical, and since $|X| = |X \cap \bar{R}| + |X \cap R|$, the condition $g(X) > g'(X)$ happens when only $|X \cap R| > \gamma$. Intuitively, $E|X \cap R| = \frac{\alpha^2}{n} = \frac{\gamma}{5}$ where R is a random set (of arbitrary size) and X is an arbitrary but fixed set of size α. So $|X \cap R|$ is located in small interval around $\frac{\gamma}{5}$ and is hardly ever be larger than γ for large n according to the law of large numbers. While this is only the intuition, a similar reasoning in [118] offers more details.

Therefore, the output $\hat{X}$ of any polynomial algorithm must satisfies $g(\hat{X}) \leq \max_{X \leq k} g'(X)$ since, otherwise the algorithm actually distinguishes the two function at $\hat{X}$, $g(\hat{X}) > \max_{X \leq k} g'(X) \geq g'(\hat{X})$. The approximate ratio $\frac{g(\hat{X})}{\mathrm{OPT}} \leq \frac{\mathrm{OPT}'}{\mathrm{OPT}} = \frac{\alpha - \kappa^g \alpha}{\alpha - \kappa^g \gamma} = (1 - \kappa^g) \frac{1}{1 - \kappa^g \sqrt{\frac{\omega(\log n)}{n}}} \leq 1 - \kappa^g + \epsilon$. Therefore, the guarantee of any polynomial algorithm, that, by definition, the best case approximate ratio, is no greater than $1 - \kappa^g + \epsilon$ for any $\epsilon > 0$ since, otherwise contradicts the information theoretic hardness.

□

2.H Proof of Theorem 6

Theorem 6. Hardness for p matroids constraint case. *For all $0 \leq \beta \leq 1$, there exists an instance of a BP function $h = f + g$ with supermodular curvature $\kappa^g = \beta$ such that no poly-time algorithm can achieve an approximation factor better than $(1 - \kappa^g)O(\frac{\ln p}{p})$ unless P=NP.*

Proof. Consider the p-set problem [47], let R be the maximum disjoint sets of these p sets. No polynomial algorithm can find a larger number of disjoint sets than $O(\frac{\ln p}{p})|R|$ [47]. Let $k = O(\frac{\ln p}{p})|R|$. So no polynomial algorithm can find a feasible set with size larger than k unless P=NP.

---

[9] Note that $g(X) \geq g'(X)$ for all $X \subseteq V$ for any α and γ.

Let $h(X) = (1-\beta)|X| + \beta \max\{|X| - k, 0\}$. It is easy to check that h is a BP function with $f = 0$ and $g = h$ with $\kappa^g = \beta$.

Therefore, the output $\hat{X}$ of any polynomial algorithm that maximizes h under the p-set constraint (expressible via the intersection of p matroids) must satisfy that $|X| \leq k$ and, therefore, $h(\hat{X}) \leq (1-\beta)k$ unless P=NP. But $h(X^*) \geq h(R) = (1-\beta)|R| + \beta(|R| - k) = |R| - \beta k$.

Thus, the approximate ratio

$$\frac{h(\hat{X})}{h(X^*)} \leq \frac{(1-\beta)k}{|R| - \beta k} \leq \frac{(1-\beta)O(\frac{\ln p}{p})}{1 - \beta O(\frac{\ln p}{p})} \leq \frac{(1-\beta)O(\frac{\ln p}{p})}{\frac{1}{2}} = (1-\kappa^g)O(\frac{\ln p}{p}). \tag{2.64}$$

since the denominator $1 - \beta O(\frac{\ln p}{p}) \geq \frac{1}{2}$ asymptotically and $2O(\frac{\ln p}{p}) = O(\frac{\ln p}{p})$. □

2.I Submodularity Ratio and Generalized Curvature

In this section, we compare the pair κ_f, κ^g of curvatures with the submodularity ratio [20, 8]. We also show that both the generalized curvature introduced in [8] and the submodularity ratio [20] appear to be hard to compute in general under the oracle model. Lastly, we compare the pair κ_f, κ^g with another notion of curvature introduced in [116], showing a simple inequality relationship in general and a correspondence when $h = g$.

2.I.1 Submodularity ratio

The submodularity ratio is defined as

$$\gamma_{U,k}(h) = \min_{L \subseteq U, S: |S| \leq k, S \cap L = \emptyset} \frac{\sum_{x \in S} h(x|L)}{h(S|L)} \tag{2.65}$$

with $U \subseteq V$ and $1 \leq k \leq |V| = n$, and typically we consider $\gamma_{V,n}$. We can establish a simple lower bound of the submodularity ratio based on the supermodular curvature as follows.

Lemma 12. *$\gamma_{V,n}(h) \geq 1 - \kappa^g$ when $h = f + g$.*

Proof. For all $L \subseteq V$ and $S \cap L = \emptyset$, we have $\frac{\sum_{x \in S} h(x|L)}{h(S|L)} \geq 1 - \kappa^g$ which follows from Lemma 9(iv) Thus, $\gamma_{V,n}(h) \geq 1 - \kappa^g$. □

The function h is submodular if and only if $\gamma_{V,n} = 1$ so one might hope that given a BP function $h = f + g$, that as $\gamma_{V,n}(h) \to 1$, correspondingly $\kappa^g \to 0$. This is not the case, however, as can be seen by considering the following example.

Let a be an element of V and define the function $g(A) = |A \cap (V \setminus \{a\})| + \epsilon |A \cap (V \setminus \{a\})||A \cap \{a\}|$, where $\epsilon > 0$ is a very small number. Immediately, we have that g being supermodular and monotone. Also note, if $a \notin A$ then $g(A) = |A|$; if $a \in A$ then $g(A) = (|A| - 1)(1 + \epsilon)$.

First, we calculate the supermodular curvature κ^g. We have that $g(a) = 0$ and also $g(a|V \setminus \{a\}) = \epsilon(n - 1)$. Therefore, the function is fully curved, $\kappa^g = 1$.

Next, we calculate the submodularity ratio $\gamma_{V,n} = \min_{L,S \subset V, S \cap L = \emptyset} \frac{\sum_{v \in S} g(v|L)}{g(S|L)}$. When $|S| = 1$, $\frac{\sum_{v \in S} g(v|L)}{g(S|L)} = 1$. When $|S| \geq 2$, we have the following 3 cases (recall that $S \cap L = \emptyset$ so there is no forth case):

- $a \in S$. $g(S|L) = g(S \cup L) - g(L) = (|S| + |L| - 1)(1 + \epsilon) - |L|$ is very close to $|S| - 1$ for very small ϵ. $\sum_{v \in S} g(v|L) = \epsilon |L| + |S| - 1$, which is also very close to $|S| - 1$ for small ϵ. So $\frac{\sum_{v \in S} g(v|L)}{g(S|L)} \approx 1$ for small ϵ.

- $a \in L$. $g(S|L) = g(S \cup L) - g(L) = |S|(1 + \epsilon)$. $\sum_{v \in S} g(v|L) = |S|(1 + \epsilon)$. So $\frac{\sum_{v \in S} g(v|L)}{g(S|L)} = 1$

- $a \notin S \cup L$. $g(S|L) = |S|$ and $\sum_{v \in S} g(v|L) = |S|$. Therefore, $\frac{\sum_{v \in S} g(v|L)}{g(S|L)} = 1$.

In all cases, $\frac{\sum_{v \in S} g(v|L)}{g(S|L)}$ is either 1 or very close to 1 for small ϵ, so $\gamma_{V,n}$ has only 1 as an upper bound. That is, we have an example function that is purely supermodular and fully curved ($\kappa^g = 1$) for all non-zero values of ϵ, but the submodularity ratio can be arbitrarily close to 1. If we consider a weighted sum of a submodular function and this supermodular function, the submodularity ratio is again arbitrarily close to 1. Therefore, there does not seem to be an immediately accessible strong relationship between the supermodular curvature and the submodularity ratio.

2.I.2 Hardness of Generalized Curvature and Submodularity Ratio

The generalized curvature Bian et al. [8] of a non-negative function h is the smallest scalar α s.t.

$$h(v|S \setminus \{v\} \cup \Omega) \geq (1-\alpha)h(v|S \setminus \{v\}) \tag{2.66}$$

for all $S, \Omega \subseteq V$ and $v \in S \setminus \Omega$ and this is used, in concert with the submodularity ratio, to produce bounds such as $\frac{1}{\alpha}(1-e^{-\alpha\gamma})$ for the greedy algorithm. Unfortunately, the generalized curvature is hard to compute under the oracle model. We have the following.

Lemma 13. *There exists an instance of a non-negative function h whose generalized curvature can not be calculated in polynomial time, when we have only oracle access to the function.*

Proof. We consider a non-negative function $h' : 2^V \to R$ with ground set size equals n (n is even number). Let $h'(X) = |X|$ for all $X \subseteq V$. Let $R \subseteq V$ be an arbitrary set with $|R| = \frac{n}{2}$. Define another set function $h : 2^V \to R$, $h(X) = h'(X)$ for all $X \subseteq V$ and $X \neq R$; $h(R) = \frac{n}{2} - 1$.

First, we can easily calculate the generalized curvature of h' and h. We have that $\alpha_{h'} = 0$ since h' is a non-decreasing modular function. For h, let $S \cup \Omega = R$, $S \cap \Omega = \emptyset$, $|S|, |\Omega| \geq 1$ and $v \in S$, we have $h(v|S \setminus \{v\} \cup \Omega) = 0$ and $h(v|S \setminus \{v\}) = 1$. Therefore $\alpha = 1$ is the smallest scalar s.t. $h(v|S \setminus \{v\} \cup \Omega) \geq (1-\alpha)h(v|S \setminus \{v\})$. So, as a conclusion of this part, the generalized curvature of the two functions are not the same.

Next we use a proof technique similar to [118]. Note that $h'(X) = h(X)$ if and only if $X \neq R$. So for any algorithm trying to calculate α_h, before it evaluates $h(R)$, all function evaluations are the same with calculating $\alpha_{h'}$. Additionally, since $h(X) = |X|$, it is permutation symmetric. Therefore, the algorithm can only do random search to find R. If the algorithm acquires a polynomial number $O(n^m)$ of sets of size $\frac{n}{2}$, the probability of finding R is $\frac{O(n^m)}{\binom{n}{\frac{n}{2}}} \leq \frac{O(n^m)}{(n/\frac{n}{2})^{\frac{n}{2}}} = \frac{O(n^m)}{2^{n/2}} \leq O(2^{-n/2+\epsilon n})$ for all $\epsilon > 0$.

Therefore, no algorithm can be guaranteed to distinguish h and h' in polynomial time. Since the generalized curvature of h and h' are different, neither of them can be calculated in polynomial time.

□

Likewise, the submodularity ratio is unfortunately also hard to compute exactly, in the oracle model.

Lemma 14. *There exists an instance of a non-negative function h whose submodularity ratio (Equation (2.65)) can not be calculated in polynomial time under only oracle access to that function.*

Proof. We consider a non-negative function $h' : 2^V \to R$ with ground set size n (where n is an even number). Let $h'(X) = |X|$ for all $X \subseteq V$. Let $R \subseteq V$ be an arbitrary set with $|R| = \frac{n}{2}$. Define another set function $h : 2^V \to R$, $h(X) = h'(X)$ for all $X \subseteq V$ and $X \neq R$ and $h(R) = \frac{n}{2} - 1$.

We can easily calculate the submodularity ratio of both h' and h as follows. We have that $\gamma_{V,n}(h') = 1$ since h' is a non-decreasing modular (and thus submodular) function. For h, choose an element $v_1 \in R$ and another element $v_2 \in V \setminus R$, and let $L = R \setminus \{v_1\}$ and $S = \{v_1, v_2\}$. We have $\frac{\sum_{v \in S} h(v|L)}{h(S|L)} = \frac{h(R)+h(R\setminus\{v_1\}\cup\{v_2\})-2h(R\setminus\{v_1\})}{h(R\cup\{v_2\})-h(R\setminus\{v_1\})} = \frac{1}{2}$ and thus $\gamma_{V,n}(h) = \min_{L,S\subseteq V, S\cap L=\emptyset} \frac{\sum_{v\in S} h(v|L)}{h(S|L)} \leq \frac{1}{2}$. Therefore, the submodularity ratio of the two functions are not the same. Given the submodularity ratio of the two functions, we would be able to tell them apart.

Next we use a proof technique similar to [118]. We have that $h'(X) = h(X)$ if and only if $X \neq R$. So for any algorithm trying to calculate $\gamma_{V,n}(h)$, before it evaluates $h(R)$, all function evaluations are the same with calculating $\gamma_{V,n}(h')$. Additionally, since $h(X) = |X|$ is permutation symmetric, the algorithm can only do a random search to find R. If the algorithm queries a polynomial number $O(n^m)$ of sets of size $\frac{n}{2}$, the probability of finding R is $\frac{O(n^m)}{\binom{n}{\frac{n}{2}}} \leq \frac{O(n^m)}{(n/\frac{n}{2})^{\frac{n}{2}}} = \frac{O(n^m)}{2^{n/2}} \leq O(2^{-n/2+\epsilon n})$ for all $\epsilon > 0$.

Therefore, no algorithm can guarantee to distinguish h and h' in polynomial time. Since the submodularity ratio of h and h' are different, this means that neither of them can be calculated in polynomial time.

□

2.I.3 Comparison to Sviridenko et al. [116]'s curvature

Sviridenko et al. [116] (in their Section 8) define a notion of curvature as follows:

$$1 - c = \min_{j} \min_{A,B \subseteq V \setminus j} \frac{h(j|A)}{h(j|B)} \tag{2.67}$$

We can establish a simple upper bound on c based on submodular and supermodular curvature. We calculate $\frac{h(j|A)}{h(j|B)}$ given $h = f + g$ and κ_f and κ^g. First, $f(j|B) \leq f(j) \leq \frac{1}{1-\kappa_f} f(j|A)$ which follows from Lemma 9 (i). Thus $\frac{f(j|A)}{f(j|B)} \geq 1 - \kappa_f$. Next, $g(j|A) \geq g(j) \geq (1 - \kappa^g) g(j|B)$ which follows from Lemma 9 (ii). Thus, $\frac{g(j|A)}{g(j|B)} \geq 1 - \kappa^g$. Therefore,

$$\frac{h(j|A)}{h(j|B)} = \frac{f(j|A) + g(j|A)}{f(j|B) + g(j|B)} \geq \frac{(1 - \kappa_f) f(j|B) + (1 - \kappa^g) g(j|B)}{f(j|B) + g(j|B)} \tag{2.68}$$

$$\geq \frac{\min(1 - \kappa_f, 1 - \kappa^g)(f(j|B) + g(j|B))}{f(j|B) + g(j|B)} \geq \min(1 - \kappa_f, 1 - \kappa^g) \tag{2.69}$$

Thus we have $1 - c \geq \min(1 - \kappa_f, 1 - \kappa^g)$, or $c \leq \max(\kappa_f, \kappa^g)$.

Note that for purely supermodular functions, $\kappa_f = 0$ and, considering Equation (2.67), we have $c = \kappa^g$. This coincides with the $1 - \kappa^g$ bound and hardness for monotone supermodular functions — compare Theorem 8.1 of Sviridenko et al. [116] with the present chapter's item 3 in Section 2.3.2 and Theorem 5.

2.J Modular Approximations to g, Weak/Strong Bounds

In this section, we discuss an approach where the greedy algorithm is run, subject either to a cardinality constraint or to multiple matroid constraints, on a submodular function $h' = f + m$ that acts as a surrogate to the BP function $h = f + g$. We show that the result of maximizing this surrogate h' has a bound with respect to the original BP function for both

types of constraints. The modular function m is an approximation to the supermodular function g, and there are a set of conditions that, if they are true, guarantee the bound exists. In fact, we show there are two types of bounds: (1) weak bounds that can be derived using a simple analysis (and which are shown in in Lemma 16 for the cardinality case and Lemma 17 for the multiple matroid case), and also strong bounds which are both tight and that require a much more complex analysis technique (and which are shown in Lemma 7 for the cardinality case and Lemma 8 for the multiple matroid constraints case). These are summarized in Table 2.J.1. Included in the set of modular functions that satisfy the stated conditions are the set of subgradients of g, which are also modular approximations of the supermodular function. We also show that while all of the functions functions yield the same bound, some of the modular approximations are tighter than others, and we correspondingly show an example where optimizing the tighter approximation can lead to an unboundedly better approximation ratio.

	cardinality	multiple matroid
weak	Lemma 16	Lemma 17
strong	Lemma 7	Lemma 8

Table 2.J.1: A table indicating the four lemmas for the weak vs. strong bounds $\times$ the cardinality vs. multiple matroid constraints when optimizing a modular approximation m to g in the submodular surrogate $h' = f + m$.

2.J.1 Modular Approximations

We start by showing various relationships between three possible properties of a given modular function.

Lemma 15. *Consider the following three statements each expressing a property about a given modular function $m : V \to \mathbb{R}$:*

(15.1) : There exists a set of pairs $\{\mu_v, \nu_v\}_{v \in V}$ *such that for all* $v \in V$, $0 < \mu_v \leq 1$, $\nu_v \geq 1$, $\frac{\nu_v}{\mu_v} = \frac{1}{1-\kappa^g}$, *and* $\mu_v g(v|V \setminus \{v\}) \leq m(v) \leq \nu_v g(v)$.

(15.2) : There exists a pair μ, ν *such that* $0 < \mu \leq 1$, $\nu \geq 1$, $\frac{\nu}{\mu} = \frac{1}{1-\kappa^g}$, *and for all* $v \in V$, $\mu g(v|V \setminus \{v\}) \leq m(v) \leq \nu g(v)$.

(15.3) : There exists a pair μ, ν *such that* $0 < \mu \leq 1$, $\nu \geq 1$, $\frac{\nu}{\mu} = \frac{1}{1-\kappa^g}$, *and for all* $X \subseteq V$, $\mu g(X) \leq m(X) \leq \nu g(X)$.

Then given the above, we have that (15.2) $\Rightarrow$ (15.1) *and* (15.2) $\Rightarrow$ (15.3).

Proof. (15.2) $\rightarrow$ (15.1) is trivial. Next, we prove (15.2) $\rightarrow$ (15.3). For all $X \subseteq V$, let $X = \{x_1, x_2, \ldots, x_{|X|}\}$, we have $m(X) = \sum_{i=1}^{|X|} m(x_i)$ and $g(X) = \sum_{i=1}^{|X|} g(x_i|\{x_1, x_2, \ldots, x_{i-1}\})$. Therefore

$$m(X) = \sum_{i=1}^{|X|} m(x_i) \leq \sum_{i=1}^{|X|} \nu g(x_i) \tag{2.70}$$

$$\leq \nu \sum_{i=1}^{|X|} g(x_i|\{x_1, x_2, \ldots, x_{i-1}\}) = \nu g(X) \tag{2.71}$$

and

$$m(X) = \sum_{i=1}^{|X|} m(x_i) \geq \sum_{i=1}^{|X|} \mu g(x_i|V \setminus \{x_i\}) \tag{2.72}$$

$$\geq \mu \sum_{i=1}^{|X|} g(x_i|\{x_1, x_2, \ldots, x_{i-1}\}) = \mu g(X) \tag{2.73}$$

□

2.J.2 Weak and Strong Bounds, Cardinality Constraints

The next result shows we may obtain weak bounds in the cardinality case relatively easily.

Lemma 16 (weak bound, cardinality case)**.** *Condition (15.3) is a sufficient condition on m for the greedy solution $\hat{X}$ resulting from maximizing the surrogate $h' = f + m$ to have the following guarantee:*

$$h(\hat{X}) \geq \frac{1 - \kappa^g}{\kappa_f} \left[1 - e^{-\kappa_f}\right] h(X^*), \tag{2.74}$$

where $X^* \in \operatorname{argmax}_{|X| \leq k} h(X)$ and $h = f + g$.

Proof. Immediately, we notice that $\mu h(X) \leq h'(X) \leq \nu h(X)$ for all $X \subseteq V$. Also, h' is a monotone submodular function with $\kappa_{h'} = 1 - \min_{v \in V} \frac{h'(v|V \setminus \{v\})}{h'(v)} = 1 - \min_{v \in V} \frac{f(v|V \setminus \{v\}) + m(v)}{f(v) + m(v)} \leq 1 - \min_{v \in V} \frac{f(v|V \setminus \{v\})}{f(v)} = \kappa_f$ since $0 \leq f(v|V \setminus \{v\}) \leq f(v)$.

Using the traditional curvature-based bound for submodular maximization [15], the greedy algorithm maximizing h' provides a solution $\hat{X}$ s.t. $h'(\hat{X}) \geq \frac{1}{\kappa_{h'}} \left[1 - e^{-\kappa_{h'}}\right] h'(X'^*) \geq \frac{1}{\kappa_{h'}} \left[1 - e^{-\kappa_{h'}}\right] h'(X^*)$ where $X'^* \in \operatorname{argmax}_{|X| \leq k} h'(X)$ and $X^* \in \operatorname{argmax}_{|X| \leq k} h(X)$. Thus, we have

$$h(\hat{X}) \geq \frac{1}{\nu} h'(\hat{X}) \geq \frac{1}{\nu \kappa_{h'}} \left[1 - e^{-\kappa_{h'}}\right] h'(X^*) \geq \frac{\mu}{\nu \kappa_f} \left[1 - e^{-\kappa_f}\right] h(X^*) \tag{2.75}$$

$$\geq \frac{1 - \kappa^g}{\kappa_f} \left[1 - e^{-\kappa_f}\right] h(X^*). \tag{2.76}$$

□

The above bound can be appreciably improved using the following more complex analysis. The following Lemma uses Lemma 10 to achieve the bound.

Theorem 7 (Strong bound, Cardinality case)**.** *Condition (15.1) is a sufficient condition on m for the greedy solution $\hat{X}$ resulting from maximizing the surrogate $h' = f + m$ to have the following guarantee:*

$$h(\hat{X}) \geq \frac{1}{\kappa_f} \left[1 - e^{-\kappa_f (1 - \kappa^g)}\right] h(X^*), \tag{2.77}$$

where $X^* \in \operatorname{argmax}_{|X| \leq k} h(X)$ and $h = f + g$.

Proof. Let $\emptyset = S_0 \subset S_1 \subset S_2 \cdots \subset S_k$ chain of solutions resulting from running greedy on h' with $|S_i| = i$ and $\{s_{j+1}\} = S_{j+1} \setminus S_j$. Restating condition (15.1), we have that $m(X)$ is modular function satisfying:

$$\mu_v g(v|V \setminus \{v\}) \leq m(v) \leq \nu_v g(v) \tag{2.78}$$

where μ_v, ν_v are real values with $0 < \mu_v \leq 1$, $\nu_v \geq 1$ and $\frac{\nu_v}{\mu_v} = \frac{1}{1-\kappa^g}$ for all v.

According to Lemma 3, for the above chain of sets we have that

$$h(X^*) \leq \kappa_f \sum_{j:s_j \in S_i \setminus X^*} h(s_j|S_{j-1}) + \sum_{j:s_j \in S_i \cap X^*} h(s_j|S_{j-1}) + h(X^* \setminus S_i|S_i). \tag{2.79}$$

Since GreedMax is choosing the next feasible element with the largest gain, in the h' surrogate approximation we have $h'(v|S_i) \leq h'(s_{i+1}|S_i)$ for all $v \in V \setminus S_i$ instead of $h(v|S_i) \leq h(s_{i+1}|S_i)$ which would result if optimizing h directly. We get:

$$\begin{aligned}
h(X^* \setminus S_i|S_i) &= f(X^* \setminus S_i|S_i) + g(X^* \setminus S_i|S_i) && (2.80)\\
&\leq \sum_{v \in X^* \setminus S_i} f(v|S_i) + g(v|V \setminus \{v\}) && (2.81)\\
&\leq \sum_{v \in X^* \setminus S_i} f(v|S_i) + \frac{1}{\mu_v} m(v) && (2.82)\\
&\leq \sum_{v \in X^* \setminus S_i} \frac{1}{\mu_v} h'(v|S_i) && (2.83)\\
&\leq \sum_{v \in X^* \setminus S_j} \frac{1}{\mu_v} h'(s_{i+1}|S_i) && (2.84)\\
&\leq \sum_{v \in X^* \setminus S_j} \frac{1}{\mu_v} \left[f(s_{i+1}|S_i) + \nu_v g(s_{i+1})\right] && (2.85)\\
&\leq \sum_{v \in X^* \setminus S_j} \frac{\nu_v}{\mu_v} \left[f(s_{j+1}|S_i) + g(s_{j+1}|S_i)\right] && (2.86)\\
&= \frac{|X^* \setminus S_i|}{1 - \kappa^g} h(s_{i+1}|S_i) && (2.87)
\end{aligned}$$

And hence,

$$h(X^*) \leq \kappa_f \sum_{j:s_j \in S_i \setminus X^*} a_j + \sum_{j:s_j \in S_i \cap X^*} a_j + \frac{k - |X^* \cap S_i|}{1 - \kappa^g} a_{i+1}. \tag{2.88}$$

where $a_i = h(s_i|S_{i-1})$. We then use Lemma 10 to finish the proof. □

2.J.3 Weak and Strong Bounds, Multiple Matroid Constraints

We next show that the surrogate function $h' = f + m$ with appropriate m when optimized using the greedy procedure using multiple matroids as constraints also has guarantees. Like in the above case, a simple analysis offers weak bounds, while our more complex analysis provides better bounds.

Lemma 17 (Weak bound, multiple matroids case)**.** *Condition (15.3) is a sufficient condition on m for the greedy solution $\hat{X}$ resulting from maximizing the surrogate $h' = f + m$ to have the following guarantee:*

$$h(\hat{X}) \geq \frac{1 - \kappa^g}{\kappa_f + p} h(X^*) \tag{2.89}$$

where $X^* \in \operatorname{argmax}_{X \in \mathcal{M}_1 \cap \ldots \cap \mathcal{M}_p} h(X)$, and $h = f + g$.

Proof. Using a proof strategy similar to Lemma 16, we can show the bound is $1-\kappa^g$ multiplied by the traditional bound of maximizing a submodular function with the same submodular curvature. The latter bound is $\frac{1}{\kappa_f + p}$ [15]. Hence, the total bound is $\frac{1-\kappa^g}{\kappa_f + p}$. □

A much stronger bound can be achieved with a more complex analysis.

Theorem 8 (Strong bound, multiple matroids case)**.** *Condition (15.1) is a sufficient condition on m for the greedy solution $\hat{X}$ resulting from maximizing the surrogate $h' = f + m$ to have the following guarantee:*

$$h(\hat{X}) \geq \frac{1 - \kappa^g}{(1 - \kappa^g)\kappa_f + p} h(X^*) \tag{2.90}$$

where $X^* \in \operatorname{argmax}_{X \in \mathcal{M}_1 \cap \ldots \cap \mathcal{M}_p} h(X)$, $h = f + g$.

Proof. The proof is similar to proof of Theorem 4 and the first part is exactly the same. We order the elements in $X^* = \{x_1, \ldots, x_{pk}\}$. According to greedy, we have $h'(x_{jp+t}|S_j) \leq h'(s_{j+1}|S_j)$ for $t = 1, \ldots, p$. Moreover, if $x_{jp+t} \in X^* \cap S_k$, we have $x_{jp+t} = s_{j+1}$.

According to Lemma 3,

$$h(X^*) \leq \kappa_f \sum_{j: s_j \in S_k \setminus X^*} h(s_j|S_{j-1}) + \sum_{j: s_j \in S_k \cap X^*} h(s_j|S_{j-1}) + h(X^* \setminus S_k|S_k) \tag{2.91}$$

$$= \kappa_f \sum_{j:s_j \in S_k \setminus X^*} h(s_j|S_{j-1}) + \sum_{j:s_j \in S_k \cap X^*} h(s_j|S_{j-1}) + \sum_{i=1}^{pk} h(x_i|S_k \cup \{x_1\} \ldots \cup \{x_{i-1}\}) \mathbf{1}_{\{x_i \in X^* \setminus S_k\}} \tag{2.92}$$

$$\leq \kappa_f h(\hat{X}) + \sum_{j:s_j \in S_k \cap X^*} \frac{h'(s_j|S_{j-1})}{\mu_{s_j}} + \sum_{j=0}^{k-1} \sum_{t=1}^{p} \frac{h'(x_{jp+t}|S_j) \mathbf{1}_{\{x_{jp+t} \in X^* \setminus S_k\}}}{\mu_{x_{jp+t}}} \tag{2.93}$$

$$= \kappa_f h(\hat{X}) + \sum_{j:s_j \in S_k \cap X^*} \frac{h'(s_j|S_{j-1})}{\mu_{s_j}} + \sum_{j=0}^{k-1} \sum_{t=1}^{p} \frac{h'(x_{jp+t}|S_j)}{\mu_{x_{jp+t}}} - \sum_{j=0}^{k-1} \sum_{t=1}^{p} \frac{h'(x_{jp+t}|S_j) \mathbf{1}_{\{x_{jp+t} \in X^* \cap S_k\}}}{\mu_{x_{jp+t}}} \tag{2.94}$$

$$\leq \kappa_f h(\hat{X}) + \sum_{j:s_j \in S_k \cap X^*} \frac{h'(s_j|S_{j-1})}{\mu_{s_j}} + \sum_{j=0}^{k-1} \sum_{t=1}^{p} \frac{h'(x_{jp+t}|S_j)}{\mu_{x_{jp+t}}} - \sum_{j:s_j \in S_k \cap X^*} \frac{h'(s_j|S_{j-1})}{\mu_{s_j}} \tag{2.95}$$

$$= \kappa_f h(\hat{X}) + \sum_{j=0}^{k-1} \sum_{t=1}^{p} \frac{h'(x_{jp+t}|S_j)}{\mu_{x_{jp+t}}} \tag{2.96}$$

$$\leq \kappa_f h(\hat{X}) + \sum_{j=0}^{k-1} \sum_{t=1}^{p} \frac{h'(s_{j+1}|S_j)}{\mu_{x_{jp+t}}} \tag{2.97}$$

$$\leq \kappa_f h(\hat{X}) + \sum_{j=0}^{k-1} \sum_{t=1}^{p} \frac{\nu_{x_{jp+t}} h(s_{j+1}|S_j)}{\mu_{x_{jp+t}}} \tag{2.98}$$

$$\leq \left[\kappa_f + \frac{p}{1-\kappa^g} \right] h(\hat{X}) \tag{2.99}$$

where $\mathbf{1}_{\{\text{condition}\}}$ equals 1 if the condition is met and is 0 otherwise. Lines 2.92 to 2.93 hold because of Lemma 9 (ii). As for lines 2.94 to 2.96, we notice $x_{jp+t} = s_{j+1}$ if $x_{jp+t} \in X^* \cap S_k$. Lines 2.96 to 2.97 follow via the greedy procedure.

