博士論文

Submodular and Sparse Optimization Methods for Machine Learning and Communication

（機械学習と通信のための劣モジュラ・スパース最適化手法）

相馬　輔

Submodular and Sparse Optimization Methods for Machine Learning and Communication

Tasuku Soma

February 11, 2016

Abstract

This dissertation considers optimization problems arising from machine learning and communication, and devises efficient algorithms by exploiting submodular and sparse optimization techniques.

The first subject of this dissertation is submodular function maximization over the integer lattice. Maximizing a monotone submodular set function has been extensively studied in machine learning, social science, and related areas, since it encompasses a variety of problems considered in these areas and efficient greedy algorithms are available. However, we often face real scenarios that are beyond these existing models based on submodular set functions. For example, the existing models cannot capture the budget allocation problem in which we have to decide how much budget should be set aside. In this dissertation, we overcome this fundamental limitation of the existing models by exploiting submodularity over the integer lattice. We demonstrate that our new framework can naturally and concisely capture these difficult real scenarios, and provide efficient approximation algorithms for various constraints. In addition to theoretical guarantees, we conduct numerical experiments to establish practical efficiency of our model and algorithms.

Then we move on matrix completion problems. Roughly speaking, matrix completion problems are to determine missing entries in a given partial matrix under some criteria. Our first result is a new algorithm for constructing a multicast code for a wireless network using max-rank matrix completion. In the main ingredient of our algorithm, we employ the theory of mixed matrices, matroids, and submodular optimization. Second, we propose a new problem, the low-rank basis problem for a matrix subspace. Originally this problem comes from combinatorial optimization, but it turns out that this problem has various applications, including image separation and data compression. We present an efficient heuristic algorithm for this problem, using matroid theory and sparse optimization.

The last subject of this dissertation is compressed sensing. The main goal of compressed sensing is to recover a high-dimensional sparse signal from a few linear measurements. While the problem setup is quite simple, compressed sensing has been widely applied to machine learning, statistics, and signal processing. In this dissertation, we consider stable signal recovery in compressed sensing and show that the sum-of-squares method yields a new polynomial time recovery method for Rademacher sensing matrices. Our result sheds light on the power of sum-of-squares method on nonconvex problem arising from compressed sensing.

Acknowledgments

I have no doubt on that this dissertation would have been impossible without various support of many people. First of all, I am grateful for my supervisor, Satoru Iwata, who invited me to the world of academic research. He has been continuously giving many research ideas, suggestions, and advice during my Ph.D. program with great patience. My days in the Ph.D. course would have been completely different without his infinite support.

I also would like to acknowledge that many parts of this dissertation are based on joint works with my collaborators. Especially, I would like to thank Yuichi Yoshida. His endless mathematical ideas and great depth of knowledge on computer science have been truly remarkable and inspired me a lot.

The idea of the first part of this dissertation originates from my first joint project with Naonori Kakimura, Kazuhiro Inaba, Ken-ichi Kawarabayashi. Their research ideas and valuable discussions with them were really helpful and they took me to the research field of machine learning.

Chapter 8 benefited from many discussions and suggestions of Yuji Nakatsukasa and André Uschmajew during our joint work. They raised my small mathematical problem into really fascinating one, invited me to an exciting research area of sparse optimization, and greatly expanded the scope of my research.

I also would like to thank members and former members of our laboratory, especially Kunihiro Sadakane, Akiko Takeda, Yutaro Yamaguchi, Yu Yokoi, Naoki Ito, Shuichi Katsumata, and Kaito Fujii. Their valuable comments in our seminar and supports in my daily life in the laboratory have been really appreciated.

Furthermore, I would like to thank my parents for their support and love. Spending ten years as a student was not short, but they always encouraged me to continue my research.

Finally, I thank Japan Society for Promotion of Science, JST ERATO Kawarabayashi Large Graph Project, JST CREST Iwata team, and the University of Tokyo for their financial support.

Contents

Chapter 1

Introduction

This dissertation deals with various optimization problems arising from machine learning and communication: monotone submodular function maximization, matrix completion, and compressed sensing.

1.1 Monotone Submodular Function Maximization

Submodular functions are set functions often referred as "discrete analogue of convex functions". Submodular functions have been studied in the area of combinatorial optimization from 70's and there is a large body of work for submodular function *minimization* (see Fujishige [66], Schrijver [146], and references therein). Besides the celebrated results on submodular function minimization, *maximization* of a submodular functions had been paid less attention, since submodular function maximization is NP-hard and the known applications had been only facility location [102] and sensor placement [100] for a long time. However, in 2003, Kempe, Kleinberg, and Tardos [97] showed that the spread of influence among a social network can be modeled as maximization of a monotone submodular function, and provided a simple approximation algorithm for a problem of marketing decision called the *influence maximization*. Their paper sparked a lot of following works not only in the area of social networks but also in document summarization [111], [112], sensor placement [105], and computational advertisement [5]. The theoretical computer science community was also influenced by their paper and various new ideas and algorithms [13], [31], [43] for submodular function maximization were introduced. Now monotone submodular function maximization is widely known as a general and powerful framework in machine learning and related areas.

The beginning of studies of submodular function maximization dates back to the seminal work [128]–[130] of Nemhauser, Wolsey, and Fisher in 70's. They showed that a simple greedy algorithm achieves $(1 - 1/\mathrm{e})$-approximation for maximization of a *monotone* submodular function subject to a *cardinality constraint*. The approximation factor $1 - 1/\mathrm{e}$ cannot be improved in both the oracle model [130] without any condition, and the explicit function model [61] if $\mathrm{P} \neq \mathrm{NP}$. The greedy algorithm of Nemhauser, Wolsey, and Fisher [130] is simple enough to run in very large datasets and finds an (almost) optimal solution empirically. Algorithms for a cardinality constraint are still intensively studied, e.g., further speeding up [13], [123], [170], distributed algorithms [20], [122], and online algorithms [158].

For more complicated constraints such as a *knapsack constraint* and a *matroid constraint*, Nemhauser, Wolsey, and Fisher [130] showed that their simple greedy algorithm cannot achieve optimal approximation factor. Later, Sviridenko [159] showed that a greedy algorithm combined with partial enumeration achieves $(1 - 1/\mathrm{e})$-approximation for a knapsack constraint, which requires $O(n^5)$ function value computations. For a matroid constraint, Calinescu et

al. [31] provided a *continuous greedy algorithm* with *pipage rounding*, which achieves $(1-1/\mathrm{e})$-approximation in polynomial time. The continuous greedy algorithm has been further investigated by Vondrák and his coauthors, and turns out to be a general framework for various constraints [13], [43], [166].

Limitation of Existing Models A submodular function is usually defined as a set function over a ground set S, or equivalently, as a function over $\{0,1\}^S$. However, we often face a real-world setting that cannot be captured by the set function models. Let us give two examples.

Optimal Budget Allocation. A typical situation arises in the optimal budget allocation problem [5]. In this problem, we want to allocate a certain amount of budget among ad sources to maximize the number of influenced customers. The main difficulty is that we have to decide *how much* budget should be set aside for each ad source, rather than just deciding whether to use an ad source or not. Therefore, the optimal budget allocation problem is beyond the existing models based on submodular set functions.

Sensor Placement. For another example, let us consider the following sensor placement scenario. Suppose that we have several types of sensors with various energy levels. We assume a simple trade-off between information gain and cost. Sensors of a high energy level can collect a considerable amount of information, but we have to pay a high cost for placing them. Sensors of a low energy level can be placed at a low cost, but they can only gather limited information. Again, we want to decide *which type* of sensors should be placed at each spot, and hence set functions cannot capture the problem.

Contribution We overcome this fundamental limit of the previous models by extending them using submodularity over the *integer lattice*. This dissertation provides a generalization of the classical monotone submodular function maximization, namely, *monotone submodular function maximization over the integer lattice*, and presents efficient approximation algorithms. In our model, a function is defined on the integer lattice $\mathbb{Z}_+^S$. It turns out that our generalized model can formulate not only the problems mentioned above but also other real scenarios.

In Chapter 3, we present a formal description of the budget allocation problem and the generalized sensor placement, and give detailed motivation and technical overview of our results. Two main concepts of submodularity over the integer lattice, *lattice submodularity* and *diminishing return submodularity* (*DR-submodularity*) play key roles in our model.

In Chapter 4, we devise approximation algorithms for monotone submodular maximization over the integer lattice subject to the three constraints: a cardinality constraint, a polymatroid constraint, and a knapsack constraint. Our algorithms combine technical tools and ideas from the literature of monotone submodular (set) function maximization: decreasing threshold [13], the continuous greedy method [31], and rounding algorithms [43].

Chapter 5 is devoted to another submodular optimization problem, the *submodular cover problem* [173]. The submodular cover is a somewhat "dual" problem to the monotone submodular function maximization, but a similar greedy algorithm also works and various machine learning models were proposed based on the problem. We attempt to generalize the submodular cover into the integer lattice version, which we call the *diminishing return submodular cover problem*. We present a bicriteria approximation algorithm for this generalized submodular cover. Furthermore, we conduct numerical experiments using real and synthetic datasets, establishing practical efficiency of our algorithm.

1.2 Matrix Completion

The *matrix completion* is the following problem: For a given matrix which contains variables in its entries, find values for the variables so that the resulting matrix has a desired property, such as max-rank, low-rank, positive semidefinite, etc. Surprisingly, this simple problem encompasses many problems in various areas, e.g., maximum matching in general graphs [115], multicast with network coding [80], and recommendation systems [3].

The matrix completion problems live in a boundary of the two optimization worlds, combinatorial optimization and continuous optimization. For max-rank matrix completion, Lovász [115] proved that a simple randomized algorithm finds a solution if the underlying field is sufficiently large. This algorithm yields a surprisingly simple randomized algorithm for the graph matching problem using the Tutte matrix [165]. Based on the combinatorial structure of matrices with indeterminates, efficient deterministic algorithms were developed for various cases: mixed matrix completion [70], [80], matrix completion by rank-one matrices [89], [152], mixed skew-symmetric matrices [152].

On the other hand, convex relaxation methods have been central approaches for low-rank matrix completion (over the reals). Candès and Recht [33] showed that instead of minimizing the rank function, minimizing the *nuclear norm* is an efficient and powerful method for various settings. The nuclear norm is the convex envelope of the rank function and efficiently minimized by algorithms in convex optimization, e.g., alternating minimization [30]. These algorithms are fast and scalable, and applied to a large-scale recommendation system [3].

Contribution In Chapter 7, we show that mixed matrix completion [80] can be used for designing a faster *multicast* algorithm in a linear deterministic relay network. A linear deterministic relay network is a model of wireless communication introduced by Avestimehr, Diggavi and Tse [11] to study the capacity of a wireless information channel. We improve the time complexity of the previous multicast algorithm of Yadzi and Savari [175], by exploiting *simultaneous mixed matrix completion* of Harvey, Karger, and Murota [80].

Chapter 8 introduces a new problem, the *low-rank basis problem*. This problem is motivated by further extension of a class of matrices admitting a deterministic max-rank completion algorithm. Interestingly, it turns out that the low-rank basis problem has applications for dictionary learning and connections to matroids and tensor decompositions. For this problem, we provide a heuristic greedy algorithm and verify its performance by various numerical experiments. Our algorithm employs ideas from low-rank matrix completion such as the nuclear norm relaxation. Also, we provide some convergence analysis of our algorithm.

1.3 Compressed Sensing

Compressed sensing is a rapidly emerging field of signal processing, machine learning, and communication. The motivation of compressed sensing is the following: in the real-world setting, measurements are usually high dimensional data, while the complexity of the underlying generating model is very small. Mathematically, the most basic set up of compressed sensing is as follows: given $\boldsymbol{y} = A\boldsymbol{x} \in \mathbb{R}^m$, where $\boldsymbol{x} \in \mathbb{R}^n$ and $m \ll n$, can we reconstruct a signal $\boldsymbol{x}$? At the first sight, it seems impossible; a linear equation $\boldsymbol{y} = A\boldsymbol{x}$ has infinitely many solutions since $m < n$. However, this is not the case if we *a priori* know that $\boldsymbol{x}$ is *sparse*, i.e., most entries are zero. Candès, Romburg, and Tao [34], and Dohono [53] showed that if the measurement matrix A satisfies a certain technical condition (such as the *null space property* or the *restricted isometry property*), one can efficiently reconstruct the original signal $\boldsymbol{x}$. They also showed that *random* measurement matrix satisfies these technical conditions with high probability. After

their groundbreaking papers, compressed sensing has been widely applied to statistics, magnetic resonance image (MRI), lowrank matrix completion, etc.

ℓ_1 *minimization* (also known as the *basis pursuit*) [44] is the most basic and important reconstruction algorithm. The method is surprisingly simple; it returns a minimizer $\boldsymbol{z}$ of the ℓ_1 norm subject to $A\boldsymbol{z} = \boldsymbol{y}$. This optimization can be done by linear programming and hence ℓ_1 minimization is a polynomial time reconstruction scheme. Under reasonable condition on the measurement matrix, ℓ_1 minimization can reconstruct the original signal. In many real-world applications, the original signal is not exactly sparse but close to a sparse signal. Candès, Romburg, and Tao [35] showed that ℓ_1 minimization is also able to (approximately) reconstruct the original signal even in such a case.

ℓ_1 minimization were generalized to *nonconvex compressed sensing* [46], [106]. In nonconvex compressed sensing, we use the ℓ_q-quasi norm (where $0 < q < 1$) instead of the ℓ_1-norm. Therefore the method is called the ℓ_q *minimization*. It was shown that the ℓ_q minimization can reconstruct the original signal from samples fewer than ℓ_1 minimization. Also *any* sparse signal can be reconstructed by suitable linear measurements $\boldsymbol{y} = A\boldsymbol{x}$ by ℓ_q minimization for some small q. However, nonconvex compressed sensing has a big drawback; the optimization problem we have to solve is noncovex and therefore we are often suffered from local minima. For general A and $\boldsymbol{x}$, ℓ_q minimization is NP-hard [119], [127].

Contribution We show that for some class of measurement matrices, nonconvex compressed sensing can be done in polynomial time, by exploiting the *sum of squares (SoS) method* [107], [131], [136], [150]. The SoS method (also known as the *Lassere-SoS hierarchy*) is a relaxation approach for *polynomial optimization problem (POP)*, based on the *semidefinite programming (SDP)* and deep results on real algebraic geometry. Although POP is nonconvex programming in general, the SoS method often yields a (near) optimal solution. Our work sheds light on the power of the SoS method in nonconvex compressed sensing. The details of our work are described in Chapter 10.

1.4 Organization of This Thesis

Figure 1.1 illustrates the structure of this dissertation.

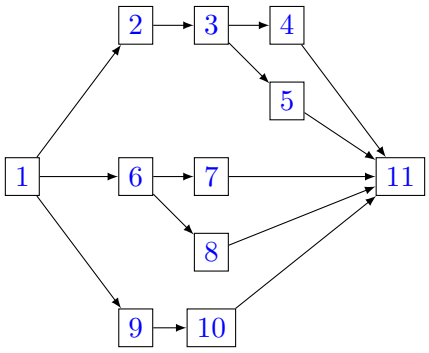


Figure 1.1: The dependence structure of chapters.

Chapters 2–5 are devoted to monotone submodular optimization over the integer lattice. The main results of this part are developing new machine learning models with submodularity over the integer lattice and designing approximation algorithms for various formulations. The first two chapters (Chapters 2 and 3) of this part reviews the theory of submodular functions and matroids, existing models in machine learning, and algorithms for monotone submodular

optimization. By investing budget allocation problem, we show the limitation of the existing models and are led to generalized models with submodularity over the integer lattice. Chapter 3 also presents a summary of our algorithmic results on our novel framework. The following two chapters (Chapters 4 and 5) describes technical details of our algorithms.