Therefore, we have our result which is

$$h(\hat{X}) \geq \frac{1-\kappa^g}{(1-\kappa^g)\kappa_f + p} h(X^*). \tag{2.100}$$

□

2.J.4 Modular Instances

There are a number of ways to easily produce modular approximations m to g that satisfy one or more of the three conditions in Lemma 15. Define the following modular approximations:

$$m(X) = (1 - \kappa^g) \sum_{v \in X} g(v|V \setminus \{v\}) \tag{2.101}$$

Lemma 18. *$m(X) = (1 - \kappa^g) \sum_{v \in X} g(v|V \setminus \{v\})$, $m_\emptyset(X) = \sum_{v \in X} g(v)$ (the sub-gradient of g at $\emptyset$) and $m_V(X) = \sum_{v \in X} g(v|V \setminus \{v\})$ (the sub-gradient of g at V, ignore the constant term[10]) satisfy the condition (15.2) in lemma 15.*

Lemma 19. *$m_X(Y)$, the subgradient of g at X, ignoring the constant term, satisfy the condition (15.1) in lemma 15*

Proof. The subgradients of supermodular g are given in Equations (2.102) and (2.104), but we restate them again for clarity:

$$m_{g,X,1}(Y) \triangleq g(X) - \sum_{j \in X \setminus Y} g(j|X \backslash j) + \sum_{j \in Y \setminus X} g(j|\emptyset), \tag{2.102}$$

$$= \underbrace{\left[g(X) - \sum_{j \in X} g(j|X \setminus j) \right]}_{\text{const. w.r.t. } Y} + \sum_{j \in X \cap Y} g(j|X \backslash j) + \sum_{j \in Y \setminus X} g(j|\emptyset), \tag{2.103}$$

and

$$m_{g,X,2}(Y) \triangleq g(X) - \sum_{j \in X \setminus Y} g(j|V \backslash j) + \sum_{j \in Y \setminus X} g(j|X) \tag{2.104}$$

$$= g(X) - \sum_{j \in X} g(j|V \backslash j) + \sum_{j \in X \cap Y} g(j|V \backslash j) + \sum_{j \in Y \setminus X} g(j|X). \tag{2.105}$$

If we ignore the terms that are constant w.r.t. Y, we get $m_{g,X,1}(v) = g(v|X \backslash \{v\})$ if $v \in X$ and $m_{g,X,1}(v) = g(v)$ if $v \notin X$. Also, $m_{g,X,2}(v) = g(v|V \setminus \{v\})$ if $v \in X$ and $m_{g,X,1}(v) = g(v|X)$ if $v \notin X$.

---

[10] subtracting $m(\emptyset)$ from m

Thus, for any of the subgradients $m_X \in \{m_{g,X,1}, m_{g,X,2}\}$, we have that $m_X(v) = g(v|A_{X,v} \setminus \{v\})$ for some subset $A_{X,v} \subseteq V$. The set $A_{X,v}$ is dependent on both X and v, but we will not need to know the specific $A_{X,v}$ to finish the proof. The reason is that for any set A, $g(v) \leq g(v|A \setminus \{v\}) \leq \frac{1}{1-\kappa^g} g(v)$ and $(1-\kappa^g) g(v|V \setminus \{v\}) \leq g(v|A \setminus \{v\}) \leq g(v|V \setminus \{v\})$.

Therefore, we have $g(v) \leq m_X(v) \leq \frac{1}{1-\kappa^g} g(v)$ and $(1-\kappa^g) g(v|V \setminus \{v\}) \leq m_X(v) \leq g(v|V \setminus \{v\})$ for any subgradient m_X. Also, we have $\frac{g(v|V \setminus \{v\})}{g(v)} \leq \frac{1}{1-\kappa^g}$.

Abiding condition (15.2) of Lemma 15, let $\mu_v = \frac{m_X(v)}{g(v|V \setminus \{v\})}$ and $\nu_v = \frac{\mu_v}{1-\kappa^g}$. We have $0 \leq \mu_v \leq 1 \leq \nu_v$ and $\frac{\nu_v}{\mu_v} = \frac{1}{1-\kappa^g}$. Also we see that $\mu_v g(v|V \setminus \{v\}) \leq m_X(v) \leq \nu_v g(v)$, the r.h.s. since $m_X(v) g(v) / (g(v|V \setminus \{v\}) \min_{v'} [g(v')/g(v'|V \setminus \{v'\})]) \geq m_X(v) g(v) / (g(v|V \setminus v) g(v)/g(v|V \setminus v)) = m_X(v) g(v)/g(v) = m_X(v)$ which thus stratifies condition (15.1) in Lemma 15.

□

Chapter 3

OPTIMIZATION OF RATIO OF TWO SUBMODULAR (RS) FUNCTIONS

We investigate a new optimization problem involving minimizing the Ratio of two Submodular (RS) functions. We argue that this problem occurs naturally in several real world applications. We then show the connection between this problem and several related problems including minimizing the difference between submodular functions [54, 93], and to submodular optimization subject to submodular constraints [55]. We show that RS optimization can be solved with bounded approximation factors. We also provide a hardness bound and show that our tightest algorithm matches the lower bound up to a log factor. Finally, we empirically demonstrate the performance and good scalability properties of our algorithms. The work was previously published in

Wenruo Bai, *Rishabh Iyer, Kai Wei, and Jeff Bilmes. Algorithms for Optimizing the Ratio of Submodular Functions. In International Conference on Machine Learning (ICML), New York, NY, July 2016.*

3.1 Introduction

In this chapter, we study a new class of discrete optimization problems that have the following form:

Problem 5. $\min_{\emptyset \subset X \subseteq V} \frac{f(X)}{g(X)}$, *where f and g are monotone non-decreasing submodular functions.*

We call the objective $\frac{f(X)}{g(X)}$ a ratio of submodular (RS) function. In this chapter only, we do not require that either f or g is normalized, we only assume $f(\emptyset) \geq 0$ and $g(\emptyset) \geq 0$. We assume, w.l.o.g., that both f and g satisfy $f(v) > 0, g(v) > 0, \forall v \in V$, since we may simply

remove any item $v \in V$ for consideration if $g(v) = 0$ and add any item $v \in V$ to the solution if $f(v) = 0$. As a consequence of this assumption and the monotonicity of f and g, we have that $f(X) > 0, g(X) > 0, \forall \emptyset \subset X \subseteq V$.

We call this problem RS minimization. In Section 3.5, we extend the algorithms here to handle non-monotone submodular functions. We also consider a related problem called RS maximization:

Problem 6. $\max_{\emptyset \subset X \subseteq V} \frac{g(X)}{f(X)}$, *where f and g are monotone non-decreasing submodular functions.*

We observe that RS Minimization and RS Maximization are equivalent, in that an algorithm for Problem 1 also solves Problem 2. To be precise, given an α-approximation algorithm ($\alpha > 1$) for Problem 1, one can achieve a $1/\alpha$ approximation for Problem 2, using the solution obtained by the algorithm for Problem 1.

3.1.1 Applications

In the below, we describe how Problems 5 and 6 appear naturally in several machine learning applications.

Maximizing the F-Measure in Information Retrieval: Consider the problem where one is given a set U of objects (e.g., documents, images etc.) which can be expressed as a bag of words W. One can construct a bipartite graph $G(U, W, E)$, where U is the set of objects, W is the set of words, and the edge $e_{u,w} \in E$ between the object u and the word w exists if the word w is present in the object u. Define a neighborhood function $\Gamma : 2^U \rightarrow 2^W$ on the bipartite graph G that maps from any subset of the objects $X \subseteq U$ to the set of words $\Gamma(X) \subseteq W$ contained in the objects. We are interested in the information retrieval problem of finding a set of objects that cover exactly a set of target words $T \subseteq W$. The quality of any subset $X \subseteq U$ may be measured as the F-Measure of the coverage on the target words

T. More formally, the quality measure can be defined as:

$$F(X) = \frac{2|\Gamma(X) \cap T|}{|T| + |\Gamma(X)|}. \tag{3.1}$$

The goal is to find a set of objects X that maximizes the F-measure $F(X)$. Note that both $|\Gamma(X) \cap T|$ and $|T| + |\Gamma(X)|$ are monotone submodular. Hence the problem is an instance of RS maximization (Problem 6).

Normalized Cuts and Ratio Cuts: Another application of RS optimization is the normalized cut and ratio cut problem, which have been extensively used in image segmentation and clustering [109, 68]. Let $G = (V, E)$ be a similarity graph defined on the set of vertices V, where $w_{a,a'}$ is the edge weight between the vertex a and $a' \in V$ and measures the similarity between these two vertices. Let $h(X) = \sum_{x \in X} \sum_{x' \in V \setminus X} w_{x,x'}$ be the graph cut function defined on the graph G, and let $m(X) = \sum_{x \in X} \sum_{v \in V} w_{x,v}$ be the function that measures the association of the subset A to the ground set V. The normalized cut problem as defined in [109] is to minimize the following objective:

$$\frac{h(X)}{m(X)} + \frac{h(X)}{m(V) - m(X)}, \tag{3.2}$$

which can further simplified as follows:

$$\frac{h(X)m(V)}{m(X)(m(V) - m(X))}. \tag{3.3}$$

Note that $m(V)$ is a constant, and both $h(X)$ and $m(X)(m(V) - m(X))$ are symmetric submodular functions. Therefore, the normalized cuts problem can be formulated as a non-monotone instance of RS minimization (Problem 5). A similar case is given in [94].

Maximizing Diversity & Minimizing Cooperative Costs: A final set of applications are related to simultaneously maximizing diversity or coverage, while minimizing cooperative costs. Applications of this involve sensor placement, feature selection [74, 54, 87], and data subset selection [80, 129, 133]. While these problems are often modeled as a difference of

submodular functions, or constrained submodular optimization, one can also model them as a ratio of submodular functions, where the submodular function f models the cooperative costs while g models information and utility. The ratio $\frac{g(X)}{f(X)}$ naturally models the cost normalized utility of the set X. Maximizing $\frac{g(X)}{f(X)}$ (Problem 6) leads to the best cost normalized subset X.

3.1.2 Recent studies

There are recent studies after the work was published in 2018. Perrault et al. [102] studied the relationship between DS and Ratio functions and proposed any algorithm that solves RS maximization can solve DS maximization, but in a different definition of guarantee, i.e. find $\hat{X}$, such that $f(\hat{X}) - g(\hat{X}) \geq \gamma f(X) - g(X)$ for all $X \subseteq V$. This is a reverse analysis of section 3.2. They also propose a knapsack $\Psi-$greedy algorithm, that finds $\hat{X}$ such that $\Psi(f(X), g(X)) \geq \Psi((1 - e^{\kappa_g - 1})\, f(X^*), g(X^*))$, assuming f and g are two non-negative, monotone, submodular functions, and $\Psi(x, y)$ is a quasi-convex 2-variables function that is non-decreasing with respect to the first variable. This is a generalization of Theorem 11, since $\frac{x}{y}$ is a quasi-convex function. Wang et al. [127] studied the ratio of monotone non-submodular functions, and find GreedyRatio has guarantee equal $\frac{1}{\gamma_{\emptyset,|X^*|}(g)} \cdot \frac{|X*|}{1+(|X^*|-1)(1-\bar{\kappa}_f)(1-\tilde{\kappa}_f)}$, where $\gamma_{\emptyset,|X^*|}(g)$ is the submodularity ratio [8] of g, $\bar{\kappa}_f = 1 - \min_{S,T \subseteq V, v \in S \setminus T} \frac{f(\{v\}|S\setminus\{v\} \cup T)}{f(\{v\}|S\setminus\{v\})}$, $\tilde{\kappa}_f = 1 - \min_{S,T \subseteq V, v \in S \setminus T} \frac{f(\{v\}|S\setminus\{v\})}{f(\{v\}|S\setminus\{v\} \cup T)}$ and X^* is the optimal solution. The result is $\frac{|X*|}{1+(|X^*|-1)(1-\bar{\kappa}_f)}$ for submodular f and g case.

3.1.3 RoadMap of this chapter

The rest of the chapter is organized as follows. We first describe connections between RS minimization and related problems studied in the literature (Section 3.2). In particular, we show how this is closely related to the problem of minimizing the difference between submodular functions and to the problem of submodular optimization subject to submodular lower bound and upper bound constraints. In Section 3.3, we provide several approximation algorithms along with the analysis of their approximation guarantees for RS minimization. The

Algorithm 4: A $(1+\epsilon)$-approximation algorithm for RS minimization using an exact algorithm for DS minimization

Input: f, g, $\epsilon \in [0,1)$ and an exact algorithm for DS minimization.
Output: A $(1+\epsilon)$-approximate solution for Prob. 5
Set $\lambda_{\max} \leftarrow \frac{f(X)}{g(X)}$ for arbitrary $X \subseteq V$, and $\lambda_{\min} \leftarrow 0$.
while $\lambda_{\max} > (1+\epsilon)\lambda_{\min}$ **do**
  $\bar{\lambda} \leftarrow \frac{\lambda_{\min}+\lambda_{\max}}{2}$.
  $\hat{X} \leftarrow \operatorname{argmin}_{X \supset \emptyset}[f(X) - \bar{\lambda} g(X)]$.
  if $\frac{f(\hat{X})}{g(\hat{X})} \geq \bar{\lambda}$ **then**
    $\lambda_{\min} \leftarrow \bar{\lambda}$
  else
    $\lambda_{\max} \leftarrow \bar{\lambda}$
  end if
end while
Return $\hat{X} \leftarrow \operatorname{argmin}_X[f(X) - \bar{\lambda} g(X)]$.

algorithms include GreedyRatio, Binary Search, Majorization-Minimization, and Ellipsoidal Approximations. In Section 3.4, we prove matching hardness bounds for this Problem. In Section 3.5, we consider extensions of RS minimization where f and g may be supermodular and/or non-monotone submodular. Empirical evaluations on synthetic data are given in Section 3.6.

3.2 Connections to Related Problems

Connections to DS optimization: A problem related to the RS minimization is the Difference of Submodular (DS) minimization [93] defined as follows:

$$\text{Problem: } \min_{X \subseteq V}[f(X) - \lambda g(X)], \tag{3.4}$$

where $\lambda \geq 0$. We call the objective $f(X) - \lambda g(X)$ a difference of submodular (DS) function. We show below that, in fact, an exact algorithm for DS minimization can be used as a subroutine to also solve RS minimization via a simple binary search scheme as described in Alg. 4.

Lemma 20. *Given $\epsilon > 0$ and an exact algorithm for solving DS minimization (Problem 3.4), Algorithm 4 provides a $(1+\epsilon)$-approximation for RS minimization (Problem 5), by solving $O(\log(1/\epsilon))$ instances of DS minimization.*

Proof. When the algorithm terminates, the following holds: $\min_X [f(X) - \lambda_{\min} g(X)] \geq 0$, which implies that $\frac{f(X)}{g(X)} \geq \lambda_{\min}, \forall X \subset V$. Therefore, it holds that: $\frac{f(\hat{X})}{g(\hat{X})} \leq \lambda_{\max} \leq (1+\epsilon)\lambda_{\min} \leq (1+\epsilon)\min_X \frac{f(X)}{g(X)}$. □

While RS minimization and DS minimization are closely related, we show below the class of set functions representable as an RS function is strictly contained by the class of DS functions. Thus, the DS minimization problem encapsulates a strictly larger class of combinatorial optimization problems.

Lemma 21. *Any RS function can be expressed as a DS function. However, there exists an instance of a DS function that cannot be represented as an RS function.*

Proof. The first half of the lemma holds since any RS function is a set function, and any set function can be expressed as a DS function [93].

To show the second half of the Lemma, we give a counter-example as follows: Let $V = \{1, 2\}$, $h(X) = f_1(X) - g_1(X)$ where $f_1(X) = \sqrt{2|X|}$ and $g_1(X) = |X|$. Note that $h(X)$, by definition, is a DS function. Assume that $h(X)$ can be expressed as $\frac{f_2(X)}{g_2(X)}$ with $f_2(X)$ and $g_2(X)$ being non-decreasing submodular functions. We then have that $f_2(\emptyset) = f_2(V) = 0$, since $\frac{f_2(\emptyset)}{g_2(\emptyset)} = \frac{f_2(V)}{g_2(V)} = 0$. However, we have that $f_2(\{1\}) = h(\{1\})g_2(\{1\}) > 0$, which contradicts the monotonicity of f_2. □

In Section 3.3, we give bounded approximation algorithms for RS minimization. This is unlike DS minimization that, in the worst case, is inapproximable [54] — hence, we cannot

simply optimize $\log f/g$ and expect guarantees. Nevertheless, there are a number of heuristic approaches to DS optimization that work well in practice [54, 93, 67]. Moreover, there are several special cases of DS minimization that can be solved exactly (Section 3.3) and hence where a $(1+\epsilon)$-approximation for RS minimization may be obtained via Algorithm 4.

Algorithm 5: Approx. algorithm for RS minimization using an approximation algorithm for SCSC.

Input: f, g, $\epsilon > 0$, and a $[\sigma, \rho]$ bicriterion approximation algorithm for SCSC.
Output: An $\frac{\sigma(1+\epsilon)}{\rho}$ approximation for Problem 5.
$c \leftarrow g(V), \hat{X}_c \leftarrow V$, and $\hat{X} \leftarrow \hat{X}_c$.
while $g(\hat{X}_c) > \min_{j \in V} g(j)$ **do**
    $c \leftarrow (1+\epsilon)^{-1} c$
    $\hat{X}_c \leftarrow [\sigma, \rho]$ approx. for Problem 3.5 with c.
    if $\frac{f(\hat{X})}{g(\hat{X})} > \frac{f(\hat{X}_c)}{g(\hat{X}_c)}$ **then**
        $\hat{X} \leftarrow \hat{X}_c$
    end if
end while
Return $\hat{X}$.

Relation to SCSC and SCSK: Another related and recently studied class of problems is submodular optimization subject to submodular cover and submodular knapsack constraints [55], namely:

$$\text{Problem (SCSC): } \min\{f(X) \,|\, g(X) \geq c\}, \tag{3.5}$$

$$\text{Problem (SCSK): } \max\{g(X) \,|\, f(X) \leq b\}, \tag{3.6}$$

which generalize [136, 1]. These problems are referred to as *Submodular Cost Submodular Cover* (SCSC) and *Submodular Cost Submodular Knapsack* (SCSK), respectively. As shown in [55], both SCSC and SCSK admit bi-criterion approximation algorithms. Without loss of

generality, we concentrate on SCSC. An algorithm is a $[\sigma, \rho]$ bi-criterion algorithm for SCSC if it is guaranteed to obtain a set X such that $f(X) \leq \sigma f(X^*)$ (approximate optimality) and $g(X) \geq \rho c$ (approximate feasibility), where X^* is the optimizer for SCSC. Note that it typically holds that $\sigma \geq 1$ and $\rho \leq 1$. Interestingly, we show in the below that any $[\sigma, \rho]$ bicriterion algorithm for SCSC can be used as a subroutine to yield a $\frac{\sigma}{\rho}$-approximation algorithm for RS minimization via a simple linear search strategy as described in Algorithm 5.

Lemma 22. *Given $\epsilon > 0$, Algorithm 5 is guaranteed to find a solution $\hat{X}$ which is a $\frac{\sigma}{\rho}(1+\epsilon)$-approximation for RS minimization in $O(1/\epsilon)$ calls to a $[\sigma, \rho]$ bi-criterion algorithm for SCSC.*

Proof. Let $X^* = \operatorname{argmin}_X \frac{f(X)}{g(X)}$ and $c^* = g(X^*)$. During the linear search procedure, we must have searched a c such that $c \leq c^* \leq (1+\epsilon)c$. For such c, we have that $f(\hat{X}_c) \leq \sigma f(X^*)$ and $g(\hat{X}_c) \geq \rho c$, thanks to the $[\sigma, \rho]$ bi-criterion guarantee. Therefore, we obtain the following: $\frac{f(\hat{X}_c)}{g(\hat{X}_c)} \leq \frac{\sigma}{\rho}\frac{f(X^*)}{c} \leq \frac{\sigma}{\rho}\frac{(1+\epsilon)f(X^*)}{c^*} \leq \frac{\sigma(1+\epsilon)}{\rho}\frac{f(X^*)}{g(X^*)}$. □

Using the same argument, we may connect SCSK to RS maximization via the same linear search strategy. While Lemma 22, — showing that a bicriterion approximation algorithm for SCSC (or SCSK) can be utilized to solve RS minimization (maximization) with bounded approximation factors — is theoretically interesting, Algorithm 5 may not be practical for large-scale problems, since it involves solving $O(1/\epsilon)$ instances of SCSC. In Section 3.3, we provide a number of more efficient approximation algorithms for RS minimization, while still offering similar guarantees.

3.3 Approximation Algorithms for RS Minimization

In this section, we study four separate cases of RS minimization depending on whether f or g, are modular or submodular.

Algorithm 6: GreedRatio for RS minimization

Input: f and g.

Output: An approximation solution $\hat{X}$.

Initialize: $X_0 \leftarrow \emptyset$, $R \leftarrow V$ and $i \leftarrow 0$.

while $R \neq \emptyset$ **do**

  $v \in \text{argmin}_{v \in R} \frac{f(v|X_i)}{g(v|X_i)}$.

  $X_{i+1} \leftarrow X_i \cup v$.

  $R \leftarrow \{v \in R | g(v|X_{i+1}) > 0\}$

  $i \leftarrow i + 1$.

end while

$i^* \in \text{argmin}_i \frac{f(X_i)}{g(X_i)}$.

Return $\hat{X} \leftarrow X_{i^*}$.

3.3.1 Modular f and Modular g

When both f and g are modular, RS minimization becomes very easy. We introduce a simple greedy algorithm–GreedRatio (Algorithm 6) to handle this scenario. The idea of GreedRatio is to, in each iteration, greedily add an element to the solution set such that the ratio of the marginal gain by this element is the smallest. When the algorithm terminates, a chain of sets $X_1 \subset \ldots, \subset X_\ell$ (ℓ is the total number of iterations) is obtained, and the algorithm simply outputs the set X_{i^*} that achieves the minimum ratio. Though simple, we show below that GreedRatio is guaranteed to yield the optimal solution for RS minimization in this case.

Theorem 9. *When f and g are modular,* GreedRatio *finds the optimal solution for RS minimization with a complexity of $O(n \log n)$.*

Proof. Since both f and g are modular functions, we have that $f(v|X) = f(v)$ and $g(v|X) = g(v)$ for all $X \subseteq V$ and $v \in V \setminus X$. As a simpler implementation of GreedRatio, one may first obtain a non-decreasing order of the items in V as $\sigma = \{v_{\sigma_1}, \ldots, v_{\sigma_n}\}$ according to their

ratio of singleton scores $\frac{f(v)}{g(v)}$. Namely, $\frac{f(v_{\sigma_1})}{g(v_{\sigma_1})} \leq \cdots \leq \frac{f(v_{\sigma_n})}{g(v_{\sigma_n})}$.

For any threshold $\tau \geq 0$, define $X^\tau = \{v \in V | \frac{f(v)}{g(v)} \leq \tau\}$. For any $1 \leq i < j \leq n$, if $v_{\sigma_j} \in X^\tau$, then $v_{\sigma_i} \in X^\tau$, since $\frac{f(v_{\sigma_i})}{g(v_{\sigma_i})} \leq \frac{f(v_{\sigma_j})}{g(v_{\sigma_j})} \leq \tau$. So X^τ is a partial order of $\{v_{\sigma_1}, \ldots, v_{\sigma_n}\}$ and one of the solution set $\{X_i\}_{i=1}^n$.

Let $X^* \in \operatorname{argmin}_X \frac{f(X)}{g(X)}$ and $r^* = \frac{f(X^*)}{g(X^*)}$. Observe that if any item v satisfies $\frac{f(v)}{g(v)} < r^*$, the item must be contained in X^*, otherwise, adding v to X^* would further decrease the objective. Let $X^{r^*} = \left\{v \in V | \frac{f(v)}{g(v)} \leq r^*\right\}$. Note that X^{r^*} is contained in the chain of the solutions $\{X_i\}_{i=1}^n$ and that $\frac{f(X^{r^*})}{g(X^{r^*})} \leq r^*$, but r^* is the optimal value, so $\frac{f(X^{r^*})}{g(X^{r^*})} = r^*$. Therefore, the $\hat{X} \leftarrow \hat{X_{i^*}}$ where $i^* \in \operatorname{argmin}_i \frac{f(X_i)}{g(X_i)}$ as the output of GreedRatio is optimal.

The complexity of algorithm involves computing all n ratio of singleton scores and then takes another $O(n \log n)$ to sort them.

□

3.3.2 Modular f and Submodular g

Next, we study a slightly more general form where f is modular and g is submodular. We show below that this scenario can still be solved up to a constant $1/(1-1/e)$ factor by the same greedy algorithm– GreedRatio.

Theorem 10. *When f is modular and g is submodular,* GreedRatio *is guaranteed to obtain a solution $\hat{X}$ such that*

$$\frac{f(\hat{X})}{g(\hat{X})} \leq \frac{e}{e-1} \frac{f(X^*)}{g(X^*)}, \tag{3.7}$$

where $X^* \in \operatorname{argmin}_{\emptyset \subset X \subseteq V} \frac{f(X)}{g(X)}$.

Proof. This simply follows as a special case of Theorem 11 (cf. Section 3.3.4) when $\kappa_f = 0$. □

We point out that GreedRatio may require a time complexity of $O(n^2)$ function evaluations in the worst case. However, thanks to the lazy evaluation trick as described in [90],

Line 5 in GreedRatio need not to recompute the marginal gain for every item in each round, allowing GreedRatio to scale to large data sets.

3.3.3 Submodular f and Modular g

In contrast to the case above where f is modular and g is submodular, we show that here, the opposite case, can actually be exactly optimized using the binary search strategy in Algorithm 4. The key observation is that the corresponding DS minimization becomes an instance of submodular minimization, which can be optimally solved in poly-time. Hence, one can achieve a $(1+\epsilon)$-approximation for this case as a Corollary of Theorem 20:

Corollary 5. *When f is submodular and g is modular, using an exact submodular minimization algorithm as a subroutine, Algorithm 4 provides a $(1+\epsilon)$-approximation for RS minimization in $O(\log(1/\epsilon))$ calls to the subroutine.*

3.3.4 Submodular f and Submodular g

Lastly, we study the most general form of RS minimization with both f and g being submodular, and GreedRatio can again be shown to yield a curvature dependent bound for this problem. The *curvature* of a submodular function f is defined as follows: [15, 125, 56, 130]:

$$\kappa_f = 1 - \min_{v \in V} \frac{f(v|V \setminus v)}{f(v)} \in [0, 1]. \tag{3.8}$$

The curvature κ_f measures how close a submodular function f is to a modular function. f is fully curved if $\kappa_f = 1$ and is modular if $\kappa_f = 0$. In the below, we show that GreedRatio approximates the RS minimization problem with a factor in terms of the curvature κ_f of the function f.

Theorem 11. *When f, g are normalized monotonic non-decreasing submodular functions,* GreedRatio *is guaranteed to obtain a solution $\hat{X}$ such that*

$$\frac{f(\hat{X})}{g(\hat{X})} \leq \frac{1}{1 - e^{(\kappa_f - 1)}} \frac{f(X^*)}{g(X^*)}, \tag{3.9}$$

where $X^* \in \operatorname{argmin}_{\emptyset \subset X \subseteq V} \frac{f(X)}{g(X)}$ and κ_f is the curvature of the submodular function f.

Proof. It is equivalent to prove the following:

$$\frac{g(\hat{X})}{f(\hat{X})} \geq (1 - e^{(\kappa_f - 1)}) \frac{g(X^*)}{f(X^*)}. \tag{3.10}$$

Denote $X_1, X_2, \ldots, X_l$ as the chain of sets obtained by the greedy algorithm and $x_1, \ldots, x_\ell$ as the sequence of items added during the algorithm. ℓ is the number of rounds when the algorithm terminates. Note that the terminate condition of GreedRatio is when all element $v \in V \setminus X_l$, $g(v|X_l) = 0$, i.e. to avoid divided by 0 when running $\operatorname{argmin}_{i \in V \setminus X_l} \frac{f(v|X_l)}{g(v|X_l)}$. $g(X_l) = g(V) \geq g(X^*)$. So we have $f(X_l) \geq f(X^*)$ since $\frac{f(X_l)}{g(X_l)} \geq \frac{f(X^*)}{g(X^*)}$. Denote k as the largest index in $\{1, \ldots, \ell\}$ such that $f(X_k) \leq f(X^*)$.

For $i = 1, 2, \ldots, k$, it holds that:

$$\begin{aligned} g(X^*) &\leq g(X^* \cup X_{i-1}) \\ &\leq g(X_{i-1}) + \sum_{v \in X^* \setminus X_{i-1}} g(v|X_{i-1}) \\ &\leq g(X_{i-1}) + \sum_{v \in X^* \setminus X_{i-1}} \frac{g(x_i|X_{i-1})}{f(x_i|X_{i-1})} f(v|X_{i-1}) \end{aligned}$$

The last inequality follows since $\frac{g(v|X_{i-1})}{f(v|X_{i-1})} \leq \frac{g(x_i|X_{i-1})}{f(x_i|X_{i-1})}$ as required by the greedy algorithm. Given the definition of the curvature κ_f, we have the following

$$\begin{aligned} g(X^*) - g(X_{i-1}) &\leq \frac{g(x_i|X_{i-1})}{f(x_i|X_{i-1})} \sum_{v \in X^* \setminus X_{i-1}} f(v|X_{i-1}) \\ &\leq \frac{g(x_i|X_{i-1})}{f(x_i|X_{i-1})} \sum_{v \in X^*} f(v) \\ &\leq \frac{g(x_i|X_{i-1})}{f(x_i|X_{i-1})} \frac{1}{1 - \kappa_f} f(X^*) \end{aligned}$$

Rearranging the inequality, we obtain the following:

$$\frac{g(X^*) - g(X_i)}{g(X^*) - g(X_{i-1})} \leq \left[1 - \frac{(1 - \kappa_f) f(x_i|X_{i-1})}{f(X^*)}\right], \tag{3.11}$$

which implies

$$g(X^*) - g(X_k) \leq \prod_{i=1}^{k} \left[1 - \frac{(1 - \kappa_f) f(x_i|X_{i-1})}{f(X^*)}\right] g(X^*) \tag{3.12}$$

Algorithm 7: EllipsoidApprox

1: **Input:** f and g

2: **Output:** An approximation solution $\hat{X}$.

3: $\sqrt{w^f} \leftarrow$ the ellipsoidal approximation for f

4: $\hat{X} \in \operatorname{argmin}_X \frac{\sqrt{w^f(X)}}{g(X)}$ \\ $\frac{e(1+\epsilon)}{e-1}-$Approximately solved by Algorithm 5

5: Return $\hat{X}$

$$\leq \prod_{i=1}^{k} e^{\left[-\frac{(1-\kappa_f)f(x_i|X_{i-1})}{f(X^*)}\right]} g(X^*) \tag{3.13}$$

$$\leq e^{\left[-\frac{(1-\kappa_f)f(X_k)}{f(X^*)}\right]} g(X^*) \tag{3.14}$$

Using the fact $1 - x \leq e^{-x}$, we then obtain the following:

$$\frac{g(X_k)}{f(X_k)} \geq \left[1 - e^{\left[-\frac{(1-\kappa_f)f(X_k)}{f(X^*)}\right]}\right] \frac{f(X^*)}{f(X_k)} \frac{g(X^*)}{f(X^*)} \tag{3.15}$$

$$\geq \left[1 - e^{-(1-\kappa_f)}\right] \frac{g(X^*)}{f(X^*)}. \tag{3.16}$$

Since $(1 - e^{-(1-\kappa_g)x})x^{-1}$ monotonically decreases in x and $\frac{f(X_k)}{f(X^*)} \leq 1$, we have the following:

$$\frac{f(\hat{X})}{g(\hat{X})} \leq \frac{f(X_k)}{g(X_k)} \leq \frac{1}{1 - e^{\kappa_f - 1}} \frac{f(X^*)}{g(X^*)}. \tag{3.17}$$

□

We point out that Theorem 11 generalizes Theorem 10 when f is modular, i.e., $\kappa_f = 0$. Note that the approximation guarantee of GreedRatio deteriorates as the curvature κ_f of the function f increases and becomes unbounded (and hence vacuous) when the f is fully curved, i.e., $\kappa_f = 1$.

Ellipsoid Approximation: To yield a bounded approximation algorithm for RS minimization, we provide an algorithmic framework–EllipsoidApprox, which involves computing the ellipsoidal approximation (EA) of a submodular function.