Chapters 6–8 deal with the matrix completion. This part heavily depends on linear algebraic techniques, such as mixed matrices, SVD, and various sparse optimization techniques. Chapter 6 contains a brief review on matrix completion. Chapters 7 and 8 are devoted to multicast algorithm in LDRN and low-rank basis problem, respectively.

In Chapters 9 and 10, we discuss the last subject, compressed sensing. First we make a technical review on compressed sensing in Chapter 9. Then we present our nonconvex compressed sensing algorithm via the SoS method in Chapter 10.

Finally we conclude this dissertation in Chapter 11, noting several open problems.

1.4.1 Publications Contained in This Dissertation

The contents of this dissertation are based on the following publications.

- T. Soma, N. Kakimura, K. Inaba, and K. Kawarabayashi, "Optimal Budget Allocation: Theoretical Guarantee and Efficient Algorithm," in *Proceedings of the 31st International Conference of Machine Learning (ICML)*, 2014 (Chapter 3).
- T. Soma and Y. Yoshida, "Maximizing Submodular Functions with the Diminishing Return Property over the Integer Lattice," in *Integer Programming and Combinatorial Optimization (IPCO)*, 2016, to appear (Chapters 3 and 4).
- T. Soma and Y. Yoshida, "A Generalization of Submodular Cover via the Diminishing Return Property on the Integer Lattice," in *Advances on Neural Information Processing Systems (NIPS)*, 2015, pp. 847–855 (Chapter 5).
- T. Soma, "Multicasting in Linear Deterministic Relay Network by Matrix Completion," *IEEE Transactions on Information Theory*, vol. 62, no. 2, pp. 870–875, 2016 (Chapter 7).
- Y. Nakatsukasa, T. Soma, and A. Uschmajew, "Finding a low-rank basis in a matrix subspace," *ArXiv Preprint*, 37 pages, 2015 (Chapter 8).
- T. Soma and Y. Yoshida, "Non-Convex Compressed Sensing with the Sum-of-Squares Method," in *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2016, pp. 570–579 (Chapter 10).

1.5 Notation

We denote the set of complex numbers, reals, integers, and natural numbers by $\mathbb{C}$, $\mathbb{R}$, $\mathbb{Z}$, and $\mathbb{N}$, respectively. Also we denote a field by $\mathbb{F}$. The set of nonnegative reals and integers are denoted by $\mathbb{R}_+$ and $\mathbb{Z}_+$, respectively. For $n \in \mathbb{N}$, $[n]$ represents the set $\{1, 2, \ldots, n\}$. For a subset X of a ground set S and $s \in S \setminus X$, we denote $X \cup \{s\}$ by $X \cup s$. Similarly, for $s \in X$, we denote $X \setminus \{s\}$ by $X \setminus s$.

We use italic letters for scalars and matrices, e.g., $a, b, c \in \mathbb{R}$ and $A, B, C \in \mathbb{R}^{m \times n}$. We use bold italic letters for vectors, e.g., $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$. We denote the ith entry of a vector $\boldsymbol{x} \in \mathbb{R}^n$ by $x(i)$. The ith standard unit vector is denoted by $\mathbf{e}_i$; i.e., $\mathrm{e}_i(i) = 1$ and $\mathrm{e}_i(j) = 0$ $(j \neq i)$. The zero vector is denoted by $\mathbf{0}$ and the all-one vector is denoted by $\mathbf{1}$. For a finite set S, $\mathbb{F}^S$ denotes the set of $|S|$-dimensional vectors over $\mathbb{F}$ whose entries are indexed by S. For $X \subseteq S$ and $\boldsymbol{x} \in \mathbb{F}^S$,

we denote $x(X) := \sum_{i \in X} x(i)$. Similarly, we denote $\boldsymbol{x}[X]$ be the subvector of $\boldsymbol{x}$ whose entries are indexed by X. We denote the characteristic vector of X by $\mathbf{1}_X$; i.e., $\mathbf{1}_X := \sum_{i \in X} \mathbf{e}_i$. For $\boldsymbol{v} \in \mathbb{R}^n$ and $X \subseteq [n]$, we define a vector $\boldsymbol{v}_X := \sum_{i \in X} x(i)\mathbf{e}_i$.

For $0 < p < \infty$, the ℓ_p norm of $\boldsymbol{x} \in \mathbb{R}^n$ is defined as $\|\boldsymbol{x}\|_p := \left(\sum_{i \in [n]} |x(i)|^p\right)^{1/p}$. The ℓ_∞ norm of $\boldsymbol{x}$ is $\|\boldsymbol{x}\|_\infty := \max_{i \in [n]} |x(i)|$. The ℓ_0 norm of $\boldsymbol{x}$ is the number of nonzero entries of $\boldsymbol{x}$. The vector inner product is denoted by $\boldsymbol{x}^\top \boldsymbol{y}$ or $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. The support of a vector $\boldsymbol{x}$ is denoted by $\mathrm{supp}(\boldsymbol{x}) = \{i \in [n] : x(i) \neq 0\}$. Similarly, the positive and negative supports of $\boldsymbol{x}$ are defined as $\mathrm{supp}^+(\boldsymbol{x}) = \{i \in [n] : x(i) > 0\}$ and $\mathrm{supp}^-(\boldsymbol{x}) = \{i \in [n] : x(i) < 0\}$, respectively.

For a matrix $A \in \mathbb{F}^{m \times n}$, the (i,j)-entry of A is denoted by A_{ij}. Similarly for vectors, we write $A \in \mathbb{F}^{S \times T}$ for finite sets S and T to denote that A is a $|S| \times |T|$ matrix whose rows and columns are indexed by S and T, respectively. For nonempty sets $X \subseteq S$ and $Y \subseteq T$, $A[X, Y]$ denotes the submatrix of A whose rows and columns are indexed by X and Y, respectively.

The trace of $A \in \mathbb{F}^{n \times n}$ is defined by $\mathrm{tr}(A) := \sum_{i \in [n]} A_{ii}$. The transpose of A is denoted by $A^\top$. The Frobenius norm of $A \in \mathbb{R}^{m \times n}$ is defined as $\|A\|_F := \sqrt{\mathrm{tr}(A^\top A)}$. The largest singular value of A is denoted by $\sigma_{\max}(A)$.

For a function $f : 2^S \to \mathbb{R}$, $X \subseteq S$, and $s \in S$, we define $f(s \mid X) := f(X \cup s) - f(X)$. Similarly, for a function $f : \mathbb{Z}^S \to \mathbb{R}$, and $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}^S$, we define $f(\boldsymbol{y} \mid \boldsymbol{x}) := f(\boldsymbol{x} + \boldsymbol{y}) - f(\boldsymbol{x})$. Any vector $\boldsymbol{x} \in \mathbb{Z}_+^S$ can be regarded as the multiset in which each $s \in S$ is contained $x(s)$ times. We denote this multiset by $\{\boldsymbol{x}\}$. We abuse several set operations for multisets. For arbitrary two multisets $\{\boldsymbol{x}\}$ and $\{\boldsymbol{y}\}$, we define $\{\boldsymbol{x}\} \setminus \{\boldsymbol{y}\}$ as the multiset such that each $s \in S$ is contained $\max\{x(s) - y(s), 0\}$ times. For a multiset $\{\boldsymbol{x}\}$, we define $|\{\boldsymbol{x}\}| := x(S)$.

We denote the expectation of a random variable Z by $\mathbf{E}[Z]$. We drop the parentheses and write $\mathbf{E}\, Z$ if the argument is clear from the context. If we need to specify a distribution $\mathcal{D}$, we write $\mathbf{E}_{Z \sim \mathcal{D}}\, Z$. The probability of an event E is denoted by $\Pr(E)$.

The base of the natural logarithm is denoted by e.

Pseudocode

Throughout this dissertation, we use the following convention for pseudocode description of algorithms, which is mostly taken from the Python programming language.

- The block structure is represented by indent. We do not use the "end" keywords such as **endif** and **endfor**.
- Also we drop **do** keyword in the **for** and **while** statements. Alternatively we simply use a colon. Similarly, we drop **then** keyword in the **if** statement.