As shown in Goemans et. al [44], one can approximate any monotone submodular function $f(X)$ in polynomial time by a surrogate function $\hat{f}(X) = \sqrt{w^f(X)}$ for a certain modular weight vector $w^f \in \mathbb{R}^V$, such that

$$\sqrt{w^f(X)} \leq f(X) \leq O(\sqrt{n}\log n)\sqrt{w^f(X)}, \forall X \subseteq V.$$

To apply the idea of EA to RS minimization, we first replace $f(X)$ by its ellipsoidal approximation $\sqrt{w^f(X)}$, and then the problem becomes

$$\min_{\emptyset \subset X \subseteq V} \frac{\sqrt{w^f(X)}}{g(X)}, \tag{3.18}$$

for which we may use Algorithm 5 (i.e., linear search with SCSC) to solve. At every round of Algorithm 5, the SCSC problem has the following form:

$$\min\{\sqrt{w^f(X)} | g(X) \geq c\}, \tag{3.19}$$

which is effectively,

$$\min\{w^f(X) | g(X) \geq c\}. \tag{3.20}$$

Note that Eqn. 3.20 can be solved by the greedy algorithm with a $[1, (1-1/e)]$ bicriterion approximation guarantee [55], which then leads to a constant factor approximation for Eqn. 3.18 thanks to Lemma 22. The following result provides a worst-case approximation factor of this approach for RS minimization:

Theorem 12. *Let $\sqrt{w^f}$ be the ellipsoidal approximation for f and $\hat{X}$ be the output solution of Algorithm 5 on $\min_{\emptyset \subset X \subseteq V} \frac{\sqrt{w^f(X)}}{g(X)}$, it then holds that*

$$\frac{f(\hat{X})}{g(\hat{X})} \leq O(\sqrt{n}\log n)\frac{f(X^*)}{g(X^*)}. \tag{3.21}$$

Proof. Let $X^* \in \operatorname{argmin}_X \frac{f(X)}{g(X)}$. Since the output solution $\hat{X}$ of Algorithm 5 has a constant $\frac{e}{e-1}(1+\epsilon)$-approximation for Eqn. 3.18, it then holds that

$$\frac{f(\hat{X})}{g(\hat{X})} \leq O(\sqrt{n}\log n)\frac{\sqrt{w^f(\hat{X})}}{g(\hat{X})} \tag{3.22}$$

Algorithm 8: MMin for RS minimization

Input: f and g

Output: An approximation solution $\hat{X}$.

Initialize: An arbitrary set X^0, and $t \leftarrow 0$.

repeat

  pick a subgradient h_t at X^t of g

  pick a supergradient m_t at X^t of f

  $A \in \operatorname{argmin}_X \frac{f(X)}{h_t(X)} \backslash\backslash$ $(1+\epsilon)-$Approximately solved by Algorithm 4

  $B \in \operatorname{argmin}_X \frac{m_t(X)}{g(X)} \backslash\backslash$ $\frac{e}{e-1}-$Approximately solved by GreedRatio

  $X^{t+1} \leftarrow \operatorname{argmin}_{X \in \{A,B\}} \frac{f(X)}{g(X)}$

  $t \leftarrow t+1$

until we have converged ($X^t = X^{t-1}$)

Return $\hat{X} \leftarrow X^t$

$$\leq O(\sqrt{n}\log n)\frac{e}{e-1}(1+\epsilon)\frac{\sqrt{w^f(X^*)}}{g(X^*)} \tag{3.23}$$

$$\leq O(\sqrt{n}\log n)\frac{f(X^*)}{g(X^*)} \tag{3.24}$$

□

As we will see, the $O(\sqrt{n}\log n)$-approximation factor provided by this approach matches the lower bound (hardness) of the RS minimization up to a log factor.

Majorization-Minimization: While the Ellipsoidal Approximation algorithm provides the tightest approximation factor, it does not scale very well even to medium scale problems [55]. To this end we propose another framework — Majorization-Minimization (MMin, see Alg. 8) — that achieves a slightly worse approximation factor, but that scales quite well to large scale problems. In the spirit of [60, 134], this framework uses modular lower and modular upper bounds of a submodular function [60] to transform the originally hard RS

minimization problem to a special case with either f or g being modular. For either case, the result admits constant approximation algorithms as shown in Section 3.3.2 and 3.3.3. Moreover, the resulting guarantees can be translated to a curvature dependent guarantee for the original RS minimization.

Akin to convex functions, submodular functions have tight modular lower bounds. These bounds are related to the subdifferential $\partial_f(Y)$ of the submodular set function f at a set $Y \subseteq V$, which is defined [39] as:

$$\partial_f(Y) = \{y \in \mathbb{R}^n : \\ f(X) - y(X) \geq f(Y) - y(Y) \text{ for all } X \subseteq V\} \tag{3.25}$$

For a vector $x \in \mathbb{R}^V$ and $X \subseteq V$, we write $x(X) = \sum_{j \in X} x(j)$. Denote a subgradient at Y by $h_Y \in \partial_f(Y)$. The extreme points of $\partial_f(Y)$ may be computed via a greedy algorithm: Let π be a permutation of V that assigns the elements in Y to the first $|Y|$ positions ($\pi(i) \in Y$ if and only if $i \leq |Y|$). Each such permutation defines a chain with elements $S_0^\pi = \emptyset$, $S_i^\pi = \{\pi(1), \pi(2), \ldots, \pi(i)\}$ and $S_{|Y|}^\pi = Y$. This chain defines an extreme point h_Y^π of $\partial_f(Y)$ with entries

$$h_Y^\pi(\pi(i)) = f(S_i^\pi) - f(S_{i-1}^\pi). \tag{3.26}$$

Defined as above, h_Y^π forms a modular lower bound of f, tight at Y — i.e., $h_Y^\pi(X) = \sum_{j \in X} h_Y^\pi(j) \leq f(X), \forall X \subseteq V$ and $h_Y^\pi(Y) = f(Y)$.

We can also define a modular upper bound of a submodular function f via its superdifferentials $\partial^f(Y)$ [62, 58] at Y:

$$\partial^f(Y) = \{y \in \mathbb{R}^n : \\ f(X) - y(X) \leq f(Y) - y(Y); \text{for all } X \subseteq V\} \tag{3.27}$$

It is possible, moreover, to provide specific supergradients [58, 60] that define the following two modular upper bounds (when referring either one, we use m_X^f):

$$m_{X,1}^f(Y) \triangleq f(X) - \sum_{j \in X \setminus Y} f(j|X \setminus j) + \sum_{j \in Y \setminus X} f(j|\emptyset), \tag{3.28}$$

$$m^f_{X,2}(Y) \triangleq f(X) - \sum_{j \in X \setminus Y} f(j|V \setminus j) + \sum_{j \in Y \setminus X} f(j|X).$$

Then $m^f_{X,1}(Y) \geq f(Y)$ and $m^f_{X,2}(Y) \geq f(Y), \forall Y \subseteq V$ and $m^f_{X,1}(X) = m^f_{X,2}(X) = f(X)$.

Having formally defined the tight modular upper and lower bounds, we are ready to discuss how to apply this machinery to RS minimization. MMin consists of two stages. In the first stage, it replaces f by its modular upper bound and keep g as it is, and then solves the resulting problem using the algorithms proposed in Section 3.3.2. In the second stage, MMin replaces g by its modular lower bound and keep f as it is, and then solves the resulting problem using the algorithms described in Section 3.3.3. Lastly, MMin outputs the better among these two solutions. We show below that MMin yields a bounded approximation for RS minimization in terms of the curvature both of κ_f and κ_g.

Theorem 13. MMin *admits a worst-case approximation factor of* $O(\min\{\frac{n}{1+(n-1)(1-\kappa_f)}, \frac{n}{1+(n-1)(1-\kappa_g)}\})$.

Proof. To prove this theorem, we utilize the Lemma from [56], that stated the following: Given a monotone submodular function f, it holds that

$$f(X) \leq \hat{f}^m(X) \triangleq \sum_{j \in X} f(j) \leq \frac{n}{1 + (n-1)(1-\kappa_f)} f(X) \tag{3.29}$$

where $\hat{f}^m(X)$ is the simple modular upper bound of f. Since, $\hat{f}^m$ approximates f by a factor of $\frac{n}{1+(n-1)(1-\kappa_f)}$, the ratio $\hat{f}^m(X)/g(X)$ also approximates $f(X)/g(X)$ by the same factor. Moreover, the same bound holds for approximating $g(X)$ by its modular lower bound. Hence MMin, produces the two subproblems as discussed in Sections 3.3.2 and 3.3.3. Both subproblems admit constant approximation algorithms. □

Observe that Theorem 13 provides a worst-case approximation for RS minimization that interpolates between the cases when f and g are modular and submodular. In particular, when either f or g are modular, MMin provides a constant factor guarantee for this problem.

3.4 Hardness of RS Optimization

In this section, we provide a worst case hardness result (lower bound) for RS minimization. We show that when f and g are submodular, the problem is NP hard, and one cannot approximate it better than a factor of $O(\sqrt{n})$, which matches the bound of EllipsoidApprox up to a log factor.

Theorem 14. *There exist an instance of submodular function f and g such that no poly-time algorithm can achieve an approximation factor better than $n^{1/2-\epsilon}$, for any $\epsilon > 0$.*

Proof. We prove this result using the hardness construction from [44, 118]. The main idea of the proof technique is to construct two submodular functions $f(X)$ and $f_R(X)$ that with high probability are indistinguishable. Thus, also with high probability, no algorithm can distinguish between the two functions and the gap in their values provides a lower bound on the approximation.

Define two monotone submodular functions $g(X) = \min\{|X|, \alpha\}$ and $f(X) = \min\{\beta + |X \cap \bar{R}|, |X|, \alpha\}$, where $R \subseteq V$ is a random set of cardinality α. Let α and β be an integer such that $\alpha = x\sqrt{n}/5$ and $\beta = x^2/5$ for an $x^2 = \omega(\log n)$. Given an arbitrary $\epsilon > 0$, set $x^2 = n^{2\epsilon} = \omega(\log n)$. Then the ratio between $g(R)$ and $f(R)$ is $n^{1/2-\epsilon}$. A Chernoff bound analysis very similar to [118] reveals that any algorithm that uses a polynomial number of queries can distinguish f and g with probability only $n^{-\omega(1)}$, and therefore cannot reliably distinguish the functions with a polynomial number of queries. Since no algorithm can distinguish between f and g, any algorithm will achieve a minimum value of 1 for the following optimization problem: $\min_X \frac{f(X)}{g(X)}$, whereas the optimal solution has a value $1/n^{1/2-\epsilon}$. □

3.5 Non-Monotone Submodular and Supermodular f and g

In this section, we investigate extensions of RS minimization to the case when f and g are non-monotone submodular, and even supermodular.

3.5.1 Non-Monotone Submodular f and g

Given a modular function g and a non-monotone submodular function f, one can use the Binary Search algorithm (Algorithm 4), in which case the corresponding DS subproblem becomes an instance of submodular minimization. Correspondingly, one can easily extend Theorem 20 to this case, and achieve a $1+\epsilon$ approximation factor for this problem. We can also extend our algorithms to the scenario when one of the functions is monotone submodular, while the other one is non-monotone. For example, if f is monotone submodular, while g is non-monotone, one can use Ellipsoidal Approximation on f, and keep g as it is. The problem then becomes, $\min_X \frac{\sqrt{w_f(X)}}{g(X)}$ which is equivalent to $\max_X \frac{g(X)}{\sqrt{w_f(X)}}$. We can then convert this to SCSK, $\max\{g(X)|\sqrt{w_f(X)} \leq b\}$, which is an instance of non-monotone submodular knapsack, which also has constant factor guarantees [33]. Furthermore, if f is non-monotone, while g is monotone, one can replace g by its modular upper bound, thereby obtaining an instance of an RS optimization problem, with a non-monotone f and a modular g, which can be solved by using the Binary search algorithm (Algorithm 4) as discussed at the beginning of this section. Finally, in case both f and g are non-monotone submodular (a generalization of [94]), one can use Algorithm 4 and repeatedly solve the resulting DS optimization problem. While this has no guarantees, this is a reasonable heuristic for this problem.

3.5.2 Extensions to Supermodular f and g

Consider an instance of Problem 1, when f is modular or submodular, but g is supermodular. One can then use Algorithm 4, and the corresponding DS optimization subproblem becomes an instance of submodular minimization, which can exactly solved. Hence in this case Problem 1 is solvable in polynomial time. One can also consider an alternate case, when f is supermodular, and g is either modular or submodular. In this case, the resulting DS optimization problem (using Algorithm 4) becomes an instance of submodular maximization, which can be approximately solved. While this does not directly correspond to an approxi-

mation guarantee for the original problem, it does provide a reasonable heuristic for solving the problem. When both f and g are monotone supermodular, it remains an open problem whether it admits any polynomial time algorithm with bounded approximations.

3.6 Experiments

We empirically evaluate our proposed algorithmic frameworks for RS minimization, including in particular MMin, GreedRatio, and EllipsoidApprox, on a synthetic data set. In particular, we evaluate on a generalized form of the F-measure function:

$$F_\lambda(X) = \frac{|\Gamma(X) \cap T|}{\lambda|T| + (1-\lambda)|\Gamma(X)|}, \tag{3.30}$$

where $0 \leq \lambda \leq 1$ is a parameter that determines a trade-off weight between precision and recall. Note $F_{\lambda=0.5}$ is the same as the F-measure function defined in Eqn. 3.1. We instantiate the F-measure function on a randomly generated bipartite graph $G(U, W, E)$. The bipartite graph is defined with $|U| = 100$ and $|W| = 100$. We define an edge between $u \in U$ and $w \in W$ independently with probability $p = 0.05$. The set of targets $T \subseteq W$ is also randomly chosen with a fixed size 20, i.e., $|T| = 20$. We run the experiments on 10 instances of the randomly and independently generated data, and we report the averaged results.

As a baseline, we also implement a random sampling method, where we randomly choose 100 subsets $X \subseteq U$ with size $|X| = 50$ and report their averaged function valuation in terms of F_λ and their standard deviation. In Figure 3.6.1, we compare the performance of all methods on with the varying λ. We observe that GreedRatio, though very efficient, achieves consistently the best performance for all cases of λ among all optimization algorithms. Comparable performance is achieved by MMin and EllipsoidApprox, although EllipsoidApprox is much more computationally intensive.

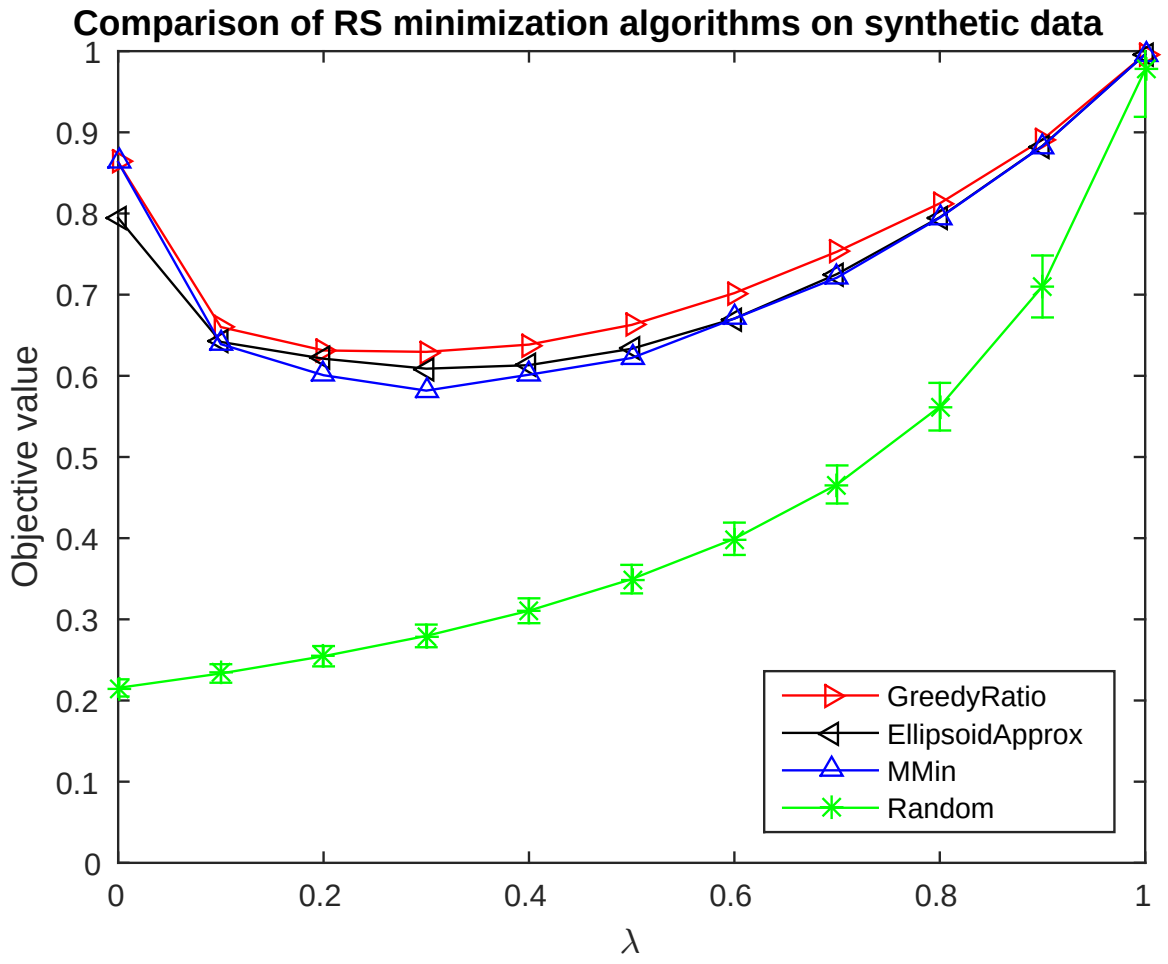


Figure 3.6.1: Synthetic data experiment

Chapter 4

OPTIMIZING MEANS OF SUBMODULAR FUNCTIONS

We study the problem of optimizing various means of submodular functions, including the geometric mean, the generalized mean (often referred to as the Hölder mean), and the harmonic mean. The three mean objectives essentially involve non-trivial combinations of submodular functions, including the product, the p-norm, and the ratio between the product and the sum. We reduce the mean objectives to one of the two problems, namely the non-negative linear combination $f(S)+\lambda g(S)$ and the robust mean combination $\min(f(S), g(S))$, depending on the specific mean formulation and whether we aim to maximize or minimize the mean value. We prove γ based approximation guarantees for the mean objectives, where γ is the approximation guarantee that we can get for solving the reduced problem (e.g., $\gamma = 1 - 1/e$ for the non-negative linear combination problem using the greedy algorithm under a cardinality constraint). Optimizing the mean objectives allows us to develop more complicated objectives of multiple submodular functions, which greatly expands the applications of submodularity. This is from an unpublished paper by

Wenruo Bai, *Shengjie Wang, Tianyi Zhou, and Jeff Bilmes, Optimizing Means of Submodular Functions.*

4.1 Introduction

In this chapter, we study the optimization of more general combinations of two non-negative submodular functions ($f(S) \geq 0 \; \forall S \subseteq V$) involving various means. The most basic mean is the arithmetic mean, i.e., $\frac{f(S)+g(S)}{2}$, which is still submodular as stated above and therefore can be solved using any algorithm that optimizes a single submodular function. The robust mean $\min(f(S), g(S))$ is no longer submodular, and Orlin et al. [98] gives a modified greedy

algorithm that can achieve a constant guarantee for the maximization case. We extend those two means to other widely-used mean metrics. To our best knowledge, we are the first to study the following means of submodular functions, which provide more flexibility in modeling the trade-offs of submodular function objectives:

- **Geometric mean**, $\sqrt{f(S) \cdot g(S)}$. From the perspective of optimization, optimizing the geometric mean is equivalent to optimizing the product of two submodular functions $f(S) \cdot g(S)$, which we denote as the unnormalized form.

- **Generalized mean**, or referred as the Hölder mean, $\left[\frac{[f(S)]^p+[g(S)]^p}{2}\right]^{1/p}$ with $p > 1$. The unnormalized form is the sum of the p-th power of each of the functions: $[f(S)]^p + [g(S)]^p$.

- **Harmonic mean**, $\frac{2}{1/f(S)+1/g(S)} = \frac{2 \cdot f(S) \cdot g(S)}{f(S)+g(S)}$. We remove the 2 factor in the unnormalized form: $\frac{f(S) \cdot g(S)}{f(S)+g(S)}$.

For a clean presentation, we will focus on the unnormalized versions of the mean objectives for the rest of the chapter. In other words, our contribution covers several non-trivial combinations of submodular functions, including the product of two submodular functions, the sum of p-th power of two submodular functions, and the ratio between the product and the sum of two submodular functions. All those combinations of submodular functions do not preserve submodularity.

We summarize our theoretical results in Table 4.1.1, where we show theoretical guarantees for both minimization and maximization problems. Our main idea to optimize those mean combinations rely on reductions to the arithmetic mean and robust mean problems. Therefore the guarantees all depend on the γ factor, which is the approximation guarantee we get when solving the reduced problem. Note that for the cases of geometric mean and generalized mean, the power on the γ factor gets normalized to one if we consider the normalized version of the problem, e.g., apply a $1/p$-th power for the generalized mean. Thus, our

results achieve γ approximations in all cases in terms of the normalized problem. Our results can be extended to combinations of m submodular functions with the same approximation guarantee at the cost with a longer running time that is exponential in m (more details in Section 4.5.1). We will focus on the $m = 2$ case for most of the chapter, and the extension will be straightforward based on the $m = 2$ result.

4.2 Applications

Submodular functions have been broadly applied in areas of machine learning, operations research and economics [140, 48, 108, 103, 74, 105, 122, 111, 110, 131, 82, 141]. The more general combinations of submodular functions studied in this chapter further expand potential applications of submodularity.

More flexible trade-offs between submodular functions. When maximizing various combinations of two submodular functions, we aim to identify diverse subsets that trade-off between qualities measured by the two functions. The robust objective focuses on finding subsets with large values in both functions, which acts like an "and" operation. The geometric mean shares a similar spirit, but we allow more flexible trade-offs as a subset may have a smaller value for one function but much larger value for the other to have an overall large objective value, something not measured (and thus not encouraged) using only $\min(f(A), g(A))$. On the other extreme, we may consider another aggregation which takes the maximum over two submodular function $\max(f(A), g(A))$. In such a case, we find subsets that are diverse based on either function, which is like an "or" operation. We note that when $f(V) = g(V) = 1$, another form of "or" operation is $h(A) = f(A) + g(A) - f(A)g(A)$ which preserves submodularity when both f and g are monotone non-decreasing submodular [46]. The generalized norm can achieve the similar effect, as $(f(A)^p + g(A)^p)^{1/p}$ converges to the max operation as $p \to \infty$. Also note that when $p \to 1$, we recover the arithmetic mean, which trades off the two functions on the linear scale.

A Chain of submodular point processes. A submodular point process [59] models the probability distribution of a subset of items based on a submodular function: $\Pr(A) \propto$

Maximization $\max_{S\subseteq V, S\in\mathbb{C}}[\cdot]$	Geometric Mean $f(S)\cdot g(S)$	Generalized Mean $[f(S)]^p + [g(S)]^p$	Harmonic Mean $\frac{f(S)\cdot g(S)}{f(S)+g(S)}$
Reduced Problem	$\max \min(f(S), \lambda g(S))$	$\max f(S) + \lambda g(S)$	$\max \min(f(S), \lambda g(S))$
Guarantee	$\frac{\gamma^2}{(1+\epsilon)^{3/2}}$	$\frac{\gamma^p}{(1+\epsilon)^p}$	$\frac{\gamma}{1+\epsilon}$
Complexity	$Tp\log_{1+\epsilon}\left[\frac{\max_S f(S)\cdot\max_S g(S)}{\min_{S,f(S)>0} f(S)\cdot\min_{S,g(S)>0} g(S)}\right]$, $p=1$ for geometric and harmonic mean.		
Monotone Comp.	$T\frac{4\log\left(\frac{2}{1-1/e}\right)}{\log(1+\epsilon)}$	$T\frac{\log\frac{4e^2}{(e-1)^2}}{\log(1+\epsilon)}$	$T\frac{-\log\epsilon}{\log(1+\epsilon)}$
m Complexity	$TO\left(\frac{1}{(\log(1+\epsilon))^{m-1}}\right)$	$TO\left(\frac{1}{(\log(1+\epsilon))^{m-1}}\right)$	$TO\left(\frac{(-\log\epsilon)^{m-1}}{(\log(1+\epsilon))^{m-1}}\right)$

Minimization $\min_{S\subseteq V, S\in\mathbb{C}}[\cdot]$	Geometric Mean $f(S)\cdot g(S)$	Generalized Mean $[f(S)]^p + [g(S)]^p$	Harmonic Mean $\frac{f(S)\cdot g(S)}{f(S)+g(S)}$
Reduced Problem	$\min f(S) + \lambda g(S)$	$\min \max(f(S), \lambda g(S))$	$\min f(S) + \lambda g(S)$
Guarantee	$(1+\epsilon+\epsilon^2/16)\cdot\gamma^2$	$(1+\epsilon)^p\cdot\gamma^p$	$(1+\epsilon)\gamma$
Complexity	$Tp\log_{1+\epsilon}\left[\frac{\max_S f(S)\cdot\max_S g(S)}{\min_{S,f(S)>0} f(S)\cdot\min_{S,g(S)>0} g(S)}\right]$, $p=1$ for geometric and harmonic mean.		
m Complexity	$TO\left(\frac{p}{(\log(1+\epsilon))^{m-1}}\right)$		

Table 4.1.1: Summary of our theoretical results. $\mathbb{C}$ is some general set constraint, e.g., cardinality or matroid constraint. Whether we can solve the constrained version of our mean objectives depends on whether we can solve the reduced problem under the same constraint. γ is the approximation guarantee for solving the reduced problem. T is the running time for solving one instance of the reduced problem. The "Monotone Comp." row of the maximization case shows the improved complexity when f and g are monotone. The "m Complexity" rows show the time complexity if we extend the problem to optimize the means of m submodular functions instead of only two.

$f(A)$ where $f(\cdot)$ is a submodular function. The diminishing-returns property naturally models the notion of diversity and cooperation in the probability distribution. By taking the product of submodular functions, we can construct a probability distribution based on a chain. Our results on optimizing the geometric mean of submodular functions enable us to do MAP inference on such a chain of probability distributions.

Submodular-supermodular Optimization. The submodular-supermodular optimization [3] optimizes the objective $f(A) + r(A)$ where $r(\cdot)$ is a supermodular function. By optimizing such an objective, we can identify subsets that exhibit both diversity and complementarity. For example, we can use such a formulation to summarize a large training dataset with limited samples, which on the one hand, are dispersed in space (diversity), and on the other hand, can maintain information about classification boundaries (complementarity). We can treat our generalized mean objective as a special case of the submodular-supermodular optimization by considering the objective $\left[f(A)^{1/p}\right]^p + [m(A)]^p = f(A) + [m(A)]^p$. Given $p > 1$, and $f(\cdot)$ monotone submodular, $f(A)^{1/p}$ is still submodular since $(\cdot)^{1/p}$ is a concave function. $m(\cdot)$ is a non-negative modular function, and $[m(A)]^p$ is thus supermodular. By restricting the supermodular part to the p-th power of a modular function, we can solve the submodular-supermodular optimization problem with a constant guarantee (taking p as a constant). Note that in [3], the proven approximation guarantees are based on the curvatures of the functions, and vary according to the function values. Even for the special case of $[m(A)]^p$ as the supermodular function, it's easy to construct the modular function so that the function is nearly fully-curved and the approximation guarantees become vacuous. On the contrary, our results show constant bounds irrespective of the function curvatures.

F-measure. F-measure or F_1 score captures the trade-offs between the precision and recall for a classification model. Precisely, $F_1 = \frac{1}{\text{recall}^{-1}+\text{precision}^{-1}}$, which is the harmonic mean between the precision and the recall. Note that recall is essentially a modular function (and thus submodular) and precision is a submodular function. Consider a binary classification task and let the predicted positive samples be $\hat{A}$ and the true positive samples as A^+. Then $\text{recall}(\hat{A}) = \frac{|\hat{A} \cap A^+|}{|A^+|}$, which is modular as A^+ is fixed given the dataset. precision =

$\frac{|\hat{A} \cap A^+|}{|\hat{A}|}$ is submodular as the gain of the smaller set is larger as $\frac{1}{|\hat{A}|} - \frac{1}{|\hat{A}|+1}$ gets smaller as $|\hat{A}|$ gets larger. Therefore, F-measure is a harmonic mean of two submodular functions. For applications such as document summarization, we can identify a subset of words/sentences that directly optimize the F-measure score with respect to the ground truth summaries.

4.3 Related Work

Optimizations of a single submodular function have been extensively studied. For maximizing monotone submodular functions, algorithms with constant theoretical guarantees have been developed for various constraints including cardinality constraint [95], knapsack constraint [73, 78], intersection of matroid constraints [38, 124], and p-system constraint [91, 2]. For maximizing non-monotone submodular functions, Buchbinder et al. [12], Buchbinder and Feldman [11] give constant approximations for the unconstrained case, Buchbinder et al. [13], Kuhnle [77] for the cardinality constraint case, and Vondrák [126], Gharan and Vondrák [43], Feldman et al. [34] for the matroid constrained case. Submodular minimization, on the other hand, can be solved exactly for the unconstrained case [45, 40, 64], while it is information theoretically hard to obtain a constant approximation for the constrained case [118]. In this chapter, for the cases where the reduced problem is the linear combination of submodular functions, i.e., $f(A) + \lambda g(A)$, we can utilize any of the above results to solve the reduced problem and the γ approximation factor carries over to our final approximation guarantee.

Several forms of combinations of submodular functions have also been studied. For the robust mean objective of two monotone submodular functions under a cardinality constraint, Orlin et al. [98] proposes a generalized greedy algorithm that gives a constant approximation, and Krause et al. [75] uses a binary search procedure, which gives a bi-criteria guarantee. Bai et al. [5] studies the ratio of two submodular functions and provides guarantees based on the curvatures of the two functions. Bai and Bilmes [3] introduces the problem of submodular-supermodular optimization, and also gives curvature based guarantees. As discussed above, we can use the generalized mean objective to construct a special case of the submodular-supermodular optimization, and our methods give constant approximations.

Also note that supermodular optimization is hard in general, but our results cover some special cases of supermodular optimizations that constant approximations are obtainable.

Algorithm 9: Algorithm for $\min_S h(S)$; h can be $f \cdot g$, $f^p + g^p$ or $\frac{f \cdot g}{f+g}$

Input: $V, \lambda_{\min}, \lambda_{\max}, \epsilon$

$\lambda \leftarrow \lambda_{\min}$, $\hat{S} \leftarrow \emptyset$

while $\lambda \leq \lambda_{\max}$ **do**

find S_λ by **minimizing** reduced function of $h(S)$ parameterized by λ.

if $h(S_\lambda) < h(\hat{S})$ **then**

$\hat{S} \leftarrow S_\lambda$

end if

$\lambda \leftarrow (1+\epsilon)\lambda$

end while

Algorithm 10: Algorithm for $\max_S h(S)$; h can be $f \cdot g$, $f^p + g^p$ or $\frac{f \cdot g}{f+g}$

Input: $V, \lambda_{\min}, \lambda_{\max}, \epsilon$

$\lambda \leftarrow \lambda_{\min}$, $\hat{S} \leftarrow \emptyset$

while $\lambda \leq \lambda_{\max}$ **do**

find S_λ by **maximizing** the reduced function of $h(S)$ parameterized by λ.

if $h(S_\lambda) > h(\hat{S})$ **then**

$\hat{S} \leftarrow S_\lambda$

end if

$\lambda \leftarrow (1+\epsilon)\lambda$

end while

4.4 Main Results: Optimizing Means of Submodular Functions

We optimize the three mean objectives using a similar approach, where we reduce the original mean objective into one of the two reduced problems: 1) the arithmetic mean like combination $f(S) + \lambda g(S)$ with $\lambda \geq 0$, and 2) the robust mean like combination $\min(f(S), \lambda g(S))$. Our algorithm essentially finds a λ value such that the optimal solution of the original mean

objective and the reduced problem coincides. To find such a λ value, we identify the lower bound (λ_{min}) and upper bound (λ_{max}) for λ, create an exponentially-spaced sequence between the two bounds, and search for the optimal λ. Taking the geometric mean objective as an example, the reduced problem is $\min f(S) + \lambda g(S)$ for the unconstrained minimization case. As shown in Alg. 10, we start with $\lambda \leftarrow \lambda_{min}$ (details about finding the λ bounds later), pick an ϵ spacing parameter, which trades off the running time and the spacing density, solve the reduced problem $\min f(S) + \lambda g(S)$ using any out-of-the-box algorithm (e.g., Fujishige-Wolfe min-norm), record the solution if the original mean objective is better, and increase λ by a factor of $1 + \epsilon$.

4.4.1 Geometric Mean

We show the detailed proof of the following theorem on the maximization of the geometric mean, which demonstrates the main idea of our reduction. All other cases will follow a similar spirit, and the details of those cases will be given in Appendix.

Theorem 15. *If there is any algorithm that solves* $\max_{S,S\in\mathbb{C}} \min(f(S), \lambda g(S))$ *with a guarantee* γ *in* T *oracle calls for function values of* f *or* g *for any* $\lambda \geq 0$*, Algorithm 10 can get a solution* $\hat{S}$ *in* $T \log_{1+\epsilon}\left[\frac{\max_S f(S)\cdot\max_S g(S)}{\min_{S,f(S)>0} f(S)\cdot\min_{S,g(S)>0} g(S)}\right]$ *oracle calls with the guarantee* $f(\hat{S}) \cdot g(\hat{S}) \geq \frac{\gamma^2}{(1+\epsilon)^{3/2}} \max_{S\subseteq V, S\in\mathbb{C}} f(S) \cdot g(S)$.

Proof. Let $S^* \in \operatorname{argmax}_{S\subseteq V,S\in\mathbb{C}} f(S) \cdot g(S)$, and let $\lambda^* = {}^{f(S^*)}/_{g(S^*)}$. Algorithm 10 searches for λ in an exponentially-spaced sequence: $\lambda_0, (1+\epsilon)\lambda_0, (1+\epsilon)^2\lambda_0, \ldots, (1+\epsilon)^N\lambda_0$ plus two boundary cases $\lambda = 0, \infty$. If we choose $\lambda_0 = \frac{\min_{v\in V, f(v)>0} f(v)}{\max_S g(S)}$ and $N \geq \log_{1+\epsilon} \frac{\max_S f(S)}{\lambda_0 \min_{v\in V, g(v)>0} g(v)}$, we can always find $\hat{\lambda}$ such that $\frac{\lambda^*}{\sqrt{1+\epsilon}} \leq \hat{\lambda} \leq \sqrt{1+\epsilon}\cdot\lambda^*$. Note that $\max_S f(S)$ can be estimated using any non-monotone submodular optimization algorithm. Let $\hat{S}$ be the output by $\hat{\lambda}$, and we have:

$$f(\hat{S}) \cdot g(\hat{S}) \geq \frac{1}{\hat{\lambda}} \left[\min\left(f(\hat{S}), \hat{\lambda} g(\hat{S})\right)\right]^2 \tag{4.1}$$

$$\geq \frac{\gamma^2}{\hat{\lambda}} \left[\min\left(f(S^*), \hat{\lambda} g(S^*)\right)\right]^2 \tag{4.2}$$

$$\geq \frac{\gamma^2}{\sqrt{1+\epsilon}\cdot\lambda^*} \left[\min\left(f(S^*), \left(\frac{\lambda^*}{\sqrt{1+\epsilon}}\right) g(S^*)\right)\right]^2 \tag{4.3}$$

$$\geq \frac{\gamma^2}{\sqrt{1+\epsilon}\cdot\lambda^*} \left[\min\left(f(S^*), \frac{f(S^*)}{\sqrt{1+\epsilon}}\right)\right]^2 \tag{4.4}$$

$$\geq \frac{\gamma^2}{(1+\epsilon)^{3/2}\cdot\lambda^*} \left[f(S^*)\right]^2 \tag{4.5}$$

$$= \frac{\gamma^2}{(1+\epsilon)^{3/2}} \cdot f(S^*)\cdot g(S^*) \tag{4.6}$$

Eq. 4.1 follows the property of the min operator. Such a step is the key for all of our reductions. In this case, we utilize a mathematical inequality that connects the product of two functions to the minimum over the two. For the cases of minimization or other mean objectives, we connect our target to the reduced problem using other inequalities, for example the Hölder's inequality. Eq. 4.2 is a result of the γ approximation of the reduced problem; note that S^* is not necessarily an optimal solution of the reduced problem, but the inequality still holds. Eq. 4.3 comes from $\frac{\lambda^*}{\sqrt{1+\epsilon}} \leq \hat{\lambda} \leq \sqrt{1+\epsilon}\cdot\lambda^*$. Eq. 4.4 and Eq. 4.6 are based on $\lambda^* = {}^{f(S^*)}/_{g(S^*)}$. □

□

Corollary 6. *For monotone submodular functions f and g, Under a k cardinality constraint, i.e. $|S| \leq k$, Algorithm 10 is guaranteed to find a solution $\hat{S}$ such that $f(\hat{S})\cdot g(\hat{S}) \geq \frac{(1-1/e-\epsilon)^2}{(1+\epsilon)^{3/2}} \max_{S\subseteq V, |S|\leq k} f(S)\cdot g(S)$.*

For maximizing the mean objective of monotone submodular functions, we can utilize better estimates of the λ_{max} and λ_{min} values to achieve better running time (presented in Table 4.1.1 row "Monotone Comp.") as discussed in Section 4.6.

Theorem 16. *If there is any algorithm that solves $\min_{S,S\in\mathbb{C}} f(S) + \lambda g(S)$ with guarantee γ in T oracle calls of function value of f or g for any λ, algorithm 2 can get a solution $\hat{S}$ in $T\log_{1+\epsilon}\left[\frac{\max_S f(S)\cdot\max_S g(S)}{\min_{S,f(S)>0} f(S)\cdot\min_{S,g(S)>0} g(S)}\right]$ oracle calls with the guarantee $f(\hat{S})\cdot g(\hat{S}) \leq (1+\epsilon+{}^{\epsilon^2}/_{16})\,\gamma^2 \min_{S\subseteq V, S\in\mathbb{C}} f(S)\cdot g(S)$*

Corollary 7. *For unconstrained cases i.e. $S \subseteq V$, Algorithm 2 is guaranteed to find a solution $\hat{S}$ such that $f(\hat{S}) \cdot g(\hat{S}) \leq (1 + \epsilon + \epsilon^2/16)(1 + \epsilon)^2 \min_{S \subseteq V} f(S) \cdot g(S)$.*

4.4.2 Generalized Mean

Theorem 17. *If there is any algorithm that solves $\max_{S, S \in \mathbb{C}} f(S) + \lambda g(S)$ with a guarantee γ in T oracle calls for function values of f or g for any $\lambda \geq 0$, Algorithm 10 can get a solution $\hat{S}$ in $Tp \log_{1+\epsilon} \left[\frac{\max_S f(S) \cdot \max_S g(S)}{\min_{S, f(S)>0} f(S) \cdot \min_{S, g(S)>0} g(S)} \right]$ oracle calls with the guarantee $f^p(\hat{S}) + g^p(\hat{S}) \geq \frac{\gamma^p}{(1+\epsilon)^p} \max_{S \subseteq V, S \in \mathbb{C}} f^p(S) + g^p(S)$.*

Corollary 8. *For monotone submodular functions f and g, under a k cardinality constraint, i.e. $|S| \leq k$, Algorithm 10 is guaranteed to find a solution $\hat{S}$ such that $f^p(\hat{S}) + g^p(\hat{S}) \geq \left(\frac{1 - 1/e}{1+\epsilon} \right)^p \max_{S \subseteq V, |S| \leq k} f^p(S) + g^p(S)$.*

Theorem 18. *If there is any algorithm that solves $\min_{S, S \in \mathbb{C}} \max(f(S), \lambda g(S))$ with guarantee γ in T oracle calls of function value of f or g for any λ, Algorithm 9 can get a solution $\hat{S}$ in in $Tp \log_{1+\epsilon} \left[\frac{\max_S f(S) \cdot \max_S g(S)}{\min_{S, f(S)>0} f(S) \cdot \min_{S, g(S)>0} g(S)} \right]$ oracle calls with the guarantee $f^p(\hat{S}) + g^p(\hat{S}) \leq (1 + \epsilon)^p \gamma^p \min_{S \subseteq V, S \in \mathbb{C}} f^p(S) + g^p(S)$.*

4.4.3 Harmonic Mean

Theorem 19. *If there is any algorithm that solves $\max_{S, S \in \mathbb{C}} \min(f(S), \lambda g(S))$ with a guarantee γ in T oracle calls for function values of f or g for any $\lambda \geq 0$, Algorithm 10 can get a solution $\hat{S}$ in $T \log_{1+\epsilon} \left[\frac{\max_S f(S) \cdot \max_S g(S)}{\min_{S, f(S)>0} f(S) \cdot \min_{S, g(S)>0} g(S)} \right]$ oracle calls with the guarantee $\frac{f(\hat{S}) \cdot g(\hat{S})}{f(\hat{S}) + g(\hat{S})} \geq \frac{\gamma}{(1+\epsilon)} \max_{S \subseteq V, S \in \mathbb{C}} \frac{f(S) \cdot g(S)}{f(S) + g(S)}$.*

Corollary 9. *For monotone submodular functions f and g, Under a k cardinality constraint, i.e. $|S| \leq k$, Algorithm 10 is guaranteed to find a solution $\hat{S}$ such that $\frac{f(\hat{S}) \cdot g(\hat{S})}{f(\hat{S}) + g(\hat{S})} \geq \frac{1 - 1/e}{(1+\epsilon)} \max_{S \subseteq V, |S| \leq k} \frac{f(S) \cdot g(S)}{f(S) + g(S)}$.*

Theorem 20. *If there is any algorithm that solves $\min_{S, S \in \mathbb{C}} f(S) + \lambda g(S)$ with guarantee γ in T oracle calls of function value of f or g for any λ, Algorithm 9 can get a solution*

$\hat{S}$ in in $T \log_{1+\epsilon}\left[\frac{\max_S f(S)\cdot\max_S g(S)}{\min_{S,f(S)>0} f(S)\cdot\min_{S,g(S)>0} g(S)}\right]$ oracle calls with the guarantee $\frac{f(\hat{S})\cdot g(\hat{S})}{f(\hat{S})+g(\hat{S})} \leq (1+\epsilon)\gamma \max_{S\subseteq V, S\in\mathbb{C}} \frac{f(S)\cdot g(S)}{f(S)+g(S)}$.

Corollary 10. *For unconstrained cases i.e. $S \subseteq V$, Algorithm 9 is guaranteed to find a solution $\hat{S}$ such that $\frac{f(\hat{S})\cdot g(\hat{S})}{f(\hat{S})+g(\hat{S})} \leq (1+\epsilon) \max_{S\subseteq V, S\in\mathbb{C}} \frac{f(S)\cdot g(S)}{f(S)+g(S)}$.*

4.5 Generalization to m Submodular Functions

4.5.1 Generalization to m Submodular Functions with Multi-linear Search

In previous sections, we have discussed the geometric/generalized/harmonic mean of two submodular functions. The proposed algorithm can be generalized to m submodular functions with the same guarantee at the costs of longer running time (still polynomial in ϵ and n but exponential in m), so the method is only scalable if m is small enough and treated as a constant.

Geometric mean (product of m submodular functions) We can extend our algorithm to optimize a reduced function $\lambda_1 f_1(S) + \lambda_2 f_2(S) + \ldots + \lambda_m f_m(S)$ for the minimization problem and $\min(\lambda_1 f_1(S), \lambda_2 f_2(S), \ldots, \lambda_m f_m(S))$ for the maximization problem respectively. The key step to prove the guarantee follows the inequality:

$$\min_i(\lambda_i f_i(S)) \leq \left(\prod_i \lambda_i f_i(S)\right)^{1/m} \leq \frac{\sum_i \lambda_i f_i(S)}{n}. \tag{4.7}$$

Assuming we know ${}^{f_i(S^*)}/_{f_1(S^*)}$, $\forall i \geq 2$, where S^* is the optimizer of the original optimization problem, we can solve the problem by setting $\lambda_1 = 1$ and $\lambda_i = {}^{f_i(S^*)}/_{f_1(S^*)}$, $\forall i \geq 2$. The optimizer of the reduced problem is also the optimizer of the original problem with the **same approximation guarantees**. Note that we assume we know the λ_i's, but we can guess the values by multi-linear search over $m-1$ dimensions (note $\lambda_1 = 1$ is fixed) with a bounded multiplicative error $1+\epsilon$. Running time is exponential in m, but polynomial on n and $\frac{1}{\log(1+\epsilon)}$.

Generalized mean (p-norm) of m submodular functions. We apply the same idea as the geometric mean case. The reduced problems are $\max_i \lambda_i f_i(S)$ for minimization and

$\sum_i \lambda_i f_i(S)$ for maximization. The key inequalities are:

$$\left(\sum_i f_i^p(S)/m\right)^{1/p} \leq \max_i \lambda_i f_i(S), \text{ and} \tag{4.8}$$

$$\sum_i \lambda_i f_i(S) \leq \left(\sum_i f_i^p(S)\right)^{1/p} \left(\sum_i \lambda_i^q\right)^{1/q}, \tag{4.9}$$

where $1/p + 1/q = 1$, according to Hölder's inequality. Then we use the same multi-linear search algorithms for λ_i's and again we can find λ_i close to $f_i^{p-1}(S^*)/f_1^{p-1}(S^*)$ with at most $1+\epsilon$ multiplicative error. Finally, we get the same approximation guarantee of the original problem (p-norm) but with an increase of running time that is exponential in m.

Harmonic mean of m submodular functions. The same idea applies here, and note that the harmonic mean can be seen as a generalized p-norm with $p = -1$. Hölder's inequality can also be generalized in the case where $p < 0$, but with a reversed direction of inequality. The multi-linear search algorithm can therefore solve the optimization problem by solving the reduced problems with proper λ_i's.

4.5.2 Generalization to Quadratic Forms of m Submodular Functions

A quadratic form is a polynomial with terms all of degree two, e.g. $\sum_{i,j} a_{ij} f_i(S) f_j(S)$. This is a natural generalization of our main results, which cover the optimization of $f^2 + g^2$ and $f \cdot g$. Without loss of generality, we assume a_{ij} is symmetric, i.e., $a_{ij} = a_{ji}$. In general, we can not approximate the problem to any factor using a polynomial time algorithm for either the maximization or the minimization case even with a constant number of functions. Consider the following example:

$$f(S) - g(S) = (f^{1/2}(S))^2 - (g^{1/2}(S))^2. \tag{4.10}$$

The right hand side is a quadratic form and the left hand side is a difference of two submodular functions, which can represent any set function and it has been shown to be inapproximatable to any factor by [57]. Properties of the coefficient matrix $A = [a_{ij}]$ plays a critical role. Consider one case where we can diagonalize the coefficient matrix $A = [a_{ij}] = P^T \Lambda P$ with

non-negative eigenvalues and eigenvectors (non-negative on all entries). We can re-arrange the objective function as

$$\sum_i \lambda_i \tilde{f}_i^2(S), \text{ where } \tilde{f}_i(S) = \sum_j P_{ij} f_j(S). \tag{4.11}$$

If $\lambda_i, P_{ij} \geq 0 \ \forall i, j$, $\lambda_i \tilde{f}_i(\cdot)$ is submodular, so the problem is equivalent to the 2-norm of m submodular functions optimizations. Consider a different case where the rank of the coefficient matrix is 1, we can then factorize $\sum_{i,j} a_{ij} f_i(S) f_j(S)$ as $\prod_i \sum_j Q_{ij} f_j(S)$. If Q_{ij} are all non-negative, the quadratic form objective be reduced to the product of m submodular functions. It remains an open problem on identifying necessary and sufficient conditions of the coefficient matrix so that the quadratic form of submodular functions can be solved approximately with constant guarantees.

4.6 Discussion

4.6.1 Other Approaches

We can apply simpler strategies to achieve weaker constant approximation guarantees on the maximization of the geometric mean and the generalized mean of two monotone submodular functions under cardinality constraints. Essentially, assuming the cardinality constraint k is an even number, we can first use the greedy algorithm to optimize $\max_{S,|S|\leq k/2} f(S)$ and $\max_{S,|S|\leq k/2} g(S)$ separately, denote the corresponding solutions as $\hat{S}_f$ and $\hat{S}_g$ respectively, and take $\hat{S} = \hat{S}_f \cup \hat{S}_g$ as the final solution. Due to submodularity, $f(\hat{S}_f) \geq \frac{1-1/e}{2} f(S^*)$ and $f(\hat{S}_g) \geq \frac{1-1/e}{2} g(S^*)$. We therefore have

$$f(\hat{S})g(\hat{S}) \geq f(\hat{S}_f)g(\hat{S}_g) \geq \frac{\gamma^2}{4} f(S^*)g(S^*), \tag{4.12}$$

$$f(\hat{S})^p + g(\hat{S})^p \geq f(\hat{S}_f)^p + g(\hat{S}_g)^p \geq \frac{\gamma^p}{2^p}(f(S^*)^p + g(S^*)^p). \tag{4.13}$$

Compared to the linear search results, the above guarantee has an extra $1/4$ factor for the product case and an extra 2^{-p} factor for the p-norm case.

Another approach to maximize the product of two monotone submodular functions is to consider the logarithmic of the objective, i.e., $\log(f(S)g(S)) = \log(f(S)) + \log(g(S))$, and

note that since $\log(\cdot)$ is concave, $\log(f(\cdot)) + \log(g(\cdot))$ remains submodular. Any submodular algorithm that achieves a multiplicative guarantee on the logarithmic objective translates into a power bound for the original product objective, i.e., $f(\hat{S})g(\hat{S}) \geq (f(S^*)g(S^*))^{\gamma}$, where $\hat{S}$ is the solution to $\max_{S,S\in\mathbb{C}} \log(f(S))+\log(g(S))$ with multiplicative approximation guarantee γ.

4.6.2 Better Running Time for Maximization of Means of Monotone Submodular Functions

Our linear search based algorithm requires estimation of $\lambda_{\min}$ and $\lambda_{\max}$, which are related to ${f(S^*)}/{g(S^*)}$ and ${g(S^*)}/{f(S^*)}$. For non-monotone submodular functions, we use a naive estimation $\left({\max_S f(S)}/{\min_{v,g(v)>0} g(v)}, {\max_S g(S)}/{\min_{v,f(v)>0} f(v)}\right)$. However, we can improve the bounds given f and g are monotone. As discussed in Section 4.6.1, there is a half-half simple greedy algorithm to achieve a ${(1-1/e)^2}/{4}$ guarantee. The solution can also be used to estimate $f(S^*)$ and $g(S^*)$, since $f(\hat{S}) \geq \frac{1-1/e}{2} f(S^*)$ and $g(\hat{S}) \geq \frac{1-1/e}{2} g(S^*)$.

For the geometric mean (product) problem, we know that $f(\hat{S}) \cdot g(\hat{S}) \leq f(S^*) \cdot g(S^*)$. We have

$$\frac{[1-1/e]^2}{4}\frac{f(\hat{S})}{g(\hat{S})} \leq \lambda^* = \frac{f(A^*)}{g(A^*)} \leq \frac{4}{[1-1/e]^2}\frac{f(\hat{S})}{g(\hat{S})} \tag{4.14}$$

The number of iterations is upper bounded by $\log_{1+\epsilon} \frac{16}{[1-1/e]^4} = O(\frac{1}{\log(1+\epsilon)})$

For the harmonic mean problem, we know $\frac{f(\hat{S})\cdot g(\hat{S})}{f(\hat{S})+g(\hat{S})} \leq \frac{f(S^*)\cdot g(S^*)}{f(S^*)+g(S^*)}$. We have

$$\left[\frac{e+1}{e-1} + \frac{2eg(\hat{S})}{(e-1)f(\hat{S})}\right]^{-2} \leq \lambda^* = \frac{f^2(S^*)}{g^2(S^*)} \leq \left[\frac{e+1}{e-1} + \frac{2ef(\hat{S})}{(e-1)g(\hat{S})}\right]^{2} \tag{4.15}$$

The number of iterations is upper bounded by $\log_{1+\epsilon} \frac{4e^2}{(e-1)^2} = O(\frac{1}{\log(1+\epsilon)})$

For Generalized mean (p-norm) problem, our bound is $\frac{\gamma^p}{(1+\epsilon)^p}$, but there is also a naive bound $\frac{\gamma^p}{1+\frac{f^p(S^*)}{g^p(S^*)}}$ by just optimizing $f^p(S)$ itself. So we only need to search for $\epsilon^{-1} \leq \frac{f(S^*)}{g(S^*)} \leq \epsilon$, i.e. $\epsilon^{1-p} \leq \lambda^p \leq \epsilon^{p-1}$. The number of iterations is upper bounded by $\frac{-\log(\epsilon)}{\log(1+\epsilon)}$.

4.6.3 Product v.s. Robust: Product Guarantees Robust but Robust does not Guarantee Product.

Orlin et al. [98] gives a constant approximation for the robust monotone submodular function

maximization of two submodular functions under a cardinality constraint:

$$\max_{S,|S|\leq k} \min(f(S), g(S)). \tag{4.16}$$

A disadvantage of the robust formulation is that the objective can saturate if only one function saturates due to submodularity, e.g., f dominates g and g saturates due to submodularity. In such situations, additional gains on f do not contribute to the robust objective at all. On the contrary, the product of f and g does not have the saturation problem, i.e. the objective function saturates if and only if both functions saturate. Moreover, we claim that maximizing the product also optimizes the robust objective with a constant guarantee. Consider $S_p^* \in \text{argmax}_{|S|\leq k} f(S) \cdot g(S)$ and $S_m^* \in \text{argmax}_{|S|\leq k} \min(f(S), g(S))$. Therefore $\min(f(S_p^*), g(S_p^*)) \geq \frac{(e-1)^2}{4e^2} \min(f(S_m^*), g(S_m^*))$, but S_m^* does not have any approximation guarantee on the product maximization problem. Thus, we claim that product guarantees robust but robust does not guarantee product.

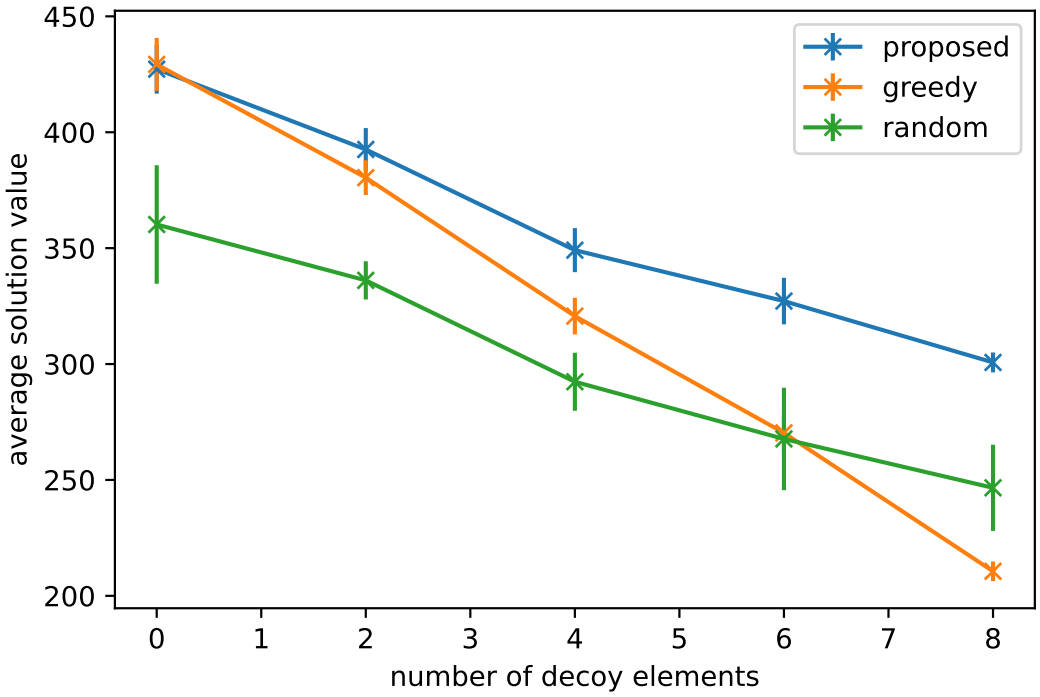
4.7 Experiments

We perform synthetic experiments for proof-of-concept and compare our proposed algorithms with two baseline heuristics methods: greedy on the objective and random subset selection. For the first experiment, we investigate the problem of maximizing a product of two submodular functions under a cardinality constraint. We use the facility location function for f and g, which is a widely used submodular function in practice. We set the ground set size $|V|$ to 30 and partition V into V_1 and V_2, $|V_1| = |V_2| = {}^{|V|}\!/_2$. We define f on V_1 and g on V_2, i.e. $f(V_2) = g(V_1) = 0$. f and g are both facility location functions, i.e., $f(S) = \sum_{v\in V_1} \max_{u\in S} w_{v,u}$ and $g(S) = \sum_{v\in V_2} \max_{u\in S} w'_{v,u}$, where $w_{v,u}$ and $w'_{v,u}$ are two randomly generated matrices with entries in $[0, 1]$. To test the robustness of various methods, we put $t = 0, 2, 4, 6, 8$ decoy elements into the ground set, where $f(v) = g(v) = 0$ for a decoy element v. For the second synthetic experiment, we test the performance of various methods on the problem of maximizing the generalized mean (p-norm) of two submodular functions. In this case, f has the same definition as the previous experiment, and we set $g(S) = \frac{\min(|S\cap V_2|,4)}{4}$ with

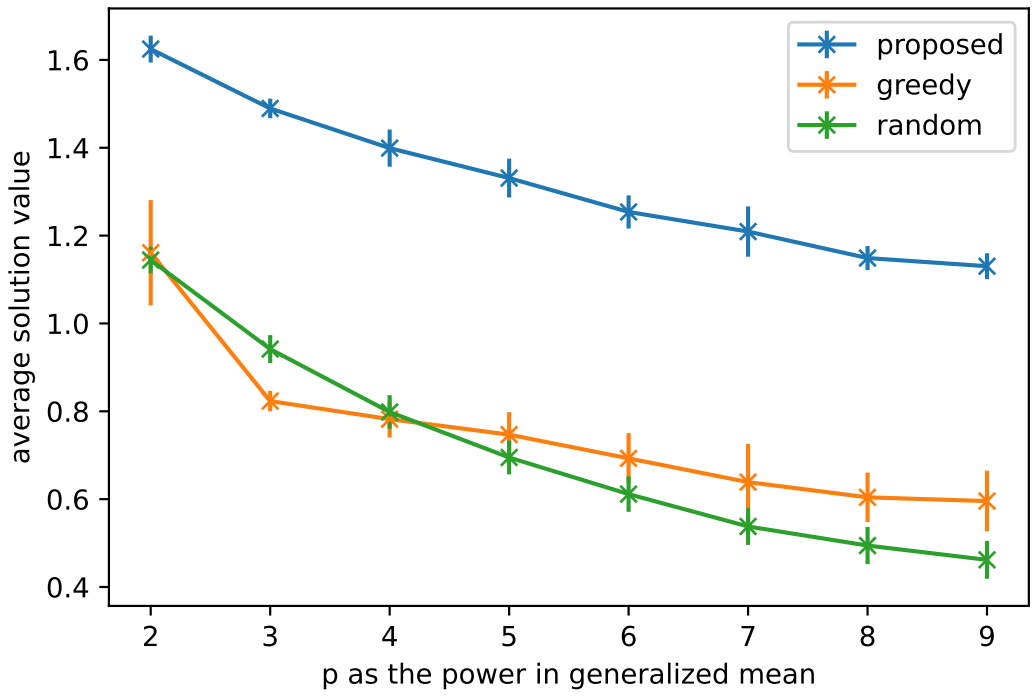
$n = 30$ and $k = 6$. g is designed to be hard for the greedy heuristics, since it has larger gains in the beginning and saturates early. Figure 4.7.1a and 4.7.1b show the results of comparison, where we find the proposed algorithms can consistently outperform the baseline heuristics. The differences also tend to be more significant as we have more adversaries in the function construction, thanks to the constant approximation guarantees of the proposed algorithms.

4.8 Conclusions

We study the maximization and minimization of three mean objectives of submodular functions, including the geometric mean, the generalized mean and the harmonic mean. We reduce those three mean objectives to the problem of arithmetic mean and robust mean. The general combinations of submodular functions in the studied three mean objectives include the product, the p-norm, and the ratio between the product and the sum of submodular functions, which allow us to design objectives involving complex trade-offs between submodular functions, and apply submodular optimization to a wider range of applications. Our approach also gives rise to many future research directions that are deeply involved about the computability of combinations of submodular functions: are there algorithms that have a polynomial dependence on m for optimizing various means of m submodular functions, and what are the necessary and sufficient conditions for the combinations of submodular functions to be approximated with constant factors in polynomial time?



(a) Maximization of Geometric Mean (product)



(b) Maximization of Generalized Mean (p-norm)

Appendices

4.A Proof of Theorem 16

Proof. Let $S^* \in \text{argmin}_{S\subseteq V, S\in\mathbb{C}} f(S)\cdot g(S)$, let $\lambda^* = {}^{f(S^*)}/_{g(S^*)}$. Algorithm 9 is searching for λ in a way like $\lambda_0, (1+\epsilon)\lambda_0, (1+\epsilon)^2\lambda_0, \ldots, (1+\epsilon)^N\lambda_0$ plus two boundary cases $\lambda = 0, \infty$. If we choose $\lambda_0 = \frac{\min_{v\in V, f(v)>0} f(v)}{\max_{A\subseteq V} g(A)}$ and $N \geq \log_{1+\epsilon} \frac{\max_{A\subseteq V} f(A)}{\lambda_0 \min_{v\in V, g(v)>0} g(v)}$, we can always find $\hat{\lambda}$ such that $\frac{\lambda^*}{\sqrt{1+\epsilon}} \leq \hat{\lambda} \leq \sqrt{1+\epsilon}\cdot\lambda^*$. Let $\hat{S}$ be the output by $\hat{\lambda}$

Therefore, we have

$$
\begin{aligned}
f(\hat{S})\cdot g(\hat{S}) &\leq \frac{1}{\hat{\lambda}}\left[\frac{f(\hat{S})+\hat{\lambda}g(\hat{S})}{2}\right]^2 && (4.17)\\
&\leq \frac{\gamma^2}{\hat{\lambda}}\left[\frac{f(S^*)+\hat{\lambda}g(S^*)}{2}\right]^2 && (4.18)\\
&\leq \frac{\sqrt{1+\epsilon}\cdot\gamma^2}{4\lambda^*}\left[f(S^*)+\sqrt{1+\epsilon}\cdot\lambda^*\cdot g(S^*)\right]^2 && (4.19)\\
&\leq \frac{\sqrt{1+\epsilon}\cdot\gamma^2\cdot\left(1+\sqrt{1+\epsilon}\right)^2}{4\lambda^*}\left[f(S^*)\right]^2 && (4.20)\\
&= \frac{\sqrt{1+\epsilon}\cdot\left(1+\sqrt{1+\epsilon}\right)^2}{4}\cdot\gamma^2\cdot f(S^*)\cdot g(S^*) && (4.21)\\
&< (1+\epsilon+{}^{\epsilon^2}/_{16})\cdot\gamma^2\cdot f(S^*)\cdot g(S^*). && (4.22)
\end{aligned}
$$

The first inequality uses simple algebra, and the second inequality uses the γ approximation guarantee on the reduced problem. The third and the fourth inequality uses relationships between $\hat{\lambda}$ and λ^*: $\frac{\lambda^*}{\sqrt{1+\epsilon}} \leq \hat{\lambda} \leq \sqrt{1+\epsilon}\cdot\lambda^*$, and the definition for λ^*: $\lambda^* = {}^{f(S^*)}/_{g(S^*)}$.