Chapter 2

Submodular Functions, Matroids, and Polymatroids

This chapter provides a high-level overview on submodular functions, matroids, and polymatroids. The readers who are familiar with these subjects can skip this chapter and proceed to Chapter 3 directly.

2.1 Submodular Functions

In this section, we briefly summarize the theory of submodular functions. For more details, refer to monographs [66], [103].

Let S be a finite set. A set function $f : 2^S \to \mathbb{R}$ is called a *submodular function* if

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y) \tag{2.1}$$

for any $X, Y \subseteq S$. A *supermodular* function is a set function f such that $-f$ is submodular. If a set function is both submodular and supermodular, it is called a *modular* function. A set function $f : 2^S \to \mathbb{R}$ is said to be *monotone* if $f(X) \leq f(Y)$ whenever $X \subseteq Y \subseteq S$.

An equivalent and more intuitive definition for submodularity is by the *diminishing return property*:

$$f(X \cup s) - f(X) \geq f(Y \cup s) - f(Y) \tag{2.2}$$

for all $X \subseteq Y$ and $s \in S \setminus Y$. The following slightly simpler form is sometimes useful:

$$f(X \cup s) - f(X) \geq f(X \cup \{s, t\}) - f(X \cup t) \tag{2.3}$$

for all $X \subseteq S$ and distinct $s, t \in S \setminus X$.

Example 2.1.1 (Coverage Function)**.** Let $G = (S, T; E)$ be a bipartite graph. For $X \subseteq S$, let $\Gamma(X)$ be the set of neighbors of S; i.e., $\Gamma(X) = \{t \in T : t \text{ is connected to some } s \in X\}$. Then $f(X) = |\Gamma(X)|$ is a monotone submodular function.

Example 2.1.2 (Cut Function)**.** Let $G = (V, E)$ be an undirected graph. For $X \subseteq V$, let $f(X)$ be the number of edges such that exactly one of its endpoint is in X. Then f is a submodular function. Furthermore, f is *symmetric*; i.e., $f(X) = f(V \setminus X)$ for any X.

Example 2.1.3 (Entropy Function)**.** Let $\boldsymbol{Z} = [Z_1, \ldots, Z_n]^\top$ be an n-dimensional discrete random variable. For $X \subseteq [n]$, define $f(X)$ as the joint entropy of Z_i $(i \in X)$; i.e., $f(X) =$

$H(\boldsymbol{Z}_X)$, where H is the Shannon entropy function. We define $f(\emptyset) = 0$. Fujishige [65] showed that f is a monotone submodular function. It is interesting to ask if an arbitrary monotone submodular function f (with $f(\emptyset) = 0$) can be constructed as above. Indeed this was an open problem in information theory and disproved by Zhang and Yeung [176].

Example 2.1.4 (Influence Function [97])**.** Let $G = (V, A)$ be a digraph. Assume that each arc $uv \in A$ has a probability p_{uv}. Let us consider the following spreading process. First we pick a subset $X \subseteq V$ called a *feed set* and all the vertices in X become *active* in time 0. Once a vertex u becomes active in time t, then u will activate each of its neighbor v independently with probability p_{uv}. If u succeeds, v becomes active in time $t+1$. If u fails, u will never try to activate v in the subsequent rounds. Let $f(X)$ be the expected number of active vertices after time $t = n$, with an initial feed set X. Then f is a monotone submodular function.

2.1.1 Extensions of Submodular Function

A set function $f : 2^S \to \mathbb{R}$ can be regarded as a boolean function $f : \{0,1\}^S \to \mathbb{R}$ via the natural corresponding between a subset $X \subseteq S$ and its characteristic vector $\mathbf{1}_X$. There are several canonical ways to extend a set function $f : 2^S \to \mathbb{R}$ to a real function $f : [0,1]^S \to \mathbb{R}$. We review two useful extensions: the Lovász extension and the multilinear extension.

Lovász Extension The *Lovász extension* is defined via correlated rounding of a fractional vector $\boldsymbol{x} \in [0,1]^S$. For $\theta \in [0,1]$ and $\boldsymbol{x} \in [0,1]^S$, let $L_\theta(\boldsymbol{x}) = \{s \in S : x(s) \geq \theta\}$. The Lovász extension $\hat{f} : [0,1]^S \to \mathbb{R}$ of $f : 2^S \to \mathbb{R}$ is defined as

$$\hat{f}(\boldsymbol{x}) = \mathbf{E}_{\theta \sim [0,1]}[f(L_\theta(\boldsymbol{x}))], \tag{2.4}$$

where θ is drawn from the uniform distribution over the interval $[0,1]$. The Lovász extension can be evaluated efficiently. Let us fix $\boldsymbol{x} \in [0,1]^S$ and sort the elements of S as $x(s_1) \geq x(s_2) \geq \cdots \geq x(s_n)$, where $n = |S|$. Then

$$\hat{f}(\boldsymbol{x}) = (1 - x(s_1))f(\emptyset) + \sum_{i=1}^{n-1}(x(s_i) - x(s_{i+1}))f(\{s_1, \ldots, s_i\}) + x(s_n)f(S). \tag{2.5}$$

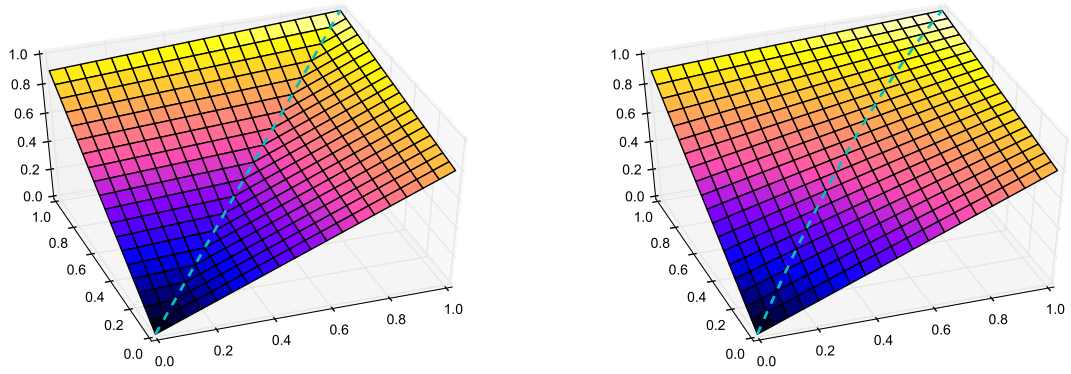
The following fundamental relation between submodularity of f and convexity of its Lovász extension $\hat{f}$ is due to Lovász [116].

Theorem 2.1.5 (Lovász [116])**.** *A set function $f : 2^S \to \mathbb{R}$ is submodular if and only if its Lovász extension $\hat{f}$ is convex.*

Multilinear Extension The *multilinear extension* is another canonical extension of a set function, which has played a central role in submodular function maximization. For $\boldsymbol{x} \in [0,1]^S$, let $R(\boldsymbol{x})$ be a random set in which each $s \in S$ is independently contained with probability $x(s)$. The multilinear extension $F : [0,1]^S \to \mathbb{R}$ of $f : \{0,1\}^S \to \mathbb{R}$ is given by

$$F(\boldsymbol{x}) = \mathbf{E}[f(R(\boldsymbol{x}))] = \sum_{X \subseteq S} f(X) \prod_{s \in X} x(s) \prod_{s \notin X} (1 - x(s)). \tag{2.6}$$

Note that since the sum contains 2^n terms, one cannot (efficiently) compute the *exact* value of $F(\boldsymbol{x})$, unless f is a very specific function such as a coverage function. However, one can compute an *approximate* value by iteratively sampling $R(\boldsymbol{x})$. As in the Lovász extension, the following relation between submodularity of f and (directional) convexity of F holds.



(a) The Lovász extension

(b) The multilinear extension

Figure 2.1: The two extensions for a monotone submodular function $f : 2^S \to \mathbb{R}$, where $S = \{1, 2\}$, $f(\emptyset) = 0$, $f(1) = 0.8$, $f(2) = 0.9$, and $f(S) = 1$. A path above the diagonal segment is drawn in a dotted line.