Since we search over all possible λ values in an exponentially-spaced sequence, we can always find the $\hat{\lambda}$ with the property stated above. As we take the best over all searched λ values, the found result will be better than $\hat{\lambda}$ and gives the desired approximation guarantee. □ □

4.B Proof of Theorem 17

Proof. The concept of proof is similar with previous ones. Let $\lambda^* = {g^{p-1}(S^*)}/{f^{p-1}(S^*)}$, where $S^* \in \operatorname{argmax}_{S \in \mathcal{C}} f^p(S) + g^p(S)$. We use the same definition for $\hat{\lambda}$ and $\hat{S}$.

$$f^p(\hat{S}) + g^p(\hat{S}) \geq \frac{\left[f(\hat{S}) + \hat{\lambda} g(\hat{S})\right]^p}{\left[1 + \hat{\lambda}^q\right]^{p/q}} \tag{4.23}$$

$$\geq \frac{\gamma^p \left[f(S^*) + \hat{\lambda} g(S^*)\right]^p}{\left[1 + \hat{\lambda}^q\right]^{p/q}} \tag{4.24}$$

$$\geq \frac{\gamma^p \left[f(S^*) + \frac{(1+\epsilon)^{-1/2} g^{p-1}(S^*)}{f^{p-1}(S^*)} g(S^*)\right]^p}{\left[1 + \left[\frac{(1+\epsilon)^{1/2} g^{p-1}(S^*)}{f^{p-1}(S^*)}\right]^q\right]^{p/q}} \tag{4.25}$$

$$\geq \frac{\gamma^p \left[f(S^*) + \frac{g^{p-1}(S^*)}{f^{p-1}(S^*)} g(S^*)\right]^p}{(1+\epsilon)^p \left[1 + \left[\frac{g^{p-1}(S^*)}{f^{p-1}(S^*)}\right]^q\right]^{p/q}} \tag{4.26}$$

$$= \frac{\gamma^p \left[f^p(S^*) + g^p(S^*)\right]^p}{(1+\epsilon)^p \left[f^{q(p-1)}(S^*) + g^{q(p-1)}(S^*)\right]^{p/q}} \tag{4.27}$$

$$= \frac{\gamma^p \left[f^p(S^*) + g^p(S^*)\right]^p}{(1+\epsilon)^p \left[f^p(S^*) + g^p(S^*)\right]^{p/q}} \tag{4.28}$$

$$= \frac{\gamma^p}{(1+\epsilon)^p} \left[f^p(S^*) + g^p(S^*)\right] \tag{4.29}$$

The first inequality follows Hölder's inequality applied to two vectors: $(f(\hat{S}), g(\hat{S}))$ and $(1, \lambda)$. The second inequality uses the γ approximation guarantee of the reduced problem. The third and the fourth inequalities use the definition for λ^* and $\hat{\lambda}$. The following equations follow ${1}/{q} + {1}/{p} = 1$, using properties such as $(p-1)q = p$, and $p - {p}/{q} = 1$. □ □

4.C Proof of Theorem 18

Proof. First, we prove the following inequality: $\forall\ a, b, \lambda \geq 0$, $p \geq 1$, $a + b > 0$,

$$[a^p + b^p]^{1/p} \leq (1 + \lambda^{-p})^{1/p} \max(a, \lambda b) \tag{4.30}$$

If $a \geq \lambda b$, the right hand side equals to $(1 + \lambda^{-p})^{1/p} \cdot a \geq (1 + \left(\frac{a}{b}\right)^{-p})^{1/p} \cdot a = [a^p + b^p]^{1/p}$.

If $a \leq \lambda b$, the right hand side equals to $(1+\lambda^{-p})^{1/p} \cdot \lambda \cdot b = (1+\lambda^p)^{1/p} \cdot b \geq (1+\left(\frac{a}{b}\right)^p)^{1/p} \cdot b = [a^p + b^p]^{1/p}$.

Therefore, let $\lambda^* = \frac{f(S^*)}{g(S^*)}$. We can use linear search to find $\frac{\lambda^*}{\sqrt{1+\epsilon}} \leq \hat{\lambda} \leq \sqrt{1+\epsilon} \cdot \lambda^*$.

$$\begin{aligned}
f^p(\hat{S}) + g^p(\hat{S}) &\leq \left[(1 + \hat{\lambda}^{-p})^{1/p} \max(f(\hat{S}), \hat{\lambda} g(\hat{S}))\right]^p && (4.31)\\
&\leq \left[(1 + \hat{\lambda}^{-p})^{1/p} \gamma \max(f(S^*), \hat{\lambda} g(S^*))\right]^p && (4.32)\\
&\leq \left[(1 + ((1+\epsilon)^{-1/2}\lambda^*)^{-p})^{1/p} \gamma \max(f(S^*), (1+\epsilon)^{1/2}\lambda^* g(S^*))\right]^p && (4.33)\\
&\leq \left[(1+\epsilon)(1 + (\lambda^*)^{-p})^{1/p} \gamma \max(f(S^*), \lambda^* g(S^*))\right]^p && (4.34)\\
&= (1+\epsilon)^p \gamma^p \left[\left(1 + \left(\frac{f(S^*)}{g(S^*)}\right)^{-p}\right)^{1/p} f(S^*)\right]^p && (4.35)\\
&= (1+\epsilon)^p \gamma^p \left[f^p(S^*) + g^p(S^*)\right] && (4.36)
\end{aligned}$$

The first inequality follows from the inequality proved in Eq. 4.30. The second inequality is based on the γ approximation guarantee of the reduced problem. The third and the fourth inequalities follow the definitions for λ^* and $\hat{\lambda}$. □

□

4.D Proof of Theorem 19

Proof. First we prove the following inequality, for any $a, b, \lambda \geq 0$, $a + b > 0$

$$\frac{ab}{a+b} \geq \frac{\min(a, \lambda b)}{1+\lambda} \tag{4.37}$$

If $a \geq \lambda b$, the right hand side equals to $\frac{\lambda b}{1+\lambda} = \frac{b}{1+1/\lambda} \leq \frac{b}{1+b/a} = \frac{ab}{a+b}$.

If $a < \lambda b$, the right hand side equals to $\frac{a}{1+\lambda} < \frac{a}{1+a/b} = \frac{ab}{a+b}$.

Let $\lambda^* = \frac{f(S^*)}{g(S^*)}$. We can use linear search to find $\frac{\lambda^*}{\sqrt{1+\epsilon}} \leq \hat{\lambda} \leq \sqrt{1+\epsilon} \cdot \lambda^*$. So we have

$$\frac{f(\hat{S}) * g(\hat{S})}{f(\hat{S}) + g(\hat{S})} \geq \frac{\min\left(f(\hat{S}), \hat{\lambda} g(\hat{S})\right)}{1+\hat{\lambda}} \tag{4.38}$$

$$\geq \frac{\gamma \min\left(f(S^*), \hat{\lambda} g(S^*)\right)}{1+\hat{\lambda}} \tag{4.39}$$

$$\geq \frac{\gamma \min\left(f(S^*), (1+\epsilon)^{-1/2} \lambda^* g(S^*)\right)}{1+(1+\epsilon)^{1/2}\lambda^*} \tag{4.40}$$

$$\geq \frac{\gamma \min\left(f(S^*), \lambda^* g(S^*)\right)}{(1+\epsilon)(1+\lambda^*)} \tag{4.41}$$

$$= \frac{\gamma f(S^*)}{(1+\epsilon)(1+\frac{f(S^*)}{g(S^*)})} \tag{4.42}$$

$$= \frac{\gamma f(S^*) g(S^*)}{(1+\epsilon)(f(S^*)+g(S^*))} \tag{4.43}$$

The first inequality follows the inequality we have proved in Eq. 4.37. The second inequality follows the γ approximation guarantee of the reduced problem. The third and the fourth inequality follows the definitions of λ^* and $\hat{\lambda}$. □

□

4.E Proof of Theorem 20

Proof. Let $\lambda^* = \frac{f^2(S^*)}{g^2(S^*)}$. We can use linear search to find $\frac{\lambda^*}{\sqrt{1+\epsilon}} \leq \hat{\lambda} \leq \sqrt{1+\epsilon} \cdot \lambda^*$. So we have

$$\frac{f(\hat{S})g(\hat{S})}{f(\hat{S})+g(\hat{S}))} \leq \frac{f(\hat{S})+\hat{\lambda}g(\hat{S})}{(1+\hat{\lambda}^{1/2})^2} \tag{4.44}$$

$$\leq \frac{\gamma\left(f(S^*)+\hat{\lambda}g(S^*)\right)}{(1+\hat{\lambda}^{1/2})^2} \tag{4.45}$$

$$\leq \frac{\gamma\left(f(S^*)+(1+\epsilon)^{1/2}\lambda^* g(S^*)\right)}{(1+(((1+\epsilon)^{-1/2})\lambda^*)^{1/2})^2} \tag{4.46}$$

$$\leq \frac{(1+\epsilon)\gamma\left(f(S^*)+\lambda^* g(S^*)\right)}{(1+(\lambda^*)^{1/2})^2} \tag{4.47}$$

$$= \frac{(1+\epsilon)\gamma f(S^*)g(S^*)}{f(S^*)+g(S^*)} \tag{4.48}$$

For the first inequality, we use Hölder's inequality for $0 < p < 1$ and $q < 0$, i.e., $|xy| \geq ||x||_p ||y||_q$ where x, y are two vectors. In our case, we apply it to vectors: $(f(\hat{S}, g(\hat{S}))$ and $(1, \lambda)$. For the second inequality, we utilize the γ approximation guarantee of the reduced problem. The third and the fourth inequalities follow the definitions of λ^* and $\hat{\lambda}$. □

□

4.F Proof for maximization of Geometric mean of m submodular functions

Here we give a detailed proof to demonstrate the extension to m submodular functions of Theorem 15 as discussed in Section 4.5.1. The extension of other theorems follow the same idea.

Proof. The optimization problem is $\max_{S\in\mathcal{C}} \prod_i^m f_i(S)$, where every $f_i(S)$ is a non-negative submodular function. We define $\lambda_1^* = 1$ and $\lambda_i^* = \frac{f_1(S^*)}{f_i(S^*)}$ for $i = 2, 3, \ldots, m$. λ_i^*'s are unknown but we use multi-linear search to find $\hat{\lambda}_i$ such that $\frac{1}{\sqrt{1+\epsilon}}\lambda_i^* \leq \hat{\lambda}_i \leq \sqrt{1+\epsilon}\lambda_i^*$. The search requires $\prod_{i=2}^m \log_{1+\epsilon} \frac{\max f_1(S) \max f_i(S)}{\min f_1(S) \min f_i(S)}$ rounds, which is exponential to m, but polynomial to other parameters for fixed m. Suppose we can find $\hat{S}$ as an approximate solution to $\max_{S\in\mathcal{C}} \min_i(\lambda_i \cdot f_i(S))$. We have

$$\prod_{i=1}^m f_i(\hat{S}) \geq \frac{\left[\min_i(\hat{\lambda}_i \cdot f_i(\hat{S}))\right]^m}{1 \cdot \prod_{i=2}^m \hat{\lambda}_i} \tag{4.49}$$

$$\geq \frac{\left[\gamma \min_i(\hat{\lambda}_i \cdot f_i(S^*))\right]^m}{\prod_{i=2}^m \hat{\lambda}_i} \tag{4.50}$$

$$\geq \frac{\left[\gamma \min_i(\frac{\lambda_i^*}{\sqrt{1+\epsilon}} \cdot f_i(S^*))\right]^m}{\prod_{i=2}^m \sqrt{1+\epsilon}\lambda_i^*} \tag{4.51}$$

$$\geq \frac{\gamma^m \left[\min_i(\lambda_i^* \cdot f_i(S^*))\right]^m}{(1+\epsilon)^{\frac{2m-1}{2}} \prod_{i=2}^m \lambda_i^*} \tag{4.52}$$

$$= \frac{\gamma^m \left[\min_i(\frac{f_1(S^*)}{f_i(S^*)} \cdot f_i(S^*))\right]^m}{(1+\epsilon)^{\frac{2m-1}{2}} \prod_{i=2}^m \frac{f_1(S^*)}{f_i(S^*)}} \tag{4.53}$$

$$= \frac{\gamma^m \prod_{i=1}^m f_i(S^*)}{(1+\epsilon)^{\frac{2m-1}{2}}} \tag{4.54}$$

The bound is still tight in terms of the γ dependence and aligns with $m = 2$ case in Theorem 15. □ □

Part II

SUBMODULAR GENERALIZED MATCHING FOR PEPTIDE IDENTIFICATION IN TANDEM MASS SPECTROMETRY

Part II focuses on the biological application of submodular generalized matching (SGM) for peptide identification in tandem mass spectrometry. In chapter 5, we introduce the problem of mass spectrometry database search and point out that a score function is a key to the success of the searching algorithm. In chapter 6, We discuss the design of submodular generalized matching as the score function. In chapter 7, we demonstrate that our proposed score function can be computed efficiently and outperforms a variety of state-of-the-art methods across multiple data sets. The work is published in

Wenruo Bai, *Jeffrey Bilmes, and William Stafford Noble. Submodular generalized matching for peptide identification in tandem mass spectrometry. IEEE/ACM transactions on Computational Biology and Bioinformatics, IEEE, 2018.*

Wenruo Bai, *Jeffrey Bilmes, and William S. Noble. Bipartite Matching Generalizations for Peptide Identification in Tandem Mass Spectrometry. In 7th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM BCB), ACM SIGBio, ACM SIGBio, Seattle, WA, October 2016.*, which is the winner of the **best paper award** at the conference.

Chapter 5
INTRODUCTION

5.1 Introduction

A shotgun proteomics experiment produces on the order of 10 mass spectra per second, each of which ideally is generated by a single peptide species. Hence, before the data can be used to answer high-level biological questions—like which functional classes of proteins are differentially expressed in one experimental condition versus another—we must first answer a simpler question, namely, "What peptide species was responsible for generating this observed spectrum?"

Over the past two decades, since the description in 1994 of the SEQUEST algorithm [29], by far the most common way to answer this question has been via database search. All such methods follow roughly the same form. The input is a set of observed spectra and a database of peptides, typically derived from the protein sequences of the organism under study. The database search algorithm is then deceptively simple: for each observed spectrum, we (1) extract from the database all peptides whose masses lie within a user-specified tolerance of the precursor mass associated with the spectrum, (2) compute a quality score for each peptide-spectrum match (PSM), and (3) assign to the spectrum the candidate peptide that received the best score.

Clearly, the success or failure of a database search method depends very strongly upon the quality of its score function. A good database search score function must exhibit at least three distinct properties. First, it must be quick to compute. At a production rate of 10 spectra per second, where each spectrum must be compared to hundreds or thousands of candidate peptides, an expensive score function will quickly become the bottleneck in any analysis pipeline. Second, the function must be accurate, in the sense that it usually

succeeds in assigning the best score to the candidate peptide that actually was responsible for generating the observed spectrum. Third, the function must be well calibrated, so that the score assigned to the top peptide for one spectrum can be compared directly to the score assigned to the top peptide for a second spectrum. This third property is important because, in practice, the output of a database search algorithm is a ranked list of PSMs, one per observed spectrum. Because many observed spectra cannot be accurately identified, it is critical that the top of this ranked list of PSMs is highly enriched for correct identifications. Calibration is also important for comparing spectra containing different numbers of peaks, since an uncalibrated algorithm tends to give higher scores to more dense spectra.

Dozens of database search score functions have been described in the literature (reviewed in [97]). Most rely on first transforming the peptide sequence into a theoretical spectrum and then computing some type of similarity score between the observed and theoretical spectra. Existing similarity functions rely on cross-correlation (SEQUEST) [29], dot product (X!Tandem) [18], Poisson scoring (OMSSA) [42], hypergeometric scores (Myrimatch) [119], probabilistic models (ProbID) [138] or simple counts of overlapping peaks (Morpheus) [135].

In this work, we propose to model the affinity between an observed and theoretical spectrum using a process we call a "submodular generalized matching" (SGM). This approach generalizes and provides greater modeling power than standard bipartite matching. In order to describe SGMs, we need first to describe bipartite matchings, submodular functions and their optimization, and matroids. We briefly do so in the next few paragraphs and further discuss submodular function in Section 5.2.

A maximum bipartite matching starts with a non-negative weighted bipartite graph (V, U, E, w), where V is a set of "left" vertices, U is a set of "right" vertices, $E \subseteq V \times U$ is a set of edges, and $w : E \to \mathbb{R}_+$ is a weight function on the edges, where $w(A) = \sum_{a \in A} w(a)$ for any edge set $A \subseteq E$. The goal of a maximum bipartite matching process is to find a set of edges $A \subseteq E$ that maximizes $w(A)$ but that is a matching, i.e., no vertex may be incident to more than one edge. Conceptually, one might treat computing a peptide-spectrum matching score as finding a maximum matching in a bipartite graph consisting of an observed spec-

trum (represented by the vertices V), a theoretical spectrum (the vertices U), and the edges E (feasible explanations of the observed by the theoretical spectra). In other words, given an edge $e \in E$ where $e = (v, u)$ with $v \in V$, $u \in U$, the weight $w(e)$ (which may be zero) indicates the degree to which theoretical peak u matches observed peak v.

For several reasons, however, maximum bipartite matching alone is inadequate to produce a good peptide-spectrum scoring function. First, only one edge in a traditional matching may be incident to a vertex, even though, as described below, a given theoretical fragmentation event might produce multiple effects in the observed spectrum. Conversely, several different theoretical peaks might potentially explain a single observed peak. Second, the score function of a bipartite matching $w(A)$ is necessarily additive, meaning that the weight of an edge does not change when considered in the context of other edges added to a matching. In practice, an optimal score function might need to combine matching scores in a non-additive fashion. To address the first problem, we use matroid constraints, and to address the second problem we use submodular functions. Together, these two approaches achieve our generalization.

In fact, bipartite matching can be described in exactly this way. Given a weighted bipartite graph (V, U, E, w), we can formulate maximum bipartite matching as maximizing $w(A)$ subject to A being independent in two matroids. Depending on the matroids (as described below) we may relax the constraint that an edge is incident only to one vertex. In fact, with this formulation, each vertex (within either V or U) may have its own limit on incident edges. This means that, for a vertex $x \in V \cup U$, we may define a limit k_x on how many edges in a generalized matching may be incident to x.

Submodular matching generalizes this idea further as follows: rather than maximize an additive weight function $w(A)$, we instead maximize a submodular function f. That is, submodular matching finds an edge set $A \subset E$ that maximizes $f(A)$ subject to multiple matroid constraints, $A \in \mathcal{I}_1 \cap \mathcal{I}_2$. Submodular matching is NP-hard, but it can be well-approximated extremely efficiently using a greedy algorithm that has a mathematical quality guarantee, namely, that the solution provided by the greedy algorithm (Alg. 11) is no worse than 1/3 times the best possible solution—this approximation ratio is constant regardless

of the problem size [95]. Submodularity can be further exploited to accelerate the greedy implementation, leading to an algorithm often called *accelerated* or *lazy greedy*[90] (Alg. 12) having almost linear time complexity in practice. Hence, computationally, the approach scales to very large data set sizes.

In this work, we demonstrate how submodular matching with matroid constraints can be used to design a natural mass spectrometry score function that incorporates two important pieces of prior knowledge about peptide fragmentation. First, the proposed score function keeps track of situations in which a single observed peak can be explained by more than one peak in the theoretical spectrum. Such a collision might occur, for example, in the fragmentation of the +2 charged peptide SSLEVHIR. One of the prefix ions (SS) has an m/z value nearly exactly equal to one of the suffix ions (R). If the observed spectrum has a peak at 175 Da/charge, then existing score functions must choose between scoring this peak as a single match or as two matches. The submodular approach, by contrast, allows us to assign a diminished score to the second match. Second, our proposed score function allows us to, in effect, assign "extra credit" to pairs of observed-theoretical matches that are mutually reinforcing. For example, when we evaluate the hypothesis that an observed spectrum was produced by the fragmentation of peptide QNSHLTIK, we expect a single cleavage event to produce a prefix ion (e.g., QNS with m/z=330 Da/charge) and its corresponding suffix ion (HLTIK with m/z=611 Da/charge). If the observed spectrum contains peaks at both 611 Da/charge and 330 Da/charge, then SGM offers full joint, or non-diminished, credit to these pair of peaks, to account for their complementary nature. The SGM approach also simultaneously discredits any other sets of peaks that should not be in a complementary relationship with each other, for the given peptide. As we see above, the edge interactions can be both local and global, and this is exactly the power of submodular functions, which can model these properties easily while allowing fast approximate maximization.

We demonstrate that our proposed score function can be computed efficiently and that the resulting score function outperforms a variety of state-of-the-art methods across multiple data sets. Specifically, we compare SGMs with four existing methods, XCorr [21], MS-GF+ [71],

XCorr p-value [50] and Mascot [17]. We compute the number of spectra identified at a 1% false discovery rate (FDR) threshold, observing statistically significant improvements relative to the second-best method ($p < 0.05$, Wilcoxon signed-rank test, Fig 7.4.2).

A related framework is called max b-matching [70]. A b-matching is a set of edges M such that at most $b(v)$ edges in M are incident on each vertex $v \in V$. Recent studies have proposed efficient algorithms to find a b-matching of maximum weight with 1/2 guarantee. We note that this framework and SGM share a similar ability to allow multiple edges incident to one vertex. However, SGM is more flexible because it allows modelling the interaction between edges, whereas b-matching requires simply adding up the edge weights.

5.2 Background: Submodular Function

In this section, we give further background introduction of submodular functions. Recall that a set function $f : 2^E \to \mathbb{R}$ is called submodular if and only if for any $A \subseteq B \subset E$ and $v \in E \setminus B$, f satisfies $f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B)$. We define the *gain of* v *in the context of* X as $f(v|X) \triangleq f(X \cup \{v\}) - f(X)$. Thus, f is submodular if $f(v|X) \geq f(v|Y)$ for $X \subseteq Y$ and $v \notin Y$. The function f is called *monotonically non-decreasing* if $f(v|A) \geq 0$ for all $v \in E$ and $A \subseteq E$, where $f(v|A) = f(\{v\} \cup A) - f(A)$ is the gain of v given A. Submodularity is preserved within a convex cone, that is, if f_1 and f_2 are both submodular functions, then $\lambda_1 f_1 + \lambda_2 f_2$ is also submodular for all $\lambda_1, \lambda_2 \geq 0$.

People have studied different classes of submodular functions such as entropy [88], set cover [41], deep submodular functions[23] and so on. Among them, perhaps the most simple class is that of the concave of modular functions. Suppose that $f : 2^E \to \mathbb{R}$ is a monotone non-decreasing submodular function and $g : \mathbb{R} \to \mathbb{R}$ is a monotone non-decreasing concave function, then $g(f(A))$ is submodular [113].

Optimizing an arbitrary set function is computationally intractable in polynomial time. However, submodularity provides possibilities for a variety of optimization problems. For example, minimizing a submodular function unconstrainedly can be exactly solved in polynomial time [53]. The minimum cut problem is a special case of this general minimization

problem. However, the corresponding maximization problem is known to be NP-hard [31]. On the other hand, maximizing a monotonically non-decreasing submodular function under a cardinality constraint can be approximately solved by a simple greedy algorithm with a worst case lower bound of $1-\frac{1}{e}$ [38]. In recent years, due in part to the increasing set of applications in machine learning that can utilize submodularity, many other related optimization problems have been studied, including submodular cover and submodular knapsack constrained submodular optimization [61], online submodular maximization [115], and optimizing ratios of submodular functions [5], to name only a few.

In this chapter, we consider the problem of submodular maximization under multiple matroid constraints. In 1978, Nemhauser et al. [95] proposed a Greed algorithm (Alg. 11) which is guaranteed to obtain a solution $\hat{A}$ such that

$$f(\hat{A}) \geq \frac{1}{m+1} f(A^*), \tag{5.1}$$

where $A^* \in \operatorname{argmax}_{A \in \cap_i \mathcal{I}_i} f(A)$ and $\mathcal{M}_i = (E, \mathcal{I}_i)$, for $i = 1, \ldots, m$ are a set of m matroids to be used as constraints. The time complexity of Greed is $O(n^2)$ and can be further accelerated by LazyGreed [90] (Alg. 12) which obtains the same solution as Greed. The $\frac{1}{m+1}$ bound can be further improved to $1 - \frac{1}{e}$ [14] for $m = 1$ and $\frac{1}{m+\epsilon}$ [79] for $m \geq 2$ using multilinear extension and the continuous greedy algorithm. These algorithm are more computationally expensive, especially since they cannot use the accelerated greedy trick. Since our goal is not only to produce a good, but also a fast, scoring function, we elected to use the faster algorithms (LazyGreed), having the $\frac{1}{m+1}$ guarantee.

Another benefit of submodularity and SGM is that we can deliberately punish certain pairs of elements by making small changes to the submodular function. This is extremely useful for designing suitable functions whenever we observe good biological properties of MS/MS. For example, if an observed peak is well-explained by a given theoretical peak, then any other theoretical peak that also explains the same observed peak should be discounted, as we will further discuss in section 6.3. Let $f_0 : 2^E \to \mathbb{R}$ be a normalized, monotonically non-decreasing submodular function, and consider two elements $a, b \in E$. Let $f' : 2^E \to \mathbb{R}$ be

Algorithm 11: GREED for submodular maximization. [95]

1: **Input:** f and m matroid constraints $\mathcal{M}_i = (E, \mathcal{I}_i)$.
2: **Output:** An approximation solution $\hat{A}$.
3: $\hat{A} \leftarrow \emptyset$
4: **while** exists $v \in E \setminus \hat{A}$ s.t. $(\{v\} \cup \hat{A}) \in \cap_i \mathcal{I}_i$ **do**
5: $\quad \hat{v} \in \text{argmax}_{v \in E \setminus \hat{A}, (\{v\} \cup \hat{A}) \in \cap_i \mathcal{I}_i} f(v|\hat{A})$
6: $\quad \hat{A} \leftarrow \{\hat{v}\} \cup \hat{A}$
7: **end while**
8: **return** $\hat{A}$

another submodular function where $f'(A) = f'(A \cap \{a, b\})$ for all $A \subseteq E$, $f'(\emptyset) = f'(\{a\}) = f'(\{b\}) = 0$ and $-\min_{v \in \{a,b\}} f_0(v|E \setminus \{v\}) \leq f'(\{a, b\}) \leq 0$. Note that $f'(a|E \setminus \{a\}) = f'(E) - f'(E \setminus \{a\}) = f'(a, b) - f'(b) = f'(a, b) = f'(b|E \setminus \{b\})$ since $f'(v) = 0$ for $v \in \{a, b\}$.

Lemma 23. *$f(A) = f_0(A) + f'(A)$ is a monotonically non-decreasing submodular function.*

Proof. $f'(A)$ is submodular by definition and so is $f(A) = f_0(A) + f'(A)$ since submodularity is preserved when adding submodular functions.

For all $v \in E \setminus \{a, b\}$,

$$f(v|E \setminus \{v\}) = f_0(v|E \setminus \{v\}) + f'(v|E \setminus \{v\}) \tag{5.2}$$

$$= f_0(v|E \setminus \{v\}) \geq 0. \tag{5.3}$$

For $v \in \{a, b\}$,

$$f(v|E \setminus \{v\}) = f_0(v|E \setminus \{v\}) + f'(v|E \setminus \{v\}) \tag{5.4}$$

$$\geq f_0(v|E \setminus \{v\}) - \min_{v \in \{a,b\}} f_0(v|E \setminus \{v\}) \geq 0. \tag{5.5}$$

According to submodularity, if all the final gains are non-negative (e.g. $f(v|E \setminus \{v\}) \geq 0$ for all $v \in E$), then $f(A)$ is monotonically non-deceasing. □

We claim that adding f' to f_0 does not encourage choosing the pair a, b when solving $\max_{A\in\mathcal{C}} f(A) = \max_{A\in\mathcal{C}} f_0(A) + f'(A)$ instead of $\max_{A\in\mathcal{C}} f_0(A)$ where $\mathcal{C}$ is any constraint.

Lemma 24. *For $A^* \in \operatorname{argmax}_{A\in\mathcal{C}} f_0(A)$, if $|A^* \cap \{a, b\}| \leq 1$, then $A^* \in \operatorname{argmax}_{A\in\mathcal{C}} f(A)$ and $\max_{A\in\mathcal{C}} f(A) = \max_{A\in\mathcal{C}} f_0(A)$. If $|A^* \cap \{a, b\}| = 2$, then $\max_{A\in\mathcal{C}} f(A) \leq \max_{A\in\mathcal{C}} f_0(A)$.*

For an α-approximate solution $\hat{A}$ of $\max_{A\in\mathcal{C}} f_0(A)$, if $|\hat{A} \cap \{a, b\}| \leq 1$, then $\hat{A}$ is still an α-approximate solution of $\max_{A\in\mathcal{C}} f(A)$ and $f(\hat{A}) = f_0(\hat{A})$. If $|\hat{A}\cap\{a, b\}| = 2$, $f(\hat{A}) \leq f_0(\hat{A})$.

Proof. Recall the definition of $f'(A)$: $f'(A) = f'(A\cap\{a, b\})$ for all $A \subseteq E$, $f'(\emptyset) = f'(\{a\}) = f'(\{b\}) = 0$ and $-\min_{v\in\{a,b\}} f_0(v|E \setminus \{v\}) \leq f'(\{a, b\}) \leq 0$. Immediately, we have $f(A) = f_0(A)+f'(A) \leq f_0(A)$ for all $A \subseteq E$, since $f'(A) \leq 0$. So $\max_{A\in\mathcal{C}} f(A) \leq \max_{A\in\mathcal{C}} f_0(A)$. This covers the case when $|A^* \cap \{a, b\}| = 2$ where we see score penalty of $\min_{v\in\{a,b\}} f_0(v|E \setminus \{v\})$.

If $|A^*\cap\{a, b\}| \leq 1$, then $f'(A^*) = 0$ and $f(A) = f_0(A)+f'(A) \leq f_0(A) \leq f_0(A^*) \leq f(A^*)$ for all $A \subseteq E$. So $A^* \in \operatorname{argmax}_{A\in\mathcal{C}} f(A)$ and $\max_{A\in\mathcal{C}} f(A) = f(A^*) = f_0(A^*) + f'(A^*) = f_0(A^*) + 0 = \max_{A\in\mathcal{C}} f_0(A)$. This shows that when $|A^* \cap \{a, b\}| \leq 1$, score is not penalized.

For the second part of α-approximate solution $\hat{A}$, again we have $f(\hat{A}) \leq f_0(\hat{A})$. This covers the case when $|\hat{A} \cap \{a, b\}| = 2$, and we see score penalty of $\min_{v\in\{a,b\}} f_0(v|E \setminus \{v\})$.

If $|\hat{A} \cap \{a, b\}| \leq 1$, then we have $f(\hat{A}) = f_0(\hat{A}) + f'(\hat{A}) = f_0(\hat{A}) + 0 = f_0(\hat{A})$. Therefore, $\frac{f(\hat{A})}{\max_{A\in\mathcal{C}} f(A)} = \frac{f_0(\hat{A})+f'(\hat{A})}{\max_{A\in\mathcal{C}} f(A)} = \frac{f_0(\hat{A})}{\max_{A\in\mathcal{C}} f(A)} \geq \frac{f_0(\hat{A})}{\max_{A\in\mathcal{C}} f_0(\hat{A})} \geq \alpha$. So $\hat{A}$ is still an α-approximate solution for $\max_{A\in\mathcal{C}} f(A)$ and the score is not penalized.

□

Hence, adding f' will only punish the score of the pair a and b; otherwise, the scores will be unaffected.

In practice, we use a concave over modular function, $f(A) = g(\sum_{v\in A\cap\{a,b\}} w_v)$, where g is a monotonically non-deceasing concave function, and w is a non-negative weight for each element. Immediately, we have $f(A) = f_0(A) + f'(A)$ where $f_0(A) = \sum_{v\in A\cap\{a,b\}} g(w_v)$ and $f'_0(A) = \mathbf{1}_{A=\{a,b\}}(g(w_a) + g(w_b) - g(w_a + w_b))$, and using lemma 24, we show that f does not encourage choosing the pair of a, b compared to $\sum_{v\in A\cap\{a,b\}} g(w_v)$, which has no interaction between elements. But unfortunately, the opposite of lemma 24, where we would

award a particular pair of complementary elements, does not hold. It is hard to model complementary properties of multiple elements using submodularity. However, in this part, we use an interesting trick (see Section 6.3.2 for full details) to achieve this goal.

5.3 Background: Mass Spectrometry Database Search

In this section, we introduce the spectrum identification problem. Given an observed spectrum dataset S and a peptide database $\mathcal{P}$, for each $s \in S$ with precursor m/z value of m^s and precursor charge value c^s, we wish to find the peptide $p \in \mathcal{P}$ responsible for generating s. With the knowledge of m^s and c^s, it is unnecessary to consider all p in the entire database. We assume that the responsible p has a mass approximately equal to the precursor m^s, subject to a tolerance ω determined by the settings in the first round of MS/MS. Let $P_s = P(m^s, c^s, \mathcal{P}, \omega) = \{p : s \in \mathcal{P}, |m(y)/c^s - m^s| \leq \omega\}$. In database search, only the peptides in P_s, the *candidate peptides*, are scored. A scored spectrum-peptide pair is called a peptide-spectrum match (PSM).

Denoting an arbitrary scoring function as $\mathfrak{s}(p, s)$, the spectrum identification problem, for a given s, computes:

$$p^* \in \underset{p \in P(m^s, c^s, \mathcal{P}, \omega)}{\operatorname{argmax}} \mathfrak{s}(p, s) \tag{5.6}$$

The scoring function $\mathfrak{s}(p, s)$ is the core of any search method and a crucial determinant of the performance. We next introduce the widely used SEQUEST method and then show how we generalize it using submodular generalized matching.

5.3.1 The SEQUEST algorithm

SEQUEST [29] was the very first database search engine. It has a rather simple mechanism and runs very fast while having good performance. Nowadays, SEQUEST is still widely used, and many modern search engines such as MS-GF+ cite [71] and the XCorr p-value [50], which will be later described in Section 7.2, build their algorithms using ideas similar to that of SEQUEST.

We begin by describing SEQUEST because it also provides a good starting point for SGM. Prior to analysis, each observed spectrum is preprocessed in two steps. First, the intensity of each peak is replaced by its square root. Second, the m/z range spanned by the spectrum is divided into 10 regions uniformly, and the intensities within each region are normalized so that the highest intensity in that region is 50. This second step reduces the amount of intensity variation along the m/z axis.

Next, a theoretical spectrum is constructed for each candidate peptide. A peptide with n amino acids is fragmented into $n-1$ prefix ions (called "b-ions") and $n-1$ suffix ions (called "y-ions"). For each b-ion and y-ion with mass m_b and m_y, theoretical peaks with intensity 50 are placed at m_b and m_y, and neutral loss peaks with intensity 10 are placed at $m_b - 17$, $m_b - 18$, $m_b - 28$, $m_y - 17$. $m_y - 18$, corresponding to losses of ammonia (NH_3, 17 Da), water (H_2O, 18 Da) and carbon monoxide (CO, 28 Da). Furthermore, if the observed spectrum has a precursor charge greater than +2, then higher charged versions of the above peaks are also added to the theoretical spectrum.

The traditional SEQUEST score function, called XCorr, is a dot product between one theoretical and one observed spectrum, and can be calculated as follows:

$$\text{SEQUEST}(p, \tilde{s}) = \langle p, \tilde{s} \rangle - \frac{1}{151} \sum_{\tau=-75}^{75} \sum_{i=1}^{N} p(i)\tilde{s}(i-\tau) \tag{5.7}$$

$$= \langle p, \tilde{s} - \frac{1}{151} \sum_{\tau=-75}^{75} \tilde{s}_\tau \rangle \tag{5.8}$$

where $\tilde{s}$ is the observed spectrum, p is the theoretical spectrum and $\tilde{s}' = \tilde{s} - \frac{1}{151}\sum_{\tau=-75}^{75} \tilde{s}_\tau$ is called the background spectrum with $\tilde{s}_\tau(i) = \tilde{s}(i-\tau)$.

Algorithm 12: LazyGreed for submodular maximization. [90]

Input: f and m matroid constraints $\mathcal{M}_i = (E, \mathcal{I}_i)$.
Output: An approximation solution $\hat{A}$.
$\hat{A} \leftarrow \emptyset$; Initialize priority queue Q.
for $v \in E$ **do**
  INSERT$(Q, f(v))$
end for
while Q not empty **do**
  $(v, a) \leftarrow$ POP(Q)
  if $(\{v\} \cup \hat{A}) \in \cap_i \mathcal{I}_i$ **then**
    isFresh $\leftarrow (\alpha = f(v|\hat{A}))$
    if not isFresh **then**
      $\alpha \leftarrow f(v|\hat{A})$
    end if
    if isFreash or $\alpha \geq \max(Q)$ **then**
      $\hat{A} \leftarrow \hat{A} \cup \{v\}$
    else
      INSERT$(Q, (v, \alpha))$
    end if
  end if
end while
return $\hat{A}$

Chapter 6

SUBMODULAR GENERALIZED MATCHINGS

Producing a score function S via SGM involves four steps. First, we create a distinct bipartite graph where the left vertices V correspond to the observed peaks and the right vertices U to theoretical peaks for each PSM needing to be scored. Second, we produce a submodular evaluation function $f(A)$ defined over edges of that bipartite graph. Third, we define a set of matroids $\mathcal{M}_v = (E, \mathcal{I}_v)$ and $\mathcal{M}_u = (E, \mathcal{I}_u)$ whose independent sets are to be used as constraints. Fourth, we compute the score itself, $\mathfrak{s} = \max_{A \in \mathcal{I}_v \cap \mathcal{I}_u} f(A)$. We discuss each of these steps in detail below.

6.1 Bipartite Graph Production

All of our models have as their core a bipartite graph representation of the matching between observed and theoretical spectra peaks (Figure 6.1.1A). A bipartite graph is one whose vertices can be divided into two disjoint sets U and V such that every graph edge $e = (v, u)$ connects a vertex $v \in V$ to a vertex $u \in U$. For each PSM, we build a bipartite graph $G = (V, U, E)$, where V and U are the sets of peaks in the observed and theoretical spectrum, respectively, and for $e = (v, u) \in E$, theoretical peak u is responsible for the existence of observed peak v with a corresponding weight $w(e)$.

For a given edge $e = \{v, u\}$, the weight $w(e)$ is defined as $w'(\{v, u\})x_v y_u$, where $w'(\{v, u\})$ is a weight matrix and x_v, y_u are intensities of v and u. $w'(\{v, u\})$ describes the general biological relationship between the observed and theoretical peaks given their mass-to-charge ratios, m_v and m_u. For example, if $m_u - m_v$ is close to 0 or 18 (a water loss), then $w'(\{v, u\})$ should be high. Note that the matrix w' is sparse since we do not expect relationships to exist between two peaks at an arbitrary m/z difference. Also, the values of w' are a function only

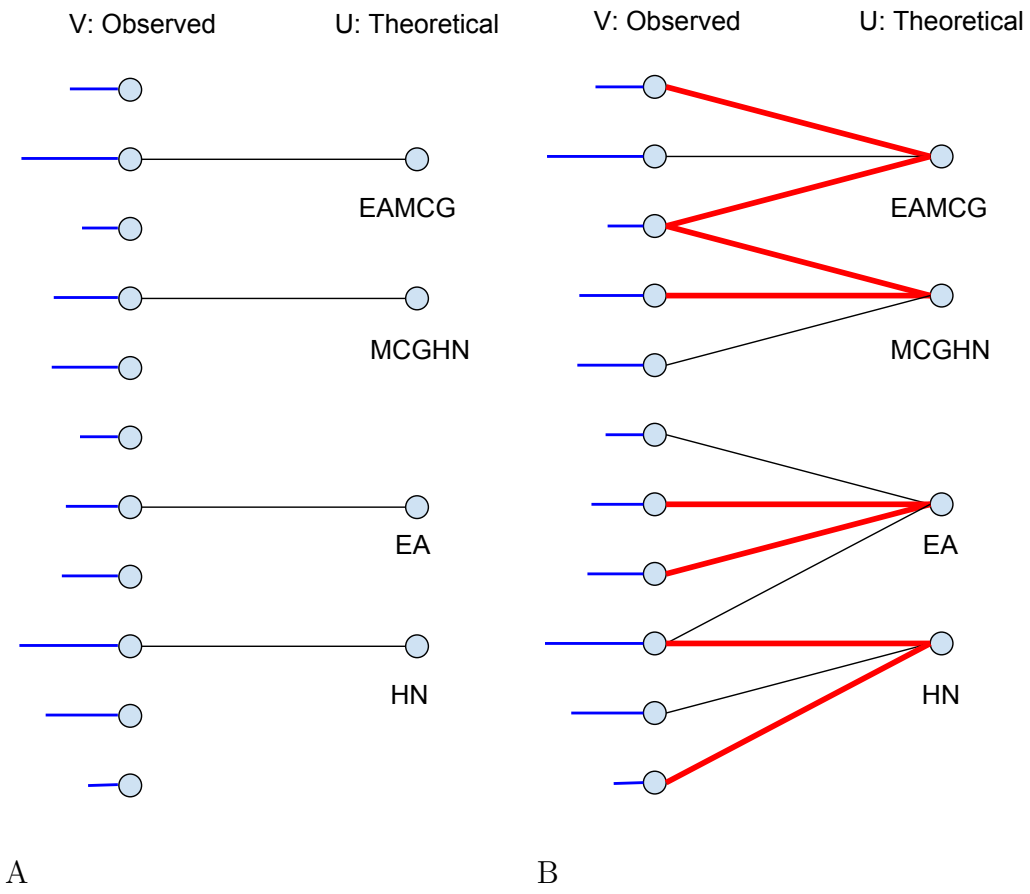


Figure 6.1.1: **PSM bipartite graphs.** (A) V is the set of all peaks in an observed spectrum. The horizontal lines attached on the left represent the peak intensities in the observed spectrum. U is the set of all fragment ions for a given theoretical spectrum derived from a given peptide. An edge (v, u) connects $v \in V$ with $u \in U$ if v might possibly be explained by u with an associated non-negative weight. (B) Red lines are selected edges for a particular match. Non-horizontal edges might correspond to neutral losses.

of the m/z difference, i.e., $w'(\{v,u\}) = h(m_v - m_u)$ where $h(\cdot)$ is a function and is learned empirically using the average intensity near high-confidence b-ions and y-ions (Section 6.2). Note that our empirical weights may also be helpful for other scoring methods. In Section 7.6, we show that using our empirical weighting scheme does indeed yield improvements for a method like SEQUEST; however, the submodular function introduced in the next section makes better use of this weighting information.

6.2 Empirical Weights

In this section, we show how we derive our empirical weights that lead to the improved performance demonstrated in Figure 7.6.1.

Figure 6.2.1 shows the average intensities of observed peaks near b-ions or y-ions in high confidence PSMs ($q = 0.01$) for the worm-01 charge +2 data set. Figure 6.2.2 shows the average intensities for one data set, *Plasmodium* TMT-10 (described in Section 7.1). In the figures, we see strong peaks at the b- and y-ions, as well as peaks with offsets of +1 Th peaks (+1 isotope peaks), -17 Th (NH_3-loss), -18 Th (H_2O-loss) and (for b-ions) -28 Th peaks (CO-loss).

The edges are added to E based on these average intensities for b-ion and y-ions separately. The weight of each edge is $w(\{v,u\}) = w'(\{v,u\})x(m_v)$, where m_v is the m/z of v, and x is the preprocessed observed spectrum as introduced in Section 5.3.1. Recall that in Section 6.1, $w'(\{v,u\})$ is determined by $m_v - m_u$. For low resolution data, $w'(\{v,u\})$ is read from Table 6.2.1. For high resolution data, the $w'(\{v,u\})$ values are obtained from Figure 6.2.2. In both settings, we calculate $m_v - m_u$ and use the number in the table or intensity in the plot.

For charge +3 data, where the b-ions and y-ions are charge +1 or charge +2, we do similar steps but using the mass-to-charge ratio of the higher charged ion, where $m_{v,c+} = \frac{m_{v,1+}+c-1}{c}$ is the m/z of charge $c+$ ion of v.

Table 6.2.1: The empirical weights $w(\{e_v, e_u\})$ used for low resolution data (yeast and worm). For each v and u, we compute the mass-to-charge ratio difference $m_v - m_u$ and read the correspond entry from the table.

b-ion						
$m_v - m_u$	-28	-27	-19	-18	-17	-16
$w'(\{v,u\})$	0.1101	0.0225	0.0121	0.3128	0.2364	0.0784
$m_v - m_u$	-15	-12	-1	0	+1	+2
$w'(\{v,u\})$	0.0112	0.0107	0.0481	0.6122	0.2514	0.0511
y-ion						
$m_v - m_u$	-18	-17	-16	0	+1	+2
$w'(\{v,u\})$	0.1364	0.1179	0.0345	1	0.4253	0.0741

6.3 Submodular Evaluation Function

In this section, we define the submodular set function $f(A)$ producing a score of an edge set A, that corresponds to how well a set of theoretical peaks (the vertex subset of U incident to edges A) explains a set of observed peaks (the vertex subset of V incident to edges A). While the function f may evaluate an arbitrary subset of edges, we only consider sets A that constitute matchings subject to the matroid constraints when producing PSM scores,. Therefore, we assume that A is a matching when describing this function.

Before discuss the particular function we have designed for PSM scoring, we describe a set of properties of f that naturally lead us to the class of monotone non-decreasing submodular functions. First, in general, we want the theoretical spectrum to explain as much of the observed spectrum as possible. This means that if one match is a subset of the other, then we expect the first to score no greater than the second. In other words, f should be monotone non-decreasing, or $f(A) \leq f(B)$ for all $A \subseteq B$. Second, in many cases, we would not wish to over-credit a given set of selected edges. For example, if an observed peak is well-explained by a given theoretical peak, then any other theoretical peak that also explains the same observed peak should be discounted or, in some sense, "explained away." Without some kind of discounting procedure, we would be overconfident about this observed peak. Moreover, a non-discounted score function might discourage an optimization algorithm from

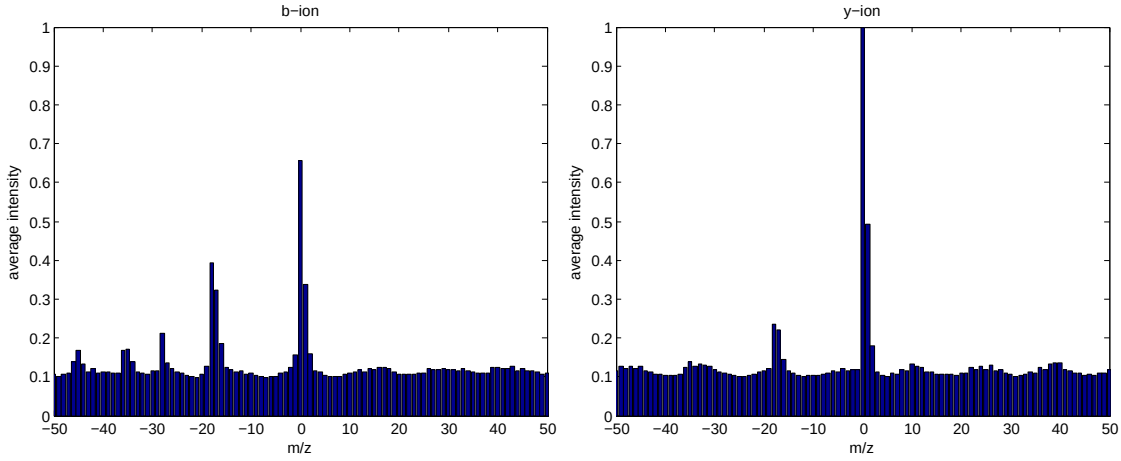


Figure 6.2.1: Average intensity near b-ion and y-ion peaks from high-confidence ($q < 0.01$) PSMs in the worm-01 dataset. We see strong signals at m/z=0 (central peak), m/z=+1 (+1 isotope), m/z=-17 (NH_3 loss), m/z=-18 (H_2O loss) and m/z=-28 (CO loss for b-ion only).

finding alternative explanations of the theoretical peak. This is a natural diminishing returns property, and can be described as $f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$, $\forall A \subseteq B \subset E$ and $e \notin B$. This is equivalent to the definition of submodular functions.

In general, even approximately maximizing an arbitrary set function is hopelessly intractable since it at least costs $O(2^{|E|})$, exponential in the number of set of edges. However, maximizing a monotonically non-decreasing submodular function subject to cardinality constraints can be approximately solved by a simple and efficient greedy algorithm with a $1 - \frac{1}{e}$ guarantee [38]. The same algorithm maximizes said function subject to intersection of two matroid constraints (described below) with a 1/3 guarantee [95]. Hence, submodular functions are both natural for the problem of generalized matching for PSM scores, but also allow efficient algorithms to obtain approximate optima of high quality.

There are rich classes of submodular set functions that one might choose from. We have developed a family of such functions, as described below, that are uniquely suited for PSM scoring and that allow for a rich and powerful relationship to exist between a peptide and its

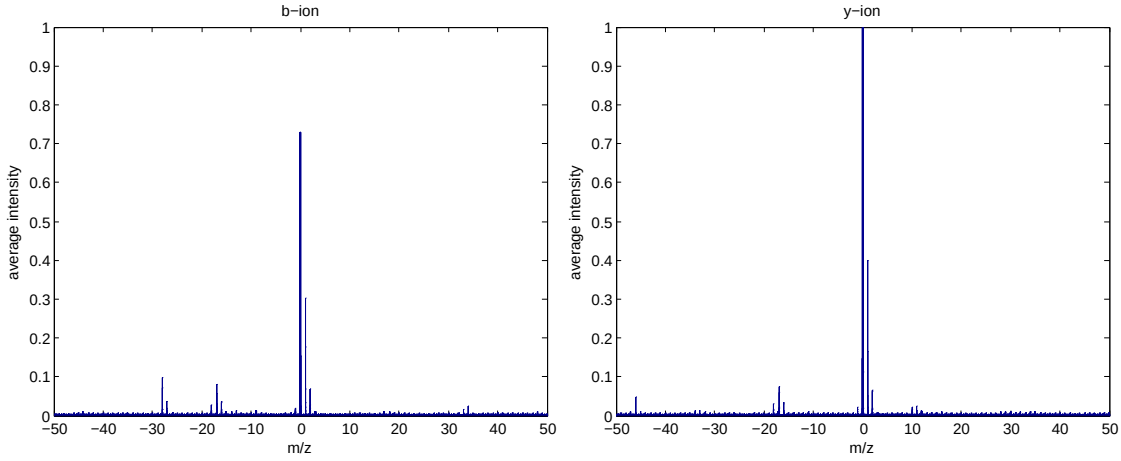


Figure 6.2.2: Average intensity near b-ion and y-ion from high-confidence ($q < 0.01$) PSMs in the *Plasmodium* TMT-10 dataset. We see strong signals at m/z=0 (central peak), m/z=+1 (+1 isotope), m/z=-17 (NH_3 loss), m/z=-18 (H_2O loss) and m/z=-28 (CO loss for b-ion only). The intensities in this figure are also used for the empirical weights $w(\{e_v, e_u\})$ for high resolution data for both *Plasmodium* and human. For each v and v, we compute the mass-to-charge ratio difference $m_v - m_u$ and then use the corresponding peak intensity.

observed spectrum. In the following two subsections, we describe two submodular functions f_1 and f_2, that each capture an important part of PSM scoring. Taking a convex combination of such functions preserves submodularity (e.g., $f = f_1 + f_2$ is submodular when the f_i's are submodular). So our final scoring method is a mixture of two functions. This is discussed next.

6.3.1 Matching one observed peak to multiple theoretical peaks

In general, we do not want an observed peak to be over-explained and hence over-valued by multiple fragment ions. When an observed peak is accounted for on the left side, if it is matched again, then the second match should not be given as much credit as when the second match is considered alone. This diminishing returns property is exactly modeled

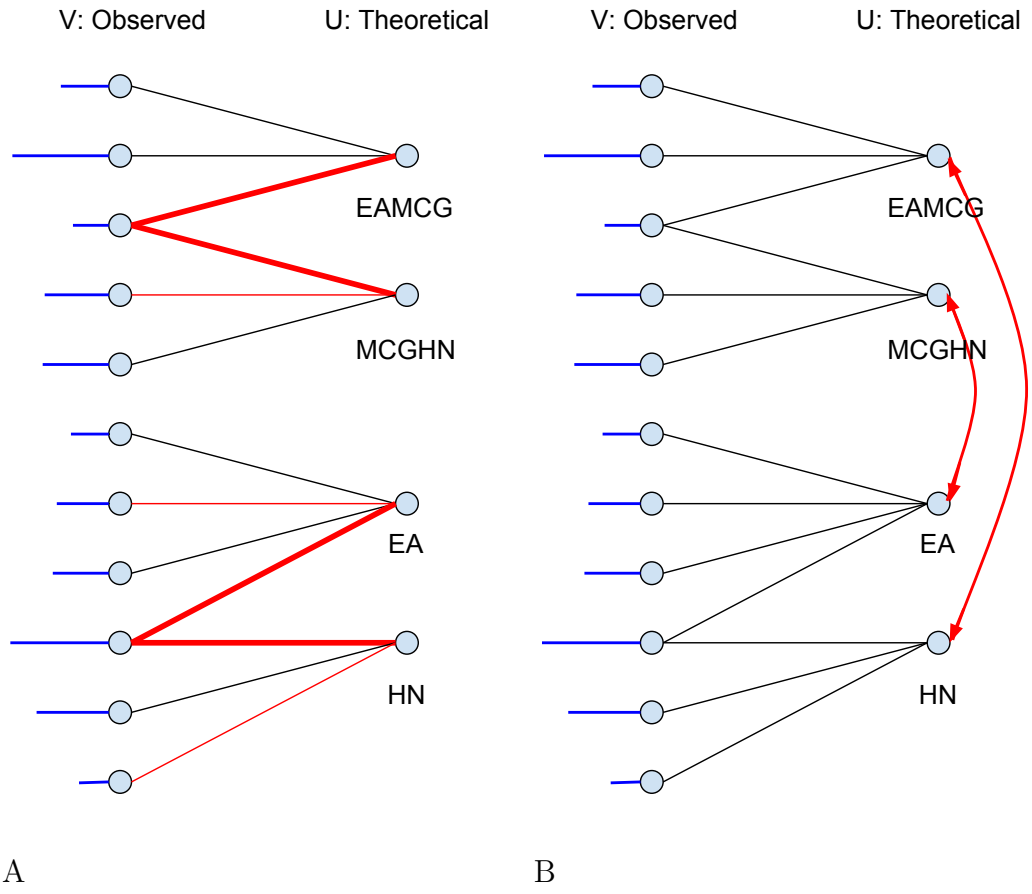


Figure 6.3.1: **Illustration of submodular functions.** V is the set of all peaks in an observed spectrum, with blue horizontal lines representing the observed spectrum intensities. U is the set of all fragment ions for a given theoretical spectrum. Edge (v, u) with $v \in V$ and $u \in U$ if v might possibly be explained by u with an associated non-negative weight. E is set of all edges. (A) Two theoretical peaks match the same observed peak. The submodular function assigns a score intermediate between the max and the sum of the two edge scores. (B) Two complementary theoretical peaks match to different observed peaks. The submodular function assigns a "bonus" for this complementarity.

by submodularity since if e_i is the i^{th} match, then $f(e_i|e_1, \ldots, e_{i-1}) \leq f(e_i)$ whenever f is a submodular function. Here we assume that $f(\emptyset) = 0$. This relationship is depicted in Figure 6.3.1A and can be represented using the following function:

$$f_1(A) = \sum_{v \in V} g_1 \left(\sum_{e \in A \cap \delta v} w(e) \right), \tag{6.1}$$

where $g_1(x) : \mathbb{R}^+$ to $\mathbb{R}$ is a monotonically non-decreasing concave function and $\delta v \subseteq E$ are the edges incident to node $v \in V$. The function $f_1(A)$ therefore provides submodularity on the edges grouped by the observed node due to the use of δv. $f_1(A)$ is a submodular function since it is a sum of monotonically non-decreasing concave functions composed with non-negative modular set functions [113]. Note that we have flexibility in the choice of $g_1(x)$, and are allowed to model this interaction in diverse ways using different $g_1(x)$ functions. In our experiments we use $g_1(x) = \beta \log(1 + \beta^{-1}x)$, where β is a parameter. We also experimented with $g_1 = x^\alpha$ for $\alpha = 1/2$, $1/3$ and $1/4$, but this class of function yielded worse empirical results (data not shown).

6.3.2 Scoring complementary pairs of matched observed peaks

In a spectrum produced by a well fragmented peptide, b-ions and corresponding y-ions always appear in pairs. Therefore, a score function should ideally provide a boost for a b-ion v_b if its corresponding y-ion $v_{\bar{b}}$ is also present (Figure 6.3.1B). This leads to a relationship of the inequality, $f(a_b \cup a_{\bar{b}}) > f(a_b) + f(a_{\bar{b}})$, where $a_b \in \delta v_b$ and $a_{\bar{b}} \in \delta v_{\bar{b}}$. Unfortunately, this relationship is supermodular (essentially a negative submodular), which is a much harder or impossible to optimize with mathematical worst case guarantees. Therefore, we use a modular function to express the inherent complementarity between b- and corresponding y-ions. Such functions are as close to supermodular as possible while still being submodular. Hence, relationships among edges that are naturally supermodular use a modular function; relationships that are naturally modular use a weakly submodular function, and relationships that are naturally submodular use a strong submodular function. The "strength" of the submodularity, in this context, corresponds precisely to the curvature (magnitude of the

second derivative) of the concave functions that are employed Given a concave over modular function $f(A) = g(m(A))$, we define curvature, in the current context, as $|\frac{d^2g(x)}{dx^2}|_{x=m(V)}$. For PSM scoring, we therefore use the following function, which acts as our submodular surrogate for a supermodular relationship between corresponding ion pairs:

$$f_2(A) = \gamma_2 \sum_{i \in U_{\text{b-ion}}} \Big[\sqrt{m\Big(A \cap \Big(E_i \cup \sum_{j \neq i} E_{\bar{j}}\Big)\Big)} + \sqrt{m\left(A \cap E_{\bar{i}}\right)} \Big] \tag{6.2}$$

where $U_{\text{b-ion}}$ is the set of b-ions for a given theoretical spectrum, and if i is an b-ion then $\bar{i}$ is the corresponding y-ion, and vice versa. (We say that $\bar{i}$ is the co-ion of i, and any $j \neq i$, $\bar{j}$ is a "non-co" ion.) The function $m(A) = \sum_{e \in A} w_e$ is a modular weight function. The coefficient $\gamma_2 = \sqrt{\sum_{e \in E} w(e)}$ scales the function to be combined with the other f_i's. Hence, we see that f_2 maximally credits any edge sets in A that correspond to an ion and its co-ion (since they are in different components in each term of the sum), whereas any edges that do not have this complementary (i.e., an ion and its non-co ions) are discounted if they are jointly selected within A.

6.3.3 Combination of submodular functions

We can use a weighted sum of the above two submodular functions as the evaluation function. The magnitude of the weight is important, because we do not want one term to dominate the other. For example, if f_1 is naturally larger than f_2 (i.e. $f_1(A) \gg f_2(A)$ for all $A \subseteq E$), then when maximizing the sum $f_1 + f_2$, the solution will focus primarily on f_1 with little consideration of f_2. To avoid this problem, we use the following combination:

$$f(A) = [\lambda_{\text{cal}} f_1(E) + (1 - \lambda_{\text{cal}}) f_2(E)] \Big[\lambda_{\text{mch}} \tfrac{f_1(A)}{f_1(E)} + (1 - \lambda_{\text{mch}}) \tfrac{f_2(A)}{f_2(E)} \Big]$$

where $0 \leq \lambda_{\text{cal}}, \lambda_{\text{mch}} \leq 1$. Inside the brackets, we use a convex mixture of normalized functions so that $0 \leq f_1(A)/f_1(E) \leq 1$ is always comparable with $0 \leq f_2(A)/f_2(E) \leq 1$. The mixture, however, is still normalized to be in the range $[0, 1]$ so we then calibrate this result by multiplying by $\lambda_{\text{cal}} f_1(E) + (1 - \lambda_{\text{cal}}) f_2(E)$ which does not affect the optimization since

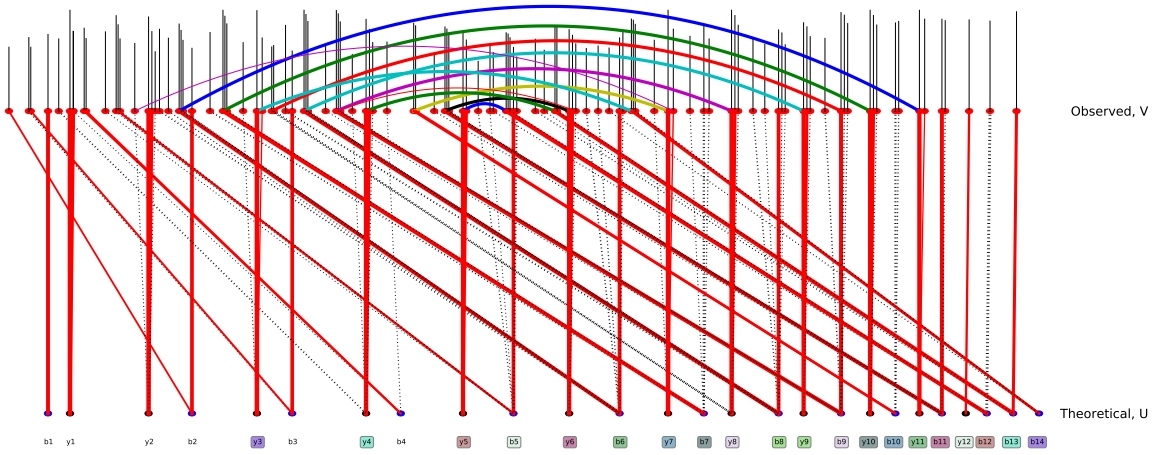


Figure 6.3.2: Visualization of the bipartite graph. The top nodes are observed peaks and the bottom ones are theoretical. Red solid lines are selected edges while dashed lines are unselected. On the top, corresponding observed peaks for b and y-ion pair are connected and thick if they are matched. On the bottom, matched b and y-ion pairs are marked in same color.

it is independent of A. Intuitively, this calibration ensures that PSM scores are comparable across different observed spectra (Figure 6.3.3).