Theorem 2.1.6 (Calinescu et al. [31]). *Let $f : 2^S \to \mathbb{R}$ be a set function and $F : [0,1]^S \to \mathbb{R}$ be its multilinear extension.*

1. *If f is monotone, then $\frac{\partial F}{\partial x_i} \geq 0$ for any $i \in S$.*
2. *If f is submodular, then $\frac{\partial^2 F}{\partial x_i \partial x_j} \leq 0$ for distinct $i, j \in S$.*
3. *Suppose that f is monotone and submodular. For $\boldsymbol{x} \in [0,1]^S$ and $\boldsymbol{v} \geq \mathbf{0}$, let $\phi(t) = F(\boldsymbol{x} + t\boldsymbol{v})$. Then $\phi(t)$ is a concave function.*

2.1.2 Submodular Function Minimization

Minimization of a submodular function has been a central problem in combinatorial optimization. Historically, the first polynomial time algorithm was due to Grötschel, Lovász, and Schrijver [77] in 1980s, via the Lovász extension and the ellipsoid method. Combinatorial algorithms for submodular function minimization were independently devised by Iwata, Fleicher, and Fujishige [91], and Schrijver [145] around 2000.

The Fujishige-Wolfe minimum norm point algorithm [66] also solves submodular function minimization and has been recognized as the fastest practical method, though not being proved as a polynomial time algorithm. Recently, Jegelka, Lin, and Bilmes [95] improved the Fujishige-Wolfe algorithm to a scalable and efficient algorithm for large scale datasets.

The submodular function minimization has served as a building block in various algorithms: pipage rounding in matroid polytope [31], finding a flow in polylinking system [72], etc. For further details of submodular function minimization, refer to a survey [90] and a monograph [12].

2.2 Matroids and Polymatroids

In this section, we make a quick review on matroids and polymatroids.

2.2.1 Matroids

A *matroid* is one of the central objects in combinatorial optimization, introduced by Whitney [172]. Let $\mathcal{I}$ be a family of subsets of a finite set S. The pair $(S, \mathcal{I})$ is called a *matroid* if the following three conditions hold:

(I1) $\emptyset \in \mathcal{I}$.

(I2) If $I \in \mathcal{I}$ and $I' \subseteq I$, then $I' \in \mathcal{I}$.

(I3) For arbitrary members I_1 and I_2 of $\mathcal{I}$ with $|I_1| < |I_2|$, there exists an element $i \in I_2 \setminus I_1$ such that $I_1 \cup i \in \mathcal{I}$.

For a matroid $(S, \mathcal{I})$, we call S the *ground set* and a member of $\mathcal{I}$ an *independent set*.

We will see several examples of matroids.

Example 2.2.1 (Free Matroid)**.** The family of all the subsets in S trivially satisfies the above axioms. This matroid is called the *free matroid* on S.

Example 2.2.2 (Uniform Matroid)**.** The family of subsets in S of size at most k also satisfies the above axioms. This matroid is called the *uniform matroid* on S with rank k.

Example 2.2.3 (Partition Matroid)**.** Assume that the ground set S is partitioned into $S_1, \ldots, S_m$. Let $k_1, \ldots, k_m$ be nonnegative integers. Define $\mathcal{I}$ to be the family of subsets intersecting S_i in at most k_i elements for $i = 1, \ldots, m$; i.e., $\mathcal{I} = \{I \subseteq S : |I \cap S_i| \leq k_i \, (i = 1, \ldots, m)\}$. Matroids obtained in this way are called *partition matroids*.

Example 2.2.4 (Linear Matroid)**.** Let S be the set of column indices of a matrix A. A subset $I \subseteq S$ is said to be independent if column vectors of A indexed by I are linearly independent. Let $\mathcal{I}$ denote the family of independent sets. Then $(S, \mathcal{I})$ is a matroid called the *linear matroid* or the *vector matroid* of a matrix A, which we denote by $\mathbf{M}[A]$. One can easily check that the matroids mentioned above are special cases of linear matroids.

Rank function The *rank* of a subset X of S is the maximum size of an independent set contained in X. The function $r : 2^S \to \mathbb{Z}_+$ that returns the rank of each subset is called the *rank function* of a matroid $(S, \mathcal{I})$. The rank function r satisfies the following properties:

(R1) $0 \leq r(X) \leq |X|$ for an arbitrary subset X.

(R2) r is monotone.

(R3) r is submodular.

In fact, matroids can be characterized by the rank function; i.e., if a function $r : 2^S \to \mathbb{Z}_+$ satisfies the above conditions (R1) to (R3), then the pair of set S and the family $\mathcal{I} := \{I \subseteq S : r(I) = |I|\}$ forms a matroid.

Bases of a matroid Another characterization of matroids is in terms of *bases*. Let $(S, \mathcal{I})$ be a matroid with the rank function r. An independent set B is called a *base* if $r(B) = r(S)$. Equivalently, B is a base if it is a maximal independent set. Bases of a matroid admit the following *exchanging property*:

(BM) For arbitrary bases B_1 and B_2 and for $i \in B_1 \setminus B_2$, there exists an element $j \in B_2 \setminus B_1$ such that $(B_1 \setminus i) \cup j$ is a base.

Conversely, if a nonempty family $\mathcal{B}$ satisfies the property (BM), then the family $\mathcal{I} := \{I \subseteq S : I$ is contained in some member of $\mathcal{B}\}$ is an independent set family of a matroid.

Matroid polytope A *matroid polytope* is one of the landmarks in polyhedral combinatorics. For a matroid $\mathbf{M} = (S, \mathcal{I})$, the matroid polytope $P(\mathbf{M})$ is the convex full of characteristic vectors of independent sets; i.e., $P(\mathbf{M}) = \mathrm{conv}\{\mathbf{1}_I \in \mathbb{R}^S : I \in \mathcal{I}\}$. Similarly, the *base polytope* $B(\mathbf{M})$ is the convex full of characteristic vectors of bases. Edmonds [57] proved that a matroid polytope can be written as $P(\mathbf{M}) = \{\boldsymbol{x} \in \mathbb{R}^S_+ : x(X) \leq r(X) \quad (X \subseteq S)\}$ and $B(\mathbf{M}) = \{\boldsymbol{x} \in P(\mathbf{M}) : x(S) = r(S)\}$.

The *(strong) matroid polytope membership problem* is the following task. Given $\boldsymbol{x} \in [0,1]^S$ and matroid $\mathbf{M}$, decide $\boldsymbol{x} \in P(\mathbf{M})$, and if so, represent $\boldsymbol{x}$ as a convex combination of vertices in $P(\mathbf{M})$. Tardos, Tovey, and Trick [162] proved that the matroid polytope membership can be solved in $O(n^8)$ time, based on the algorithm of Cunningham [50].

2.2.2 Polymatroids

A *polymatroid* is a generalization of a matroid polytope. Let $\rho : 2^S \to \mathbb{Z}_+$ be a monotone submodular set function with $\rho(\emptyset) = 0$. The (integral) *polymatroid* associated with ρ is the polytope $P = P(\rho) = \{\boldsymbol{x} \in \mathbb{R}^S_+ : x(X) \leq \rho(X) \quad (X \subseteq S)\}$, and ρ is called the *rank function* of P. The *base polytope* of a polymatroid P is defined as $B := \{\boldsymbol{x} \in P : x(S) = \rho(S)\}$. The set of integral points in B satisfies the following *simultaneous exchange property*:

($\mathbf{BP}_\pm$) For any $\boldsymbol{x}, \boldsymbol{y} \in B \cap \mathbb{Z}^S_+$ and $s \in \mathrm{supp}^+(\boldsymbol{x} - \boldsymbol{y})$, there exists $t \in \mathrm{supp}^+(\boldsymbol{y} - \boldsymbol{x})$ such that $\boldsymbol{x} - \mathbf{e}_s + \mathbf{e}_t \in B \cap \mathbb{Z}^S_+$ and $\boldsymbol{y} + \mathbf{e}_s - \mathbf{e}_t \in B \cap \mathbb{Z}^S_+$.

The following lemma can be derived by the simultaneous exchange property.

Lemma 2.2.5. *Let $\boldsymbol{x}, \boldsymbol{y} \in B \cap \mathbb{Z}^S_+$ and $I(\boldsymbol{x}) := \{(s,i) : s \in \mathrm{supp}^+(\boldsymbol{x}), 1 \leq i \leq x(s)\}$. Then there exists a map $\phi : I(\boldsymbol{x}) \to \mathrm{supp}^+(\boldsymbol{y})$ such that $\boldsymbol{x} - \mathbf{e}_s + \mathbf{e}_{\phi(s,i)} \in B \cap \mathbb{Z}^S_+$ for each $s \in \mathrm{supp}^+(\boldsymbol{x})$ and $1 \leq i \leq x(s)$, and that $\boldsymbol{y} = \sum_{(s,i) \in I(\boldsymbol{x})} \mathbf{e}_{\phi(s,i)}$.*

Proof. Induction on $|\{\boldsymbol{x}\} \setminus \{\boldsymbol{y}\}|$. If $|\{\boldsymbol{x}\} \setminus \{\boldsymbol{y}\}| = 0$, then $\phi(s,i) := s$ satisfies the condition. Let us assume that $|\{\boldsymbol{x}\} \setminus \{\boldsymbol{y}\}| > 0$. Let us fix $s \in \mathrm{supp}^+(\boldsymbol{x} - \boldsymbol{y})$ arbitrarily. By the simultaneous exchange property, we can find $t \in \mathrm{supp}^+(\boldsymbol{y} - \boldsymbol{x})$ such that $\boldsymbol{x} - \mathbf{e}_s + \mathbf{e}_t \in B \cap \mathbb{Z}^S_+$ and $\boldsymbol{y}' := \boldsymbol{y} + \mathbf{e}_s - \mathbf{e}_t \in B \cap \mathbb{Z}^S_+$. By the induction hypothesis, one can obtain $\phi' : I(\boldsymbol{x}) \to \mathrm{supp}^+(\boldsymbol{y}')$ satisfying the conditions. The desired ϕ can be obtained by extending ϕ' as $\phi'(s, y(s)+1) := t$. □

Tight Sets and Exchange Capacity For a vector $\boldsymbol{x}$ in a polymatroid $P(\rho)$, a subset $X \subseteq S$ is said to be *tight* (with respect to $\boldsymbol{x}$) if $x(X) = \rho(X)$. The family of tight sets forms a distributive lattice; i.e., for arbitrary tight sets X and Y, $X \cup Y$ and $X \cap Y$ are also tight. This could be easily verified via the submodularity of ρ:

$$x(X) + x(Y) = \rho(X) + \rho(Y) \geq \rho(X \cup Y) + \rho(X \cap Y) \geq x(X \cup Y) + x(X \cap Y).$$

The *exchange capacity* of $\boldsymbol{x}$ and $i, j \in S$ is defined as

$$\hat{c}(\boldsymbol{x}, i, j) = \max\{\alpha \geq 0 : \boldsymbol{x} + \alpha(\mathbf{e}_i - \mathbf{e}_j) \in P(\rho)\}.$$

Note that the exchange capacity can be computed by minimizing $\rho(X) - x(X)$ over $\{X \subseteq S \setminus j : i \in X\}$, which can be done by submodular function minimization.

Greedy Algorithm on Polymatroid Let us consider the following problem

$$\begin{array}{ll} \text{minimize} & \boldsymbol{w}^\top \boldsymbol{x} \\ \text{subject to} & \boldsymbol{x} \in P(\rho), \end{array} \tag{2.7}$$

where $\boldsymbol{w} \in \mathbb{R}^S$ and $P(\rho)$ is a polymatroid on S. Edmonds [57] presented the greedy algorithm for this problem, which works as follows. Let us assume that $S = \{s_1, \ldots, s_n\}$ with $w(s_1) \leq w(s_2) \leq \cdots \leq w(s_n)$. Define $S_i = \{s_1, \ldots, s_i\}$ $(i = 1, \ldots, n)$ and $S_0 = \emptyset$. The greedy algorithm returns a vector $\boldsymbol{x} \in \mathbb{R}^S$ such that

$$x(s_i) = \rho(S_i) - \rho(S_{i-1}) \quad (i = 1, \ldots, n).$$

Edmonds [57] proved that $\boldsymbol{x}$ is an optimal solution of this problem. We note that maximization can be also solved in a similar manner. Therefore, optimization of a linear function on a polymatroid can be solved in $O(n \log n)$ time, provided that function values can be computed in $O(1)$ time.

Chapter 3

Submodularity over the Integer Lattice and Budget Allocation

In this chapter, we present our novel framework, namely, monotone submodular function optimization over the integer lattice. First we review models and algorithms in the literature in Section 3.1. Then we introduce the budget allocation problem and point out that these existing models based on set functions cannot capture this problem in Section 3.2. To overcome this issue, in Section 3.3, we try to generalize the previous models using submodularity over the integer lattice and show that the budget allocation problem can be viewed as maximization of monotone submodular functions over the integer lattice. In Section 3.4, we present the details of our framework and an overview of algorithmic results. We then show that our framework is able to capture not only the original budget allocation problem but also its variants, generalized sensor placement, and text summarization allowing multiple choices. We conclude this chapter with useful facts on submodular functions over the integer lattice in Section 3.5.

3.1 Monotone Submodular Optimization: Literature Overview

In the last decade, optimization of a submodular function has attracted particular interest in the machine learning community. One reason of this is that many real-world models naturally admit the diminishing return property. For example, document summarization [111], [112], influence maximization in viral marketing [97], object detection [45], [153], and sensor placement [105] can be described with the concept of submodularity, and efficient algorithms have been devised by exploiting submodularity.

Roughly speaking, a variety of proposed models in machine learning boil down to one of the following two problems.

Problem 3.1.1 (Monotone Submodular Function Maximization [130])**. Given:** a monotone submodular function $f : 2^S \to \mathbb{R}_+$ with $f(\emptyset) = 0$ and a family $\mathcal{F} \subseteq 2^S$.

$$\begin{aligned} \text{maximize} \quad & f(X) \\ \text{subject to} \quad & X \in \mathcal{F} \end{aligned}$$

Problem 3.1.2 (Submodular Cover [173])**. Given:** monotone submodular functions $c, f : 2^S \to \mathbb{R}_+$ with $c(\emptyset) = f(\emptyset) = 0$ and $\alpha > 0$.

$$\begin{aligned} \text{minimize} \quad & c(X) \\ \text{subject to} \quad & f(X) \geq \alpha \end{aligned}$$

In the submodular function maximization, the following classes of a family $\mathcal{F}$ are particularly important from a both theoretical and practical perspective.

Cardinality Constraint $\mathcal{F} = \{X : |X| \leq k\}$ for a given $k \in \mathbb{Z}_+$.

Matroid Constraint $\mathcal{F}$ is the family of independent sets of a given matroid $\mathbf{M}$.

Knapsack Constraint $\mathcal{F} = \{X : w(X) \leq 1\}$ for a given weight vector $\boldsymbol{w} \in \mathbb{R}_+^S$.

Note that a cardinality constraint is a common special case of a matroid constraint and a knapsack constraint.

Intuitively, $\mathcal{F}$ represents a feasibility constraint and $f(X)$ measures "quality" of solution X. Similarly, $c(X)$ represents the cost of a solution. In monotone submodular function maximization, we are to find X of maximum quality satisfying the constraint $X \in \mathcal{F}$. On the other hand, the objective of submodular cover is to find X of minimum cost with the worst quality guarantee α. This intuition is illustrated by investigating the following three examples, each of which is an existing machine learning model.

Example 3.1.3 (Influence Maximization [97])**.** Let us recall the influence function in Example 2.1.4. Kempe, Kleinberg, and Tardos [97] considered the following marketing problem called influence maximization. In this problem, we want to promote new products in a social network. Precisely, a social network is a graph whose vertices are customers. We can distribute products to a set X of initial feed customers for free, expecting that other customers will be influenced by the spreading process described in Example 2.1.4. Since we do not want to give products for many people, it is natural to pose a constraint on the number of free products. The task of problem is to maximize the expected number of active customers under constraint $|X| \leq k$. This problem boils down to monotone submodular function maximization.