To understand further why the calibration procedure is helpful, consider the expression for $f(A)$:

$$\begin{aligned} f(A) &= [\lambda_{\text{cal}} f_1(E) + (1-\lambda_{\text{cal}}) f_2(E)] \left[\lambda_{\text{mch}} \tfrac{f_1(A)}{f_1(E)} + (1-\lambda_{\text{mch}}) \tfrac{f_2(A)}{f_2(E)}\right] \\ &= \left(\lambda_{\text{cal}}\lambda_{\text{mch}} f_1(E) + \bar{\lambda}_{\text{cal}}\lambda_{\text{mch}} f_2(E)\right) \tfrac{f_1(A)}{f_1(E)} \\ &\quad + \left(\bar{\lambda}_{\text{cal}}\bar{\lambda}_{\text{mch}} f_2(E) + \lambda_{\text{cal}}\bar{\lambda}_{\text{mch}} f_1(E)\right) \tfrac{f_2(A)}{f_2(E)} \\ &= \alpha_1 \bar{f}_1(A) + \alpha_2 \bar{f}_2(A), \end{aligned}$$

where $\bar{\lambda}_i = 1 - \lambda_i$ for convenience, where $\bar{f}_i(A) = f_i(A)/f_i(E)$ for $i \in \{1, 2\}$ are now two $[0, 1]$-normalized (and hence compatible) submodular functions as mentioned above, and where α_i, $i \in \{1, 2\}$ are two coefficients. These coefficients, in fact, both mix and scale the functions, because $\alpha_1 = \lambda_{\text{mch}}(\lambda_{\text{cal}} f_1(E) + \bar{\lambda}_{\text{cal}} f_2(E))$ and $\alpha_2 = \bar{\lambda}_{\text{mch}}(\bar{\lambda}_{\text{cal}} \bar{f}_2(E) + \lambda_{\text{cal}} \bar{f}_1(E))$. Hence, we see how λ_{mch} influences the optimization while λ_{cal} selects a calibration score. These are hyperparameters over the algorithm and can be tuned on a development set. Figure 6.7.1 below shows the effects of these parameters.

6.4 Matroid Constraints

Now that we have an appropriate submodular evaluation function f, we next discuss how we produce the final PSM score $\mathfrak{s}$. One possible approach would use $\mathfrak{s} = f(E)$, thereby allowing *all* possible edges to comprise a score. This would be a poor choice, however, since the observed spectrum contains many noise peaks, and we have no oracle to decide whether a given peak is real signal or not. Although preprocessing steps can be very helpful to limit the influence of noise, it is impossible to fully eliminate all such cases. Moreover, the observed peaks of one fragment ion may happen to have the same m/z as other ions. In this case, the latter ion will have an unexpected co-ion, and we do not want to reward this in any way.

Thus, using all edges E will produce a final score that suffers from many false interactions. Fortunately, when we use a submodular function as described above, we expect the edges

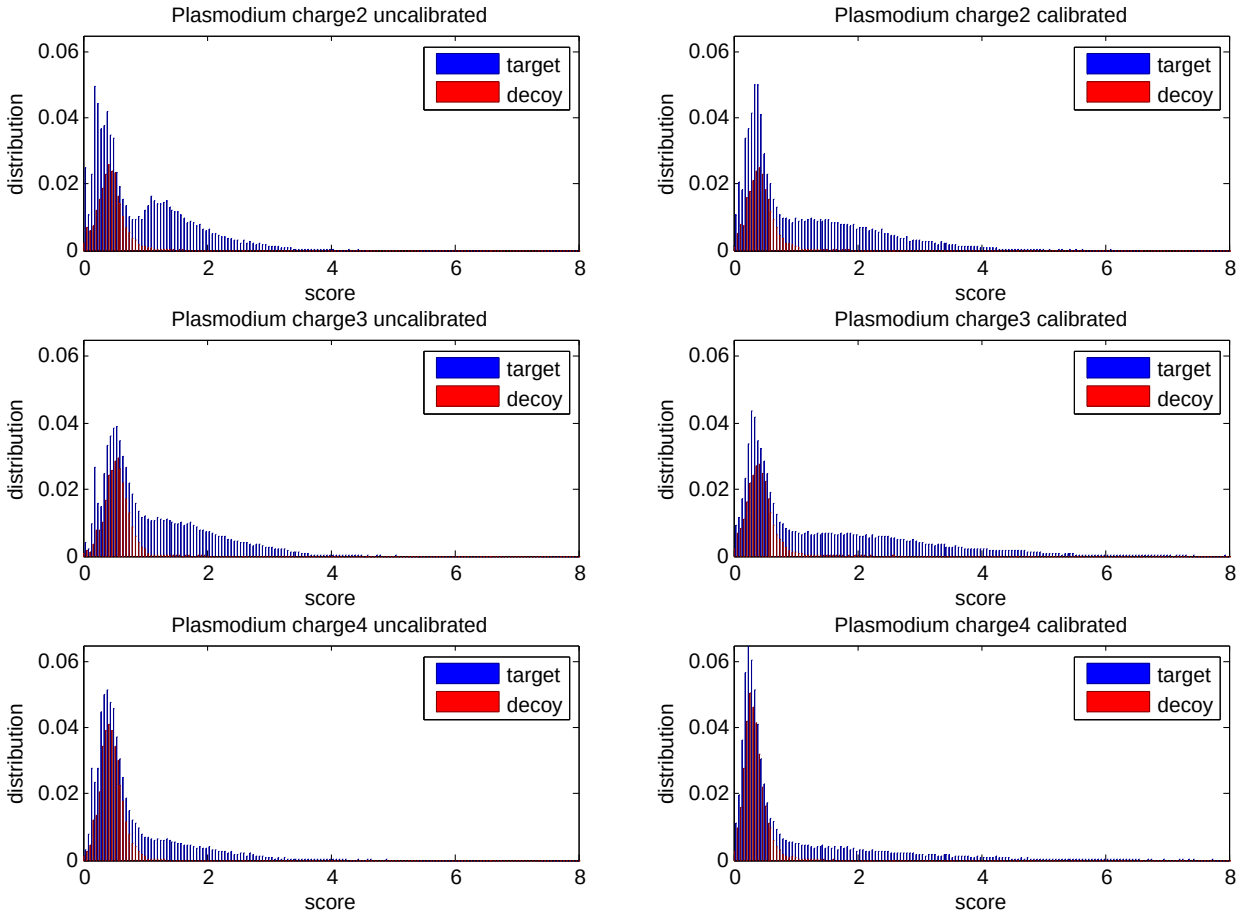


Figure 6.3.3: **Score calibration of SGM.** Each panel plots, for a single charge state, the SGM score distribution of top-scoring PSMs, separated into target and decoy distributions. Panels on the left are uncalibrated scores, and panels on the right are calibrated. In each plot, the x-axis is normalized so that the score threshold at $q = 0.01$ equals 1.

that do accurately explain the observed spectrum, in general, to have much greater weights than the noise edges. Moreover, limiting the set of edges being scored to satisfy certain constraints (which we call a "generalized matching") forces the edges comprising a score to compete with each other. This competition limits the ability of incorrect edges to artificially boost the score. Finally, for most false PSMs, even the best set of edges will not lead to a high score.

The next question is how to set up the right constraints. A simple way is to use standard bipartite matching constraints. This, as mentioned in Section 5.1, is problematic since every edge in such a matching can be incident to at most only one vertex. In practice, one observed peak might be explained by multiple fragment ions, and one fragment ion might truly explain multiple observed peaks. What we would like is a "generalized matching" where every vertex may be incident to more than one, but we still limit the total number of edges.

The intersection of two matroid constraints perfectly represents this generalized matching property. As mentioned in Section 1.3, Part I, a matroid is a pair $(E, \mathcal{I})$, where E is a finite set and $\mathcal{I}$ is a family of what are called "independent" subsets of E. An edge set solution A satisfies a matroid constraint if and only if $A \in \mathcal{I}$. A particular kind of matroid that is useful for our model is called a *partition matroid*. A partition matroid is based on a partition of E into ℓ disjoint subsets sets $\{E_i\}_{i=1}^{\ell}$, and ℓ non-negative integers $\{k_i\}_{i=1}^{\ell}$, where $\cup_{i=1}^{\ell} E_i = E$ and $E_i \cap E_j = \emptyset$ for $i \neq j$. A set $A \subseteq E$ is independent if and only if $|A \cap E_i| \leq k_i, \forall i$. Therefore, $\mathcal{I} = \{A | A \subseteq E, |A \cap E_i| \leq k_i, \forall i\}$. We have two natural partitions of edges E, namely $\{\delta v, v \in V\}$ and $\{\delta u, u \in U\}$ where, again, $\delta(v)$ is the set of edges incident to v and likewise for $\delta(u)$. Therefore, we define two partition matroids $\mathcal{M}_v = (E, \mathcal{I}_v)$ and $\mathcal{M}_u = (E, \mathcal{I}_u)$, where $\mathcal{I}_v = \{A | A \subseteq E, |A \cap \delta(v)| \leq k_v \, \forall v \in V\}$ and $\mathcal{I}_u = \{A | A \subset E, |A \cap \delta(u)| \leq k_u \, \forall u \in U\}$. We immediately see that a matching constraint for A is equivalent to $A \in \mathcal{I}_v \cap \mathcal{I}_u$, where $k_v = k_u = 1$ for all $v \in V$ and $u \in u$. A natural and immediate generalization of bipartite matching, moreover, is to set $k_v \geq 1$ and/or $k_u \geq 1$, where k_v (resp. k_u) corresponds to the maximum allowed number of incident edges to vertex v (resp. u) in any generalized matching. The values k_v and k_u are seen as parameters of

the constraint and correspond, for example, to allowing an observed peak to be explained by multiple fragment ions and one ion to explain multiple observed peaks in the PSM. In practice, we set $k_v = \infty$, because in the four datasets we studied, the maximum number of edges connected to v is only 2. Hence, there is no need for constraints on the observed side.

6.5 The Final Score Function

Our final score, for a given PSM, is computed as $\max_{A \in \mathcal{I}_v \cap \mathcal{I}_u} f(A)$, where $\mathcal{I}_v$ and $\mathcal{I}_u$ are the independent sets of two partition matroids with appropriate values of k_v and k_u. An exact solution is computationally intractable, but fortunately, as mentioned previously, an approximate solution can be easily and scalably calculated using a simple greedy algorithm with a mathematical guarantee of 1/3 [15].

Our score function can be regarded as a generalization of both SEQUEST and maximum bipartite matching. SEQUEST uses a dot product operation to calculate the score which is, in fact, equivalent to maximizing a modular set function subject to a particular bipartite matching constraint, namely, one where $|\delta v| = |\delta u| = 1$, i.e., where edges exist only between a theoretical peak and its corresponding observed peak. One difference between SEQUEST and such a bipartite score is that SEQUEST uses a background spectrum $\tilde{s}' = (\tilde{s} - \frac{1}{151}\sum_{\tau=-75}^{75} \tilde{s}_\tau)$ (i.e., the difference between a foreground and average background spectra, which can be negative) rather than just a non-negative foreground spectrum. In a submodular matching, however, we cannot use the background spectrum directly since it is not always non-negative, something that would violate both the monotonicity and submodularity of our objective $f(A)$ and render the efficient greedy algorithm mathematically vacuous. However, we can use the background information without resulting in negative values if we subtract a similar background factor after finding the maximum matching, as in: $\mathfrak{s} = \max_{A \in \mathcal{I}_v \cup \mathcal{I}_u} f(A) - \alpha\tau$, where α is a parameter and $\tau = \sum_{i \in U} \frac{1}{151} \sum_{j=-75}^{75} \tilde{s}(m/z_i + j)$ is a background factor. Subtracting τ from the score is not precisely the same as using the background spectrum in SEQUEST but has the same intended purpose and, as we show below, works well while preserving monotonicity, submodularity, and hence the mathematical

guarantees and applicability of the greedy algorithm.

6.6 Score Calibration

As described in Section 5.1, calibration is a critical step of a good score function. We say that a PSM score function is well calibrated if a score of x assigned to spectrum s_i has the same meaning or significance as a score of x assigned to spectrum s_j. During a database search, the top-scoring PSMs from different observed spectra are combined into a final, ranked list. The ranking will be reflective of the true qualities of the PSMs only if the scores of different observed spectra are comparable. In SGM, we calibrate each PSM's score by subtracting the average score of all the PSMs involving that spectrum, as follows

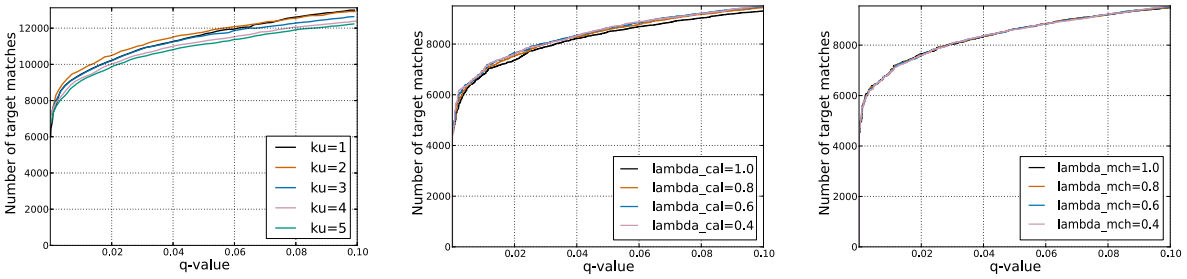
$$\mathfrak{s}^*(p, s) = \mathfrak{s}(p, s) - \frac{\sum_{p' \in P_s} \mathfrak{s}(p', s)}{|P_s|}, \tag{6.3}$$

where $\mathfrak{s}(p, s)$ is the non-calibrated score, and P_s is the set of candidate peptides associated with spectrum s.

The subtraction term is a constant with respect to each spectrum and does not affect which peptide is chosen. Rather, it only helps to produce a good overall ranking of top-scoring PSMs. Other search methods, such as MS-GF+ and the XCorr p-value, calibrate scores using dynamic programming. Using similar techniques for SGM scoring is left to future research, because SGM scoring is inherently non-linear, unlike SEQUEST which is a simple linear dot-product-based score.

6.7 Selection of Score Function Parameters

Our submodular functions use the hyperparameters λ_{cal}, λ_{mch} and $\{k_u\}_{u \in U}$. To select values for these hyperparameters, we performed an FDR-based evaluation with cases $k_u \in \{1, 2, 3, 4, 5\}$ when $\lambda_{\text{cal}} = \lambda_{\text{mch}} = 1.0$ on the data set worm-01-ch2 (described in Section 7.1). The results (Figure 6.7.1(a)) show that $k_u = 2$ yields the best performance. Next, we tested cases $\lambda_{\text{cal}} \in \{0.4, 0.6, 0.8, 1.0\}$ on yeast-01-ch2, while fixing $\lambda_{\text{mch}} = 1.0$. We then selected $\lambda_{\text{cal}} = 0.6$ based on these results (Figure 6.7.1(b)). Next, we tested $\lambda_{\text{mch}} \in \{0.4, 0.6, 0.8, 1.0\}$



k_u λ_{cal} λ_{mch}

Figure 6.7.1: FDR-based evaluation of SGM using different hyperparameters.

while fixing $\lambda_{\text{cal}} = 0.6$. The value $\lambda_{\text{mch}} = 0.8$ had the best performance (Figure 6.7.1(c)). For α, we calculate the average score of SGM,

$$\alpha_1 = \frac{\sum_{s \in S} \sum_{p \in P_s \cup D_s} \mathfrak{s}(p, s)}{\sum_{s \in S} |P_s \cup D_s|}$$

and the average foreground score of SEQUEST,

$$\alpha_2 = \frac{\sum_{s \in S} \sum_{p \in P_s \cup D_s} \langle p, s \rangle}{\sum_{s \in S} |P_s \cup D_s|},$$

where S is the set of all observed spectra and P_s and D_s are corresponding target and decoy sets of $s \in S$. The value α is then chosen to be $\frac{\alpha_2}{\alpha_1}$. Although the derivation of these parameters was empirical in our study, we hope in future work to develop strategies that can learn these automatically.

Ideally, these hyperparameters generalize across the different datasets. To test this, we tried all combinations of hyperparameters on one run from the yeast and worm data sets (yeast-01 and worm-01). The results (Figure 6.7.2) show that the parameters that work well on one dataset tend also to work well on the other dataset, implying that the parameters generalize well across datasets.

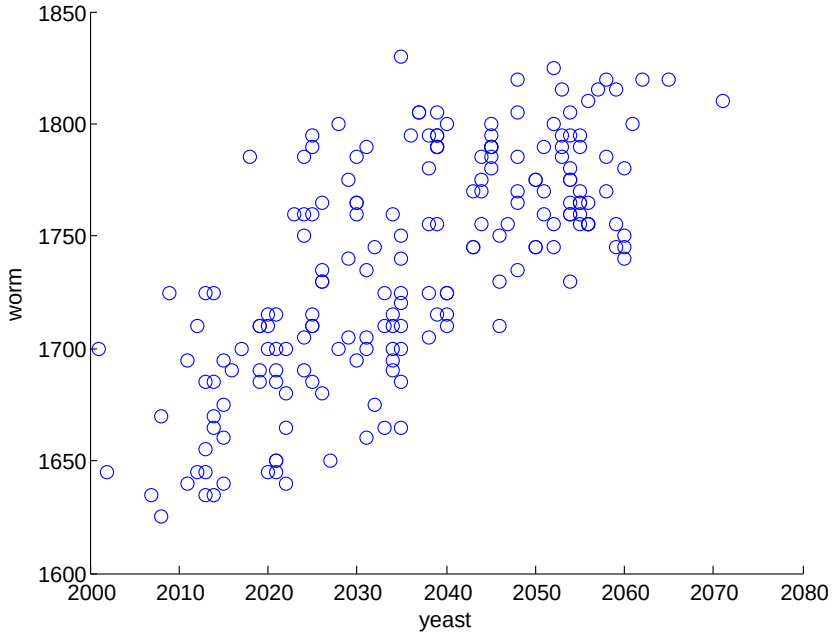


Figure 6.7.2: The plot shows the number of high-confidence PSMs ($q \leq 0.01$) obtained by SGM on the yeast data (x-axis) versus the worm data set (y-axis). Every point represents the performance of one combination of parameters.

Chapter 7

METHODS AND RESULTS

We use four different data sets to benchmark the performance of SGM relative to three state-of-the-art methods, and we employ several different quality measures to compare the results.

7.1 Data Sets

The yeast (*S. cerevisiae*) and worm (*C. elegans*) data sets were collected using tryptic digestion followed by acquisition using low-resolution precursor scans and low-resolution fragment ions. A total of 108,291 yeast and 68,252 worm spectra with charges ranging from 1+ to 3+ were collected. Each search was performed using a ±3.0 Da tolerance for selecting candidate peptides. Peptides were derived from proteins using tryptic cleavage rules without proline suppression and allowing no missed cleavages. A single fixed carbamidomethyl modification was included. Further details about these data sets, along with the corresponding protein databases, may be found in [65].

The malaria parasite *Plasmodium falciparum* was digested using Lys-C, labeled with an isobaric tandem mass tag (TMT) relabeling agent, and collected using high-resolution precursor scans and high-resolution fragment ions. The data set consists of 240,762 spectra with charges ranging from 2+ through 6+. Searches were run using a 50 ppm tolerance for selecting candidate peptides, a 0.03 Da fragment mass tolerance, a fixed carbamidomethyl modification, and a fixed TMT labeling modification of lysine and N-terminal amino acids. Further details may be found in [101].

The human dataset was digested using trypsin, labeled with an isobaric tandem mass tag (TMT) relabeling agent, and collected using high-resolution precursor scans and high-

resolution fragment ions. The data set consists of 1,133,534 spectra with charges ranging from 2+ through 6+. Searches were run using a 10 ppm tolerance for selecting candidate peptides, a 0.02 Da fragment mass tolerance, a fixed carbamidomethyl modification, and a fixed TMT labeling modification of lysine and N-terminal amino acids. Further details may be found in [137].

7.2 Database Search Methods

We compare SGM with four state-of-art methods: SEQUEST, MS-GF+ [71], XCorr p-value [50] and Mascot[17]. In Section 5.3, we have already described SEQUEST and its XCorr score function. In the experiments, we use a re-implementation of SEQUEST called "Tide" [21], available in Crux version 2.1.16790 [100]. One problem with SEQUEST in practice, as demonstrated below, is that the raw XCorr score is poorly calibrated. The MS-GF+ [71] search engine uses an alternative score function and employs dynamic programming to exactly compute the score distribution over the universe of candidate peptides for a linear scoring function. This gives far better calibration. In the experiment, we use MS-GF+ version 9980, and PSMs are ranked by the "Evalue" score. The XCorr p-value [50] uses a similar dynamic programming approach to calibration, applied to the SEQUEST XCorr score. We also use Crux for XCorr p-value. For clarity, we refer to the two variants of SEQUEST as "XCorr" (for results based on ranking with the raw XCorr score) and "p-value" (for results based on ranking by the XCorr p-value). Mascot is another traditional searching method. Unlike Sequest, Mascot uses a probabilistic metric to measure the likelihood of observed spectra and candidate peptides. We use an online Mascot server (version 2.3.01).

To ensure a fair comparison, we use equivalent values for search settings for all search engines whenever possible. In general, we use the appropriate discretization of the fragment m/z axis for each given data set. The only exception is that, for technical reasons related to the dynamic programming procedure, the XCorr p-value can only be calculated using an m/z resolution of 1.0005079 Da. For Tide, MS-GF+ and Mascot, default search parameters are used, except that, to make a fair comparison, handling of isotope peak errors is turned

off in MS-GF+. (In the Appendix, we show that this option does not affect our conclusions.) Furthermore, to avoid variability in how proteins are digested to peptides and how "decoy" peptides (see Section 7.3) are generated, we use the same digested peptide database as input to all search algorithms. We create these databases by using the "tide-index" command in Crux, with "clip-nterm-methionine" set to "True".

7.3 Evaluation of Methods

We employ a widely used approach, target/decoy search [28], to assign confidence estimates to PSM scores. These confidence estimates allow us to compare the performance of different search engines, since there is no ground truth to measure accuracy. We create a "decoy" set by randomly permuting the (non-terminal) amino acids of each peptide in a "target" set, which is the real candidate peptide set. For each spectrum, all peptides in a database comprised of targets plus decoys are searched, and the single peptide with the highest score is selected. If more than one peptide has the highest value, then ties are broken randomly.

Two complementary metrics for comparing two algorithms using target/decoy search were considered. The first, simpler approach is the "target match percentage" (TMP), defined as the fraction of observed spectra for which the top-scoring match involves a target peptide. For a perfectly random score function, the TMP is expected to be $\sim$50%. The best possible TMP is 100%; however, this is not achievable in practice, because any real data set will contain spectra that cannot be identified, either because the corresponding generating peptide is not in the given peptide database or because the spectrum was generated by a non-peptide contaminant. These "foreign" spectra are expected to match targets and decoys with equal frequency. TMP is not a widely used performance measure; however, we employ it here because TMP provides a measure of the quality of a score function that is independent of a score function's calibration. This is because the TMP only compares scores for PSMs within one spectra. Hence, the distribution of PSM scores for spectrum A can be dramatically different from another spectrum B, but the TMP achieved by the score function can still be high.

Table 7.4.1: Target match percentage achieved by the four score functions on four data sets. In each row, the maximal value is shaded red.

Dataset	SGM	MS-GF+	p-value	XCorr
yeast	70.59	66.19	65.47	64.91
worm	82.83	77.59	77.39	76.43
Plasmodium	69.91	66.85	65.53	69.39
human	74.40	60.40	73.26	74.12

Additional steps are required to evaluate the calibration of a score function. After obtaining a ranked list of the top-scoring PSM for each spectrum, we set a score threshold and label every PSM scoring better than the threshold as "accepted." The false discovery rate at a given threshold can be estimated as FDR $= \frac{\text{number of accepted decoy PSMs}}{\text{total number of accepted PSMs}}$ [28]. In practice, we compute for each PSM its corresponding q-value, defined as the minimum FDR at which a PSM with that score is accepted [114]. Because many mass spectrometry studies report results using an FDR threshold of 1%, we sometimes report the number of target PSMs accepted at $q \leq 0.01$. To evaluate the performance of a search engine over a variety of q-value thresholds, we also plot the number of accepted target PSMs as a function of q-value threshold and compute the area under the plot from $0 \leq q \leq 0.1$. Because the FDR-based evaluation involves creating a ranked list of top-scoring PSMs from many different spectra, this metric requires good cross-spectrum calibration.

7.4 Comparison of Four Search Methods

We begin by computing the target match percentage of the four search methods—SGM, MS-GF+, p-value and XCorr—on the four data sets described in Section 7.1. In all four cases, SGM achieves the greatest TMP (Table 7.4.1). Each of these data sets consists of multiple mass spectrometry runs: 3, 3, 20, and 100, for the yeast, worm, *Plasmodium* and human data

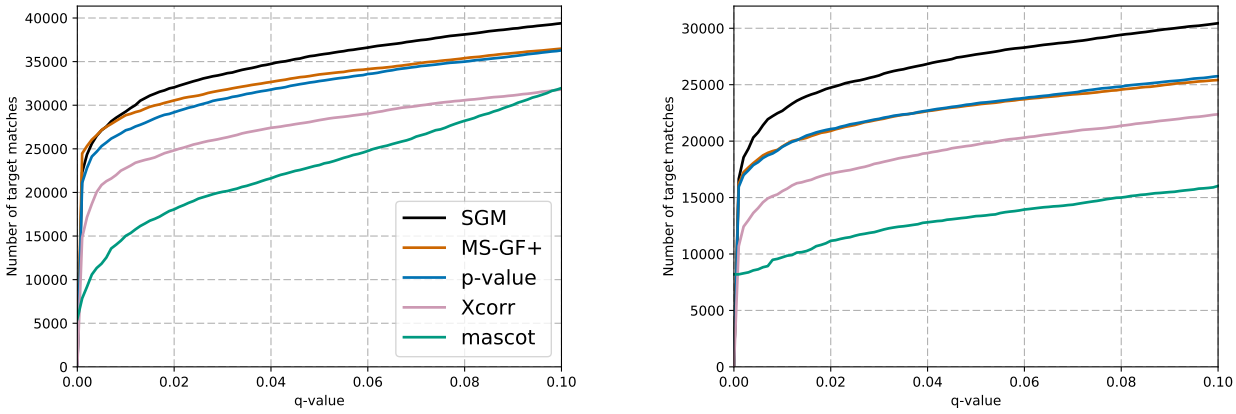
sets, respectively. Consequently, for the latter two data sets we were able to compute the TMP separately for each run and then use a Wilcoxon signed-rank test to identify statistically significant differences. This analysis (Figure 7.4.2) indicates that, for the malaria data set, SGM performs better than XCorr ($p = 0.11$), and for the human data set, SGM performs significantly better than all three competing methods ($p = 1.6 \times 10^{-5}$). The consistently good TMP performance of the SGM method on these diverse data sets indicates that, for each observed spectrum, this score function does a very good job of ranking the generating peptide above all other candidate peptides.

Next we evaluate the methods using false discovery rate estimation, thereby additionally taking into account the calibration of the scores. In practice, this evaluation is the most important, since it directly reflects how the end user will interpret the results of the search. The results (Figure 7.4.1) suggest that, once again, SGM performs better than MS-GF+, XCorr, XCorr p-value and Mascot for the yeast, worm and *Plasmodium* data sets. We quantified the performance of the first four methods by counting the number of accepted PSMs at a q-value threshold of 0.01, and we again used a Wilcoxon signed-rank test to estimate significant differences for the data sets comprised of many runs. This analysis shows that SGM significantly outperforms all three competing methods on both the *Plasmodium* and human data sets. The p-values relative to the second-ranked method, XCorr, are 0.0015 for *Plasmodium* and 0.0014 for the human data set (Figure 7.4.2).

7.5 Verifying the Utility of Submodularity

The SGM approach employs a combination of two submodular functions, each of which is designed to capture a particular property of high quality matches between spectra and peptides. To verify that the good results in Section 7.4 are indeed a reflection of these properties, we examined more closely the high-confidence identifications produced by SGM. For this analysis, we focus on a single, randomly selected run ("TMT10") from the *Plasmodium* data set.

First, we show that f_1 discourages choosing multiple edges incident to one observed peak.



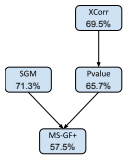
(A) Yeast

(B) Worm

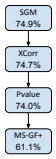
(C) *Plasmodium*

(D) Human

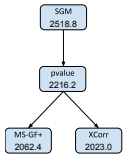
Figure 7.4.1: **FDR-based comparison of search methods.** Each panel plots, for a single data set and a variety of score functions, the number of spectra identified as a function of FDR threshold.



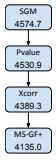
(A) *Plasmodium* TMP



(B) human TMP

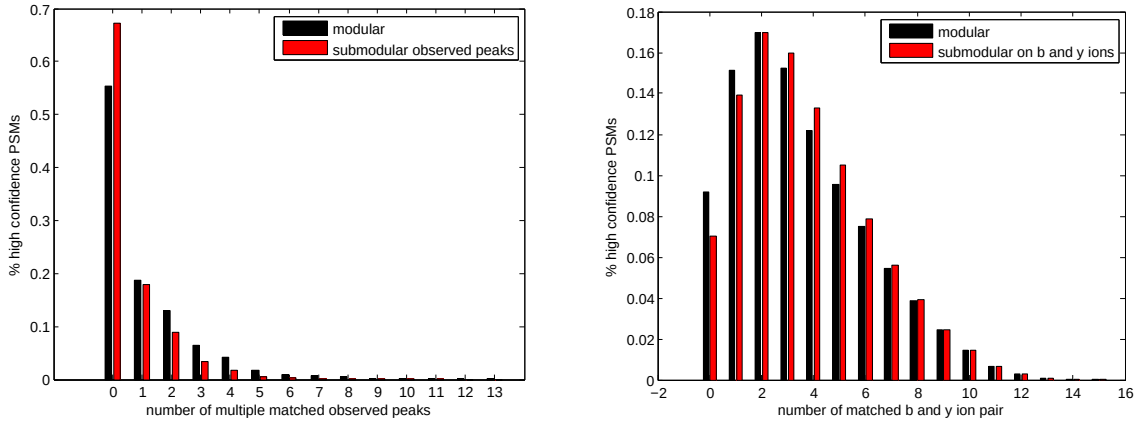


(C) *Plasmodium* accepted PSMs



(D) human accepted PSMs

Figure 7.4.2: **Statistical comparison of methods.** Each panel plots, for a single data set, the comparison between four methods in terms of the target match percentage or the number of targets PSMs accepted at $q < 0.01$. A directed edge from A to B means that method A's mean score is significantly larger ($p < 0.05$) than method B's mean score, according to a Wilcoxon signed-rank test. The numbers in the nodes are mean values.



A B

Figure 7.5.1: **PSM properties captured by the submodular function.** (A) The number of multiple matched observed peaks decreases when we use the submodular function f_1. (B) The number of multiple matched b- and y-ion pairs increases when we use the submodular function f_2.

To do so, we compare the number of multiply matched observed peaks in high confidence PSMs ($q \leq 0.01$) generated using two methods: a simple, modular approach versus using f_1. The distribution of the number of multiply matched peaks decreases when we use f_1 (Figure 7.5.1A), which implies that our submodular function discourages multiply matched observed peaks.

Second, we show that f_2 encourages choosing a b-ion if its corresponding y-ion is already chosen, and vice versa. As before, we compute the number of jointly matched b- and y-ion pairs among the high confidence PSMs, with and without inclusion of f_2. As expected, the number of matched b- and y-ion pairs increases when we use f_2 (Figure 7.5.1B), implying that our submodular function indeed encourages such matching.