One also can consider another scenario. We want to activate at least α customers on average, while minimizing the number of free items. Then this problem is a instance of submodular cover (with $c(X) = |X|$).

Example 3.1.4 (Document Summarization [111], [112])**.** The task of document summarization is to choose a small set of sentences that summarizes the original document well. Lin and Bilmes [111], [112] proposed various document summarization models and we will review one of them. Let S be a set of sentences in a given document and $a_{i,j} \geq 0$ be the similarity of two sentences $i, j \in S$. Let $f_i : 2^S \to \mathbb{R}_+$ be a set function defined by $f_i(X) = \sum_{j \in X} a_{i,j}$. Finally define $f : 2^S \to \mathbb{R}_+$ by $f(X) = \sum_{i \in S} \min\{f_i(X), \gamma\}$, where $\gamma > 0$ is a parameter. Then f is a monotone submodular function. Intuitively, f measures how well a summary X captures the entire document and the parameter γ prevents that a specific sentences is captured too much.

What is a cost of a summary? The most natural cost is the total length of a summary; i.e, the sum of lengths of sentences in a summary. Thus the task is finding a summary X with, say, at most 300 words, that maximizes the "quality" $f(X)$. This setting could be formulated as monotone submodular function maximization subject to a knapsack constraint, in which the cost of each sentence equals its length.

Example 3.1.5 (Sensor Placement [104])**.** Let us consider the following sensor placement in a water network. We are given a set of sensors S in a water network. We are also given a set E of possible random scenarios of pollution. In each scenario, pollution starts at some place $s \in S$ at time 0, and gradually spreads out as time goes. We can activate sensors, and if pollution reaches to an active sensor then we can detect the pollution. Let us denote the detection time of a sensor s in a scenario e by $z(s, e)$ and let $z_\infty = \max_{e \in E, s \in S} z(s, e)$. The task is to minimize the

average detection time over all the pollution scenario using at most k sensors. This boils down to submodular maximization in which the objective function is

$$f(X) = \mathbf{E}_{e \in E}[z_\infty - \min_{s \in X} z(s, e)]$$

for a set X of active sensors.

The following two classic problems are also included as special cases.

Example 3.1.6 (Maximum Coverage)**.** Monotone submodular function maximization encompasses *maximum coverage*. To see this, let f be a coverage function associated with a bipartite graph $G = (S, T; E)$. The maximum coverage problem is to find a set $X \subseteq S$ of size at most k with neighbor of maximum size. Since f is monotone and submodular (see Example 2.1.1), maximum coverage is a special case of monotone submodular function maximization (subject to a cardinality constraint).

Example 3.1.7 (Set Cover)**.** As its name suggests, submodular cover generalizes *set cover*. Let f and $G = (S, T; E)$ be the same as in the previous example. The task of set cover is to find a set of $X \subseteq S$ of minimum size such that $\Gamma(X) = T$. One can see that the set cover is a special case of submodular cover with $c(X) = |X|$ and $\alpha = |T|$.

In this dissertation, we assume that a submodular function is given via an *evaluation oracle*, unless explicitly stated otherwise.

3.1.1 Monotone Submodular Function Maximization

We review standard methods for monotone submodular function maximization.

Cardinality Constraint Nemhauser and Wolsey [128] proposed that the greedy algorithm for a cardinality constraint. The algorithm is extremely simple. We start with $X = \emptyset$. As long as $|X| < k$ we find an element of maximum marginal increase

$$s \in \operatorname*{argmax}_{t \in S \setminus X} f(t \mid X)$$

and update $X \leftarrow X \cup s$. They showed that this greedy algorithm achieves $(1-1/\mathrm{e})$-approximation.

Now we will follow their proof since the proof contains a "template" for obtaining $(1 - 1/\mathrm{e})$-approximation in monotone submodular function maximization in the simplest form. The readers will find that this "template" appears in various form throughout this dissertation. Let $s_1, \ldots, s_k$ be the elements chosen by the greedy algorithm in this order and define $X_i := \{s_1, \ldots, s_i\}$ for $i = 1, \ldots, k$. We also define $X_0 := \emptyset$. Let us fix an optimal solution X^* and denote $\mathrm{OPT} := f(X^*)$.

The first step is to show a lower bound of the marginal gain in every iteration.

Lemma 3.1.8. *For $i = 1, \ldots, k$,*

$$f(s_i \mid X_{i-1}) \geq \frac{1}{k} \left(\mathrm{OPT} - f(X_{i-1})\right). \tag{3.1}$$

Proof. Let us denote $\ell = |X^* \setminus X_{i-1}|$ and $X^* \setminus X_{i-1} = \{s_1^*, \dots, s_\ell^*\}$. Then by monotonicity and diminishing return property,

$$\begin{aligned}
\mathrm{OPT} - f(X_{i-1}) &\le f(X^* \cup X_{i-1}) - f(X_{i-1}) = \sum_{j=1}^{\ell} f(s_j^* \mid X_{i-1} \cup \{s_1^*, \dots, s_{j-1}^*\}) \\
&\le \sum_{j=1}^{\ell} f(s_j^* \mid X_{i-1}) && \text{(by the diminishing return)} \\
&\le \sum_{j=1}^{\ell} f(s_i \mid X_{i-1}) && \text{(by the definition of } s_i\text{)} \\
&\le k f(s_i \mid X_{i-1}). && \text{(since } \ell \le |X^*| \le k.\text{)}
\end{aligned}$$

Dividing the both sides by k finishes the proof. □

Roughly speaking, once we get an inequality like (3.1), obtaining $(1 - 1/\mathrm{e})$-approximation is a routine.

Lemma 3.1.9. *For $i = 0, \dots, k$,*

$$\mathrm{OPT} - f(X_i) \le \left(1 - \frac{1}{k}\right)^i (\mathrm{OPT} - f(X_0)). \tag{3.2}$$

Proof. If $i = 0$ the statement is trivial. If $i > 0$,

$$\begin{aligned}
\mathrm{OPT} - f(X_i) &= \mathrm{OPT} - f(X_{i-1}) - f(s_i \mid X_{i-1}) \\
&\le \mathrm{OPT} - f(X_{i-1}) - \frac{1}{k}(\mathrm{OPT} - f(X_{i-1})) = \left(1 - \frac{1}{k}\right)(\mathrm{OPT} - f(X_{i-1}))\,,
\end{aligned}$$

where the inequality follows from (3.1). Iterative applications of this complete the proof. □

Since $f(X_0) = f(\emptyset) = 0$, (3.2) yields

$$f(X_k) \ge \left[1 - \left(1 - \frac{1}{k}\right)^k\right] \mathrm{OPT} \ge (1 - 1/\mathrm{e})\,\mathrm{OPT},$$

where in the last inequality we use the well-known inequality $1 + x \le \mathrm{e}^x$ $(x \in \mathbb{R})$. Thus the output of the greedy algorithm, namely X_k, achieves $(1 - 1/\mathrm{e})$-approximation.

Although the greedy algorithm of Nemhauser and Wolsey [128] looks quite simple, there exists heuristic speeding up called *lazy evaluation* [121]. The lazy evaluation does not improve the worst-case time complexity, but dramatically improves the running time in practice.

Algorithm 3.1 Greedy Algorithm for Cardinality Constraint [128] + Lazy Evaluation [121]

1: Initialize $X \leftarrow \emptyset$. Let Q be a max priority queue.
2: For each $s \in S$, let $\Delta(s) \leftarrow f(s)$ and enqueue s with key $\Delta(s)$.
3: **while** $|X| < k$:
4: Dequeue an element s from Q, and let $\Delta(s) \leftarrow f(s \mid X)$.
5: If $\Delta(s) \ge \max_{t \in Q} \Delta(t)$, then update $X \leftarrow X \cup s$. Otherwise, enqueue s again with key $\Delta(s)$.
6: **return** X

Knapsack Constraint Sviridenko [159] proved that the following modified greedy algorithm achieves $(1-1/\mathrm{e})$-approximation for a knapsack constraint. It starts with an initial feasible solution $X = X_0$. As long as $w(X) < 1$, we find an element maximizing the cost-gain ratio

$$s \in \operatorname*{argmax}_{t \in S \setminus X} \frac{f(t \mid X)}{w(t)}.$$

If $w(X) + w(s) \leq 1$ then update $X \leftarrow X \cup s$. Otherwise we remove s from the ground set S.