7.6 Investigation of Empirical Weights

Our method is different from XCorr in two respects. First, we use empirically derived edge weights based on an analysis of the data (Section 6.2). Second, we generalize the dot product score to one based on a submodular generalized matching. We performed further experiments to ascertain which of these two changes are primarily responsible for the good results reported above.

In particular, we contrast our empirical weights with the "classical" weights employed by methods such as XCorr and MS-GF+. The methods use a fixed weights for each ion type. The classical weights used in the XCorr score are defined as $w'(\{v,u\}) = \delta_{m_v,m_u} + 0.2\sum_{l\in\{-17,-18,-28\}} \delta_{m_v+l,m_u}$, $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise. Hence, the classical weight is 1 if $m_v - m_u = 0$; 0.2 if $m_v - m_u \in \{-17,-18,-28\}$ and 0 otherwise.

To investigate the relative importance of SGM and the empirical weighting scheme, we analyze the contributions of these two components separately. We do the test on a single run ("TMT10") from the *Plasmodium* data set, evaluating performance using target-decoy q-values. We compare four different search methods: SGM with and without empirical weights, and XCorr with and without empirical weights. In this experiment (Figure 7.6.1), the combination of SGM with empirical weights achieves by far the best performance. While

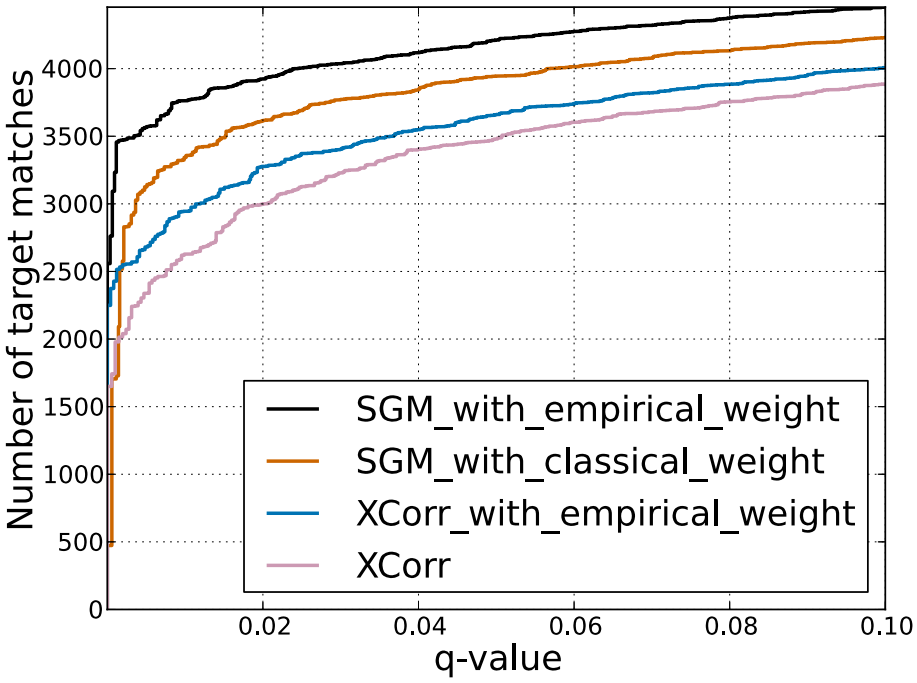


Figure 7.6.1: Evaluation of the SGM and XCorr score functions on a subset of the *Plasmodium* data set, with and without using the empirical weighting scheme.

SGM	MS-GF+	p-value	XCorr
8.48×10^{-2}s	8.99×10^{-2}s	6.89×10^{-2}s	$3.39 \times 10^{-3}s$

Table 7.7.1: Run time of four methods per spectrum on the *Plasmodium* TMT-10 data set.

the empirical weights help both XCorr and SGM, the SGM with the standard XCorr weights is still better than XCorr with our empirical weights. Hence, it is fair to say that the SGM process is the more important of the two. This is not surprising because the power of SGM, and the use of submodular functions, is that we can allow global long-range interaction amongst edge scores, something that is not possible with XCorr.

7.7 Running Time

To evaluate the running time of SGM relative to other search tools, we measured the wall clock time of four search methods on a single Intel Core 2 Quad Q9550 2.83GHz CPU on the *Plasmodium* TMT-10 dataset. The dataset consists of 8841 spectra, each with an average of 365 target peptides and an equal number of decoy peptides. This analysis (Table 7.7.1) shows that SGM has a comparable run time with MS-GF+ and the Tide p-value. The raw XCorr score calculation is extremely fast because it simply consists of calculating a dot product, with no explicit calibration procedure.

Chapter 8
CONCLUSION

In Part I, we have introduced a novel class of set functions: Arithmetic Combinations of Submodular and Supermodular functions, and the associated optimization problems. The class covers all classical arithmetic operations, *sums* $f_1 + g_1$, *divisions* f_1/f_2, f_1/g_1, g_1/f_1, g_1/g_2, *products* $f_1 * f_2$ and *p-norms* $f_1^p + f_2^p$. We also show these functions naturally appear in real-world applications. Submodular measures redundancies while supermodular measures complementarities. Therefore, the arithmetic combinations of these naturally measure both properties, and using different combinations allows different trade-offs, thus falling into different applications.

We propose classical or novel algorithms that solve these problems with performance guarantees in polynomial time. We also study the hardness of these optimization problems, and our proposed algorithm's performance guarantee has 0 or a small gap with hardness.

- In chapter 2, we study sum of suBmodular and suPermodular (**BP) maximization and propose supermodular curvature κ^g that measures the degree of supermodularity. We propose GreedMax and SemiGrad have performance guarantee of $\frac{1}{\kappa_f}\left[1 - e^{-\kappa_f(1-\kappa^g)}\right]$ and $\frac{1-\kappa^g}{(1-\kappa^g)\kappa_f+p}$ for cardinality constraint and p matroid constraints, respectively. We also calculate hardness is $1 - \kappa^g + \epsilon$ and $(1 - \kappa^g)O(\frac{\ln p}{p})$, respectively.**

- **In chapter 3, we study Ratio function minimization. For the submodular over supermodular case, we propose a $1+\epsilon$ linear search algorithm; for modular over modular case, we propose GreedRatio can find the exact optimal solution; for submodular over submodular case, we propose GreedRatio**

has performance guarantee $\frac{1}{1-e^{(\kappa_f-1)}}$, and EllipsoidApprox has performance guarantee $O(\sqrt{n}\log n)$, where hardness is $O(\sqrt{n})$ in this case.

- **In chapter 4, we study generalized mean optimization, including Geometric Mean (Product), Generalized Mean (p-norm) and Harmonic Mean (product over sum). We propose a linear search algorithm that can solve the problems with at most $(1+\epsilon)^p$ or $(1-\epsilon)^p$ reduction (multiplicative) from the guarantees of the reduced problems, for example, submodular optimizations, which match the known hardness result of the problem.**

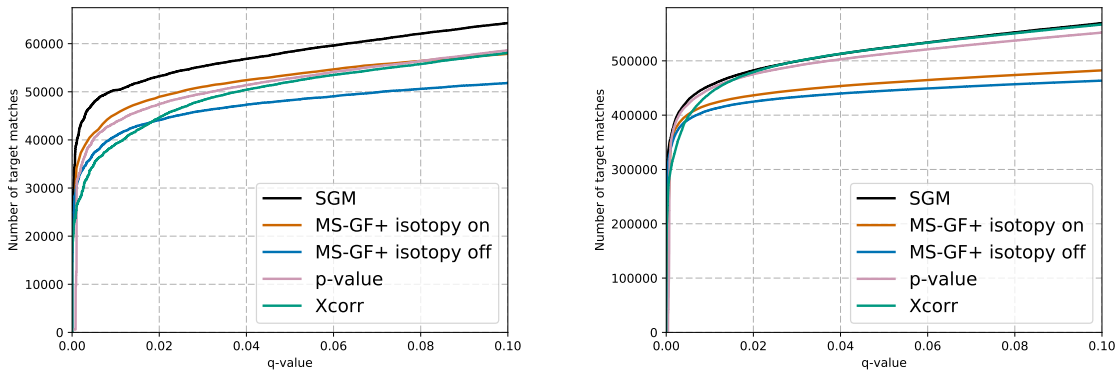
In Part II, we have introduced a novel class of score functions for use in tandem mass spectrometry database search. A key advantage of our SGM is that we can model many PSM properties, including long-range interactions among peaks in an observed spectrum, using a rich and powerful framework, namely that of submodularity and various matroid constraints. An additional advantage is that our model runs fast since we may use a simple accelerated greedy algorithm to find the maximum value of the submodular function with a mathematical quality guarantee of $\frac{1}{3}$. We show that our approach achieves statistically significant improvements in performance relative to several state-of-the-art methods according to two different evaluation metrics.

In SGM, we use three hyperparameters, chosen from a grid of possible values (Section 6.7). Our empirical analysis suggests that these hyperparameters generalize well across datasets. Therefore, we do not need to re-tune these hyperparameters to different datasets.

In future studies, we will explore other submodular functions in an attempt to further improve performance. For example, we can explore algorithms that can learn submodular functions and parameters from training data [85, 121]. We can will also consider other generalizations of our framework to better model the process of spectrum generation.

Appendices

MSGF+ isotopic peak option. In Section 7.2, to ensure a fair comparison, the options designed to handle errors in precursor isotopic peak selection are turned off for all methods. For completeness, we also include performance of MS-GF+ with isotope peak errors on (Figure 8..1). The results show that MS-GF+ benefits from appropriate handling of isotope peak errors but still does not outperform SGM.



(A) *Plasmodium* (B) Human

Figure 8..1: **FDR-based comparison of MSGF+ with isotopic error handling on and off.** Each panel plots, for a single data set and a variety of score functions, the number of spectra identified as a function of FDR threshold.

BIBLIOGRAPHY

[1] Alper Atamtürk and Vishnu Narayanan. The submodular knapsack polytope. *Discrete Optimization*, 2009.

[2] Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. *In SODA*, 2014.

[3] Wenruo Bai and Jeff Bilmes. Greed is still good: maximizing monotone submodular+ supermodular (bp) functions. In *International Conference on Machine Learning*, pages 304–313. PMLR, 2018.

[4] Wenruo Bai, Jeffrey Bilmes, and William S. Noble. Bipartite matching generalizations for peptide identification in tandem mass spectrometry. In *7th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM BCB)*, ACM SIGBio, Seattle, WA, October 2016. ACM, ACM SIGBio.

[5] Wenruo Bai, Rishabh Iyer, Kai Wei, and Jeff Bilmes. Algorithms for optimizing the ratio of submodular functions. In *International Conference on Machine Learning (ICML), New York, USA*, 2016.

[6] J?rgen Bang-Jensen, Gregory Gutin, and Anders Yeo. When the greedy algorithm fails. *Discrete Optimization*, 1(2):121 – 127, 2004. ISSN 1572-5286. doi: https://doi.org/10.1016/j.disopt.2004.03.007. URL `http://www.sciencedirect.com/science/article/pii/S1572528604000222`.

[7] Witold Bednorz, editor. *Advances in greedy algorithms*, volume 14. Wienna: I-Tech Education and Publishing KG, 2008.

[8] Andrew An Bian, Joachim M Buhmann, Andreas Krause, and Sebastian Tschiatschek. Guarantees for greedy maximization of non-submodular functions with applications. *arXiv preprint arXiv:1703.02100*, 2017.

[9] Jeffrey Bilmes and Wenruo Bai. Deep Submodular Functions. *Arxiv*, abs/1701.08939, Jan 2017. URL `http://arxiv.org/abs/1701.08939`.

[10] Allan Borodin, Dai Le, and Yuli Ye. Proportionally (formerly weakly) submodular functions. *CoRR*, abs/1401.6697, 2014. URL `http://arxiv.org/abs/1401.6697`. `http://www.cs.toronto.edu/~bor/Papers/proportional-talg-submit.pdf`.

[11] Niv Buchbinder and Moran Feldman. Deterministic algorithms for submodular maximization problems. *ACM Transactions on Algorithms (TALG)*, 14(3):1–20, 2018.

[12] Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *FOCS*, 2012.

[13] Niv Buchbinder, Moran Feldman, Joseph Seffi Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1433–1452. SIAM, 2014.

[14] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 182–196. Springer, 2007.

[15] M. Conforti and G. Cornuejols. Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discrete Applied Mathematics*, 7(3):251–274, 1984.

[16] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[17] John S Cottrell and U London. Probability-based protein identification by searching sequence databases using mass spectrometry data. *electrophoresis*, 20(18):3551–3567, 1999.

[18] R. Craig and R. C. Beavis. Tandem: matching proteins with tandem mass spectra. *Bioinformatics*, 20:1466–1467, 2004.

[19] W.H. Cunningham. On submodular function minimization. *Combinatorica*, 5(3):185–192, 1985.

[20] A. Das and D. Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In *ICML*, 2011.

[21] B. Diament and W. S. Noble. Faster SEQUEST searching for peptide identification from tandem mass spectra. *Journal of Proteome Research*, 10(9):3871–3879, 2011. PMC3166376.

[22] Brenda L Dietrich and Alan J Hoffman. On greedy algorithms, partially ordered sets, and submodular functions. *IBM Journal of Research and Development*, 47(1):25–30, 2003.

[23] Brian W Dolhansky and Jeff A Bilmes. Deep submodular functions: Definitions and learning. In *Advances in Neural Information Processing Systems*, pages 3396–3404, 2016.

[24] Andreas WM Dress and Walter Wenzel. Valuated matroids: A new look at the greedy algorithm. *Applied Mathematics Letters*, 3(2):33–35, 1990.

[25] FDJ Dunstan and DJA Welsh. A greedy algorithm for solving a certain class of linear programmes. *Mathematical Programming*, 5(1):338–353, 1973.

[26] J. Edmonds. Submodular functions, matroids and certain polyhedra. *Combinatorial structures and their Applications*, 1970.

[27] Jack Edmonds. Matroids and the greedy algorithm. *Mathematical programming*, 1(1): 127–136, 1971.

[28] J. E. Elias and S. P. Gygi. Target-decoy search strategy for increased confidence in large-scale protein identifications by mass spectrometry. *Nature Methods*, 4(3):207–214, 2007.

[29] J. K. Eng, A. L. McCormack, and J. R. Yates, III. An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *Journal of the American Society for Mass Spectrometry*, 5:976–989, 1994.

[30] Ulrich Faigle. The greedy algorithm for partially ordered sets. *Discrete Mathematics*, 28(2):153–159, 1979.

[31] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 1998.

[32] Uriel Feige and Rani Izsak. Welfare maximization and the supermodular degree. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 247–256. ACM, 2013.

[33] Uriel Feige, Vahab Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM J. COMPUT.*, 40(4):1133–1155, 2011.

[34] M. Feldman, J.S. Naor, and R. Schwartz. A unified continuous greedy algorithm for submodular maximization. In *FOCS*, 2011.

[35] Moran Feldman and Rani Izsak. Constrained monotone function maximization and the supermodular degree. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, pages 160–175, 2014. doi: 10.4230/LIPIcs.APPROX-RANDOM.2014.160. URL `https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2014.160`.

[36] Moran Feldman and Rani Izsak. Constrained monotone function maximization and the supermodular degree. *arXiv preprint arXiv:1407.6328*, 2014.

[37] M.L. Fisher, G.L. Nemhauser, and L.A. Wolsey. An analysis of approximations for maximizing submodular set functions—ii. *Polyhedral combinatorics*, pages 73–87, 1978.

[38] M.L. Fisher, G.L. Nemhauser, and L.A. Wolsey. An analysis of approximations for maximizing submodular set functions—ii. *Polyhedral combinatorics*, pages 73–87, 1978.

[39] S. Fujishige. *Submodular functions and optimization*, volume 58. Elsevier Science, 2005.

[40] Satoru Fujishige, Takumi Hayashi, and Shigueo Isotani. *The minimum-norm-point algorithm applied to submodular function minimization and linear programming*. Citeseer, 2006.

[41] Toshihiro Fujito. Approximation algorithms for submodular set cover with applications. *IEICE Transactions on Information and Systems*, 83(3):480–487, 2000.

[42] L. Y. Geer, S. P. Markey, J. A. Kowalak, L. Wagner, M. Xu, D. M. Maynard, X. Yang, W. Shi, and S. H. Bryant. Open mass spectrometry search algorithm. *Journal of Proteome Research*, 3:958–964, 2004. OMSSA.

[43] Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1098–1116. SIAM, 2011.

[44] Michel X Goemans, Nicholas JA Harvey, Satoru Iwata, and Vahab Mirrokni. Approximating submodular functions everywhere. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 535–544. Society for Industrial and Applied Mathematics, 2009.

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

[46] Andrew Guillory and Jeff A. Bilmes. Simultaneous learning and covering with adversarial noise. In *International Conference on Machine Learning (ICML)*, Bellevue, Washington, 2011.

[47] Elad Hazan, Shmuel Safra, and Oded Schwartz. On the complexity of approximating k-set packing. *computational complexity*, 15(1):20–39, 2006.

[48] Steven CH Hoi, Rong Jin, Jianke Zhu, and Michael R Lyu. Batch mode active learning and its application to medical image classification. In *ICML*, 2006.

[49] Thibaut Horel and Yaron Singer. Maximization of approximately submodular functions. In *Advances In Neural Information Processing Systems*, pages 3045–3053, 2016.

[50] J Jeffry Howbert and William S Noble. Computing exact p-values for a cross-correlation shotgun proteomics score function. *Molecular & Cellular Proteomics*, pages mcp–O113, 2014.

[51] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[52] Victor P Il'ev. An approximation guarantee of the greedy descent algorithm for minimizing a supermodular set function. *Discrete Applied Mathematics*, 114(1):131–146, 2001.

[53] S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM (JACM)*, 48(4):761–777, 2001.

[54] R. Iyer and J. Bilmes. Algorithms for approximate minimization of the difference between submodular functions, with applications. *In UAI*, 2012.

[55] R. Iyer and J. Bilmes. Submodular Optimization with Submodular Cover and Submodular Knapsack Constraints. In *NIPS*, 2013.

[56] R. Iyer, S. Jegelka, and J. Bilmes. Curvature and Optimal Algorithms for Learning and Minimizing Submodular Functions . In *Neural Information Processing Society (NIPS)*, 2013.

[57] Rishabh Iyer and Jeff Bilmes. Algorithms for approximate minimization of the difference between submodular functions, with applications. *arXiv preprint arXiv:1207.0560*, 2012.

[58] Rishabh Iyer and Jeff A Bilmes. Submodular-bregman and the lovász-bregman divergences with applications. In *Advances in Neural Information Processing Systems*, pages 2933–2941, 2012.

[59] Rishabh Iyer and Jeffrey Bilmes. Submodular point processes with applications to machine learning. In *Artificial Intelligence and Statistics*, pages 388–397. PMLR, 2015.

[60] Rishabh Iyer, Stefanie Jegelka, and Jeff Bilmes. Fast semidifferential-based submodular function optimization. In *International Conference on Machine Learning*, pages 855–863, 2013.

[61] Rishabh K Iyer and Jeff A Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *Advances in Neural Information Processing Systems*, pages 2436–2444, 2013.

[62] S. Jegelka and J. A. Bilmes. Submodularity beyond submodular energies: coupling edges in graph cuts. In *CVPR*, 2011.

[63] Sai Ji, Dachuan Xu, Min Li, Yishui Wang, and Dongmei Zhang. Stochastic greedy algorithms for maximizing constrained submodular+ supermodular functions. *Concurrency and Computation: Practice and Experience*, page e6575, 2021.

[64] Haotian Jiang. Minimizing convex functions with integral minimizers. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 976–985. SIAM, 2021.

[65] Lukas Käll, Jesse D Canterbury, Jason Weston, William Stafford Noble, and Michael J MacCoss. Semi-supervised learning for peptide identification from shotgun proteomics datasets. *Nature methods*, 4(11):923–925, 2007.

[66] Brad Karp and Hsiang-Tsung Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254. ACM, 2000.

[67] Yoshinobu Kawahara and Takashi Washio. Prismatic algorithm for discrete dc programming problems. In *NIPS*, 2011.

[68] Yoshinobu Kawahara, Kiyohito Nagano, and Yoshio Okamoto. Submodular fractional programming for balanced clustering. *Pattern Recognition Letters*, 32(2):235–243, 2011.

[69] Yulia Kempner, Vadim E Levit, and Ilya Muchnik. Quasi-concave functions and greedy algorithms. In *Greedy Algorithms*. InTech, 2008.

[70] Arif Khan, Alex Pothen, Md Mostofa Ali Patwary, Nadathur Rajagopalan Satish, Narayanan Sundaram, Fredrik Manne, Mahantesh Halappanavar, and Pradeep Dubey. Efficient approximation algorithms for weighted b-matching. *SIAM Journal on Scientific Computing*, 38(5):S593–S619, 2016.

[71] Sangtae Kim and Pavel A Pevzner. MS-GF+ makes progress towards a universal database search tool for proteomics. *Nature Communications*, 5, 2014.

[72] Bernhard Korte, László Lovász, and Rainer Schrader. *Greedoids*, volume 4. Springer Science & Business Media, 2012.

[73] A. Krause and C. Guestrin. A note on the budgeted maximization of submodular functions, 2005. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.130.3314`.

[74] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *JMLR*, 9:235–284, 2008.

[75] Andreas Krause, H Brendan McMahan, Carlos Guestrin, and Anupam Gupta. Robust submodular observation selection. *Journal of Machine Learning Research*, 9(12), 2008.

[76] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[77] Alan Kuhnle. Interlaced greedy algorithm for maximization of submodular functions in nearly linear time. *arXiv preprint arXiv:1902.06179*, 2019.

[78] A. Kulik, H. Shachnai, and T. Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *SODA*, 2009.

[79] Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Mathematics of Operations Research*, 35 (4):795–806, 2010.

[80] H. Lin and J. Bilmes. Optimal selection of limited vocabulary speech corpora. In *Interspeech*, 2011.

[81] H. Lin and J. Bilmes. Learning mixtures of submodular shells with application to document summarization. In *UAI*, 2012.

[82] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *The 49th Meeting of the Assoc. for Comp. Ling. Human Lang. Technologies (ACL/HLT-2011)*, Portland, OR, June 2011.

[83] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics, 2011.

[84] Hui Lin and Jeff Bilmes. Word alignment via submodular maximization over matroids. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 170–175. Association for Computational Linguistics, 2011.

[85] Hui Lin and Jeff Bilmes. Learning mixtures of submodular shells with application to document summarization. In *Uncertainty in Artificial Intelligence (UAI)*, Catalina Island, USA, July 2012. AUAI.

[86] Yajing Liu, Edwin KP Chong, and Ali Pezeshki. Improved bounds for the greedy strategy in optimization problems with curvatures. *arXiv preprint arXiv:1705.04195*, 2017.

[87] Yuzong Liu, Kai Wei, Katrin Kirchhoff, Yisong Song, and Jeff Bilmes. Submodular feature selection for high-dimensional acoustic score spaces. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7184–7188. IEEE, 2013.

[88] Mokshay Madiman and Prasad Tetali. Information inequalities for joint distributions, with interpretations and applications. *IEEE Transactions on Information Theory*, 56 (6):2699–2713, 2010.

[89] I Dan Melamed. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249, 2000.

[90] M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques*, pages 234–243, 1978.

[91] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *International Conference on Machine Learning*, pages 1358–1367. PMLR, 2016.

[92] Adhyyan Narang, Omid Sadeghi, Lillian J Ratliff, Maryam Fazel, and Jeff Bilmes. Interactive combinatorial bandits: Balancing competitivity and complementarity. *arXiv preprint arXiv:2207.03091*, 2022.

[93] Mukund Narasimhan and Jeff Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *Uncertainty in Artificial Intelligence (UAI)*, Edinburgh, Scotland, July 2005. Morgan Kaufmann Publishers.

[94] Mukund Narasimhan and Jeff Bilmes. Local search for balanced submodular clusterings. In *IJCAI*, pages 981–986, 2007.

[95] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978.

[96] 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.

[97] A. I. Nesvizhskii. A survey of computational methods and error rate estimation procedures for peptide and protein identification in shotgun proteomics. *Journal of Proteomics*, 73(11):2092 – 2123, 2010.

[98] James B Orlin, Andreas S Schulz, and Rajan Udwani. Robust monotone submodular function maximization. *Mathematical Programming*, 172(1):505–537, 2018.

[99] James G Oxley. *Matroid theory*, volume 3. Oxford University Press, USA, 2011.

[100] C. Y. Park, A. A. Klammer, L. Käll, M. P. MacCoss, and W. S. Noble. Rapid and accurate peptide identification from tandem mass spectra. *Journal of Proteome Research*, 7(7):3022–3027, 2008.

[101] Brittany N Pease, Edward L Huttlin, Mark P Jedrychowski, Eric Talevich, John Harmon, Timothy Dillman, Natarajan Kannan, Christian Doerig, Ratna Chakrabarti, Steven P Gygi, et al. Global analysis of protein expression and phosphorylation of three stages of plasmodium falciparum intraerythrocytic development. *Journal of Proteome Research*, 12(9):4028–4045, 2013.

[102] Pierre Perrault, Jennifer Healey, Zheng Wen, and Michal Valko. On the approximation relationship between optimizing ratio of submodular (rs) and difference of submodular (ds) functions. *arXiv preprint arXiv:2101.01631*, 2021.

[103] Adarsh Prasad, Stefanie Jegelka, and Dhruv Batra. Submodular meets structured: Finding diverse subsets in exponentially-large structured item sets. In *NIPS*, 2014.

[104] Robert Clay Prim. Shortest connection networks and some generalizations. *Bell Labs Technical Journal*, 36(6):1389–1401, 1957.

[105] Colorado Reed and Zoubin Ghahramani. Scaling the indian buffet process via submodular maximization. *arXiv preprint arXiv:1304.3285*, 2013.

[106] Rubén Ruiz and Thomas Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049, 2007.

[107] Kanthi K Sarpatwar, Baruch Schieber, and Hadas Shachnai. Interleaved algorithms for constrained submodular function maximization. *arXiv preprint arXiv:1705.06319*, 2017.

[108] Manohar Shamaiah, Siddhartha Banerjee, and Haris Vikalo. Greedy sensor selection: Leveraging submodularity. In *Proc. CDC*, pages 2572–2577. IEEE, 2010.

[109] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[110] Yusuke Shinohara. A submodular optimization approach to sentence set selection. In *ICASSP 2014*.

[111] Adish Singla, Ilija Bogunovic, Gábor Bartók, Amin Karbasi, and Andreas Krause. Near-optimally teaching the crowd to classify. *arXiv preprint arXiv:1402.2092*, 2014.

[112] J David Smith and My T Thai. Breaking the bonds of submodularity: Empirical estimation of approximation ratios for monotone non-submodular greedy maximization. *arXiv preprint arXiv:1702.07002*, 2017.

[113] P. Stobbe and A. Krause. Efficient minimization of decomposable submodular functions. In *NIPS*, 2010.

[114] J. D. Storey and R. Tibshirani. Statistical significance for genome-wide studies. *Proceedings of the National Academy of Sciences of the United States of America*, 100: 9440–9445, 2003.

[115] Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. In *Advances in Neural Information Processing Systems*, pages 1577–1584, 2009.

[116] Maxim Sviridenko, Jan Vondrák, and Justin Ward. Tight bounds for submodular and supermodular optimization with bounded curvature. *CoRR*, abs/1311.4728, 2013. URL `http://arxiv.org/abs/1311.4728`.

[117] Maxim Sviridenko, Jan Vondrák, and Justin Ward. Optimal approximation for submodular and supermodular optimization with bounded curvature. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1134–1148. Society for Industrial and Applied Mathematics, 2015.

[118] Zoya Svitkina and Lisa Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. volume 40, pages 1715–1737. SIAM, 2011.

[119] D. L. Tabb, C. G. Fernando, and M. C. Chambers. Myrimatch: highly accurate tandem mass spectral peptide identification by multivariate hypergeometric analysis. *Journal of Proteome Research*, 6:654–661, 2007.

[120] Donald M Topkis. *Supermodularity and complementarity*. Princeton university press, 2011.

[121] Sebastian Tschiatschek, Rishabh Iyer, Haochen Wei, and Jeff Bilmes. Learning mixtures of submodular functions for image collection summarization. In *Neural Information Processing Society (NIPS)*, Montreal, Canada, December 2014.

[122] Sebastian Tschiatschek, Rishabh K Iyer, Haochen Wei, and Jeff A Bilmes. Learning mixtures of submodular functions for image collection summarization. In *Advances in Neural Information Processing Systems*, pages 1413–1421, 2014.

[123] usul (https://cstheory.stackexchange.com/users/8243/usul). Maximizing a monotone supermodular function s.t. cardinality. Theoretical Computer Science Stack Exchange. URL `https://cstheory.stackexchange.com/q/33967`. URL:https://cstheory.stackexchange.com/q/33967 (version: 2016-03-03).

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

[125] J. Vondrák. Submodularity and curvature: the optimal algorithm. *RIMS Kokyuroku Bessatsu*, 23, 2010.

[126] Jan Vondrák. Symmetry and approximability of submodular maximization problems. *SIAM Journal on Computing*, 42(1):265–304, 2013.

[127] Yi-Jing Wang, Da-Chuan Xu, Yan-Jun Jiang, and Dong-Mei Zhang. Minimizing ratio of monotone non-submodular functions. *Journal of the Operations Research Society of China*, 7:449–459, 2019.

[128] Zengfu Wang, Bill Moran, Xuezhi Wang, and Quan Pan. Approximation for maximizing monotone non-decreasing set functions with a greedy method. *Journal of Combinatorial Optimization*, 31(1):29–43, 2016.

[129] Kai Wei, Yuzong Liu, Katrin Kirchhoff, and Jeff Bilmes. Using document summarization techniques for speech data subset selection. In *NAACL-HLT*, 2013.

[130] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Fast multi-stage submodular maximization. Beijing, China, 2014.

[131] Kai Wei, Yuzong Liu, Katrin Kirchhoff, Chris Bartels, and Jeff Bilmes. Submodular subset selection for large-scale speech training data. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Florence, Italy, 2014.

[132] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning (ICML)*, Lille, France, 2015.

[133] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *International conference on machine learning*, pages 1954–1963. PMLR, 2015.

[134] Kai Wei, Rishabh K Iyer, Shengjie Wang, Wenruo Bai, and Jeff A Bilmes. Mixed robust/average submodular partitioning: Fast algorithms, guarantees, and applications. In *Advances in Neural Information Processing Systems*, pages 2233–2241, 2015.

[135] Craig D Wenger, Douglas H Phanstiel, M Lee, Derek J Bailey, and Joshua J Coon. COMPASS: a suite of pre-and post-search proteomics software tools for OMSSA. *Proteomics*, 11(6):1064–1074, 2011.

[136] Laurence A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

[137] Linfeng Wu, Sophie I Candille, Yoonha Choi, Dan Xie, Lihua Jiang, Jennifer Li-Pook-Than, Hua Tang, and Michael Snyder. Variation and genetic control of protein abundance in humans. *Nature*, 499(7456):79–82, 2013.

[138] N. Zhang, R. Aebersold, and B. Schwikowski. ProbID: A probabilistic algorithm to identify peptides through sequence database searching using tandem mass spectral data. *Proteomics*, 2:1406–1412, 2002.

[139] Zheng Zhang, Scott Schwartz, Lukas Wagner, and Webb Miller. A greedy algorithm for aligning dna sequences. *Journal of Computational biology*, 7(1-2):203–214, 2000.

[140] Jingjing Zheng, Zhuolin Jiang, Rama Chellappa, and Jonathon P Phillips. Submodular attribute selection for action recognition in video. In *NIPS*, 2014.

[141] Tianyi Zhou and Jeff Bilmes. Minimax curriculum learning: Machine teaching with desirable difficulties and scheduled diversity. In *ICLR*, 2018.