Obviously the performance of this algorithm depends on an initial solution X_0. Sviridenko [159] showed that if we try *every* initial feasible solution X_0 of cardinality at most three, at least one initial solution gives $(1-1/\mathrm{e})$-approximation. Since there exist $O(n^3)$ initial solutions and each execution of greedy algorithm makes $O(n^2)$ queries to a value oracle, the overall algorithm makes $O(n^5)$ queries.

Matroid Constraint Algorithms for matroid constraint are generally quite complicated. Here we review the *continuous greedy algorithm* developed by Vondrák and his coauthors. At a very high level, it solves the following relaxation problem:

$$\begin{aligned} \text{maximize} \quad & F(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{x} \in P(\mathbf{M}), \end{aligned} \tag{3.3}$$

where F is the multilinear extension of f. Their algorithm actually find a point $\boldsymbol{x}$ in the matroid base polytope $B(\mathbf{M})$ such that $F(\boldsymbol{x})$ is at least $1-1/\mathrm{e}$ times the optimal value of (3.3).

To this end, the algorithm generates a sequence $\{\boldsymbol{x}^t\}_{t\in[0,1]}$ inside the matroid polytope $P(\mathbf{M})$. Formally, the sequence $\{\boldsymbol{x}^t\}$ is a solution of the following differential equation

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} &= \operatorname*{argmax}_{\boldsymbol{v}\in B(\mathbf{M})} \langle \nabla F(\boldsymbol{x}), \boldsymbol{v} \rangle, \\ \boldsymbol{x}^0 &= \mathbf{0}. \end{aligned}$$

At any fixed time $t \in [0,1]$, one can evaluate the velocity vector by solving $\max_{\boldsymbol{v}\in B(\mathcal{M})} \langle \nabla F(\boldsymbol{x}^t), \boldsymbol{v} \rangle$, which is easily done by Edmonds' greedy algorithm. Calinescu et al. [31] showed that $\boldsymbol{x}^1$ satisfies the condition mentioned above. Note that to obtain a "real" algorithm, we need to discretize time steps and control numerical errors. Calinescu et al. [31] also provide such an algorithm, which is quite complicated though.

Lastly, we have to convert a fractional solution $\boldsymbol{x} \in B(\mathbf{M})$ to an independent set X. This task is called *rounding* and several algorithms are available. Here we review two rounding algorithms: pipage rounding [31] and swap rounding [43].

The pipage rounding first computes the minimal tight set T (see Section 2.2) that contains at least two fractional components i, j . Let $\boldsymbol{x}_1 = \boldsymbol{x} + \hat{c}(\boldsymbol{x}, i, j)(\mathbf{e}_i - \mathbf{e}_j)$ and $\boldsymbol{x}_2 = \boldsymbol{x} + \hat{c}(\boldsymbol{x}, j, i)(\mathbf{e}_j - \mathbf{e}_i)$, where $\hat{c}(\boldsymbol{x}, \cdot, \cdot)$ is the exchange capacity (see Section 2.2). Let us express $\boldsymbol{x} = \lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2$. With probability λ, we replace $\boldsymbol{x}$ by $\boldsymbol{x}_1$ and otherwise by $\boldsymbol{x}_2$. By this update, either (i) $x(i)$ or $x(j)$ becomes integral, or (ii) we find a new tight set T' (and repeat this with $T \cap T'$). Computing the minimal tight set T and the exchange capacities requires submodular function minimization. The overall algorithm can be implemented to run in $O(n^7)$ time.

On the other hand, the swap rounding is based on the simultaneous exchange property of bases. In the swap rounding, we assume that a given fractional vector $\boldsymbol{x}$ is represented as a convex combination of extreme points; i.e., we know that $\boldsymbol{x} = \sum_{i=1}^{k} \lambda_i \mathbf{1}_{B_i}$, where B_i is a base of $\mathbf{M}$. This can be done by solving matroid polytope membership. In fact, many algorithms maintain $\boldsymbol{x}$ as such a convex combination and in such a case we can simply skip this step. For

the sake of simplicity, we assume that $k = 2$ in what follows, but a general case can be treated by repeating the algorithm for $k = 2$. We first find $i \in B_1 \setminus B_2$ and $j \in B_2 \setminus B_1$ such that both $B_1' := (B_1 \setminus i) \cup j$ and $B_2' := (B_2 \setminus j) \cup i$ are bases. Then we replace B_i by B_1' with probability $\lambda_2/(\lambda_1 + \lambda_2)$ and otherwise replace B_2 by B_2'. Repeating yields a single base $B = B_1 = B_2$, and the algorithm outputs B. The algorithm runs in $O(n^8)$ time including finding a convex combination and $O(n^3)$ time without finding a convex combination.

Approximation Hardness The factor $1 - 1/\mathrm{e}$ cannot be improved even for a cardinality constraint. Nemhauser, Wolsey, and Fisher [129] proved that any algorithm that invokes polynomially many queries to a value oracle of f cannot achieve an approximation ratio better than $1 - 1/\mathrm{e}$. For an explicitly represented submodular function, Feige [61] proved that the approximation factor of $1 - 1/\mathrm{e}$ cannot be improved even for maximum coverage unless P = NP.

3.1.2 Submodular Cover

Greedy Algorithm Submodular cover was introduced by Wolsey [173] as a generalization of set cover. He described a greedy algorithm for $c(X) = |X|$. Later, Wan et al. [167] provided an approximation algorithm for a general integral c and f.

The idea of their algorithms is similar to the algorithm of Sviridenko [159] for monotone submodular maximization subject to a knapsack constraint. We initialize $X = \emptyset$. As long as $c(X) < \alpha$, we find an element maximizing the cost-gain ratio

$$s \in \operatorname*{argmax}_{t \in S \setminus X} \frac{f(t \mid X)}{c(t \mid X)}.$$

Then update $X \leftarrow X \cup s$. Wolsey [173] showed that if $c(X) = |X|$, this greedy algorithm yields $(1 + \log \frac{d}{\beta})$-approximation, where $d = \max_s f(s)$ is the maximum value of f over all singletons, and $\beta = \min\{f(s \mid X) : X \subseteq S, s \in S, f(s \mid X) > 0\}$ is the minimum value of the positive increments of f in the feasible region. Wan et al. [167] proved that the greedy algorithm achieves $\rho H(d)$-approximation, where

$$\rho = \min_{X^* : \text{an optimal solution}} \frac{\sum_{s \in X^*} c(s)}{c(X^*)} \tag{3.4}$$

is the curvature of c and $H(d) = 1 + 1/2 + \cdots + 1/d$ is the dth harmonic number. Note that Wan et al. [167] considered the case in which c and f are integer valued, thus $\beta \geq 1$.

Approximation Hardness For set cover, we cannot obtain $o(\log|T|)$-approximation unless NP $\subseteq$ DTIME$(n^{O(\log \log n)})$ [61]. The approximation ratio of the above greedy algorithm matches this ratio, since we have $d \leq |T|$ and $\beta \geq 1$ for set cover.

3.2 Optimal Budget Allocation: Motivating Problem

The aforementioned submodular models are based on the submodularity of a set function, a function defined on 2^S. However, we often encounter problems that cannot be captured by a set function. Let us give a motivating example, *optimal budget allocation problem* [5].

In this problem, we want to allocate a given amount of budget among ad sources to maximize the number of "influenced" customers. We model relation between ad sources and customers as a bipartite graph $G = (S, T; E)$ in which S is the set of ad sources, T is the population of customers, and E is an edge set. An edge between an ad source s and a customer t indicates

that s may influence t with some probability that depends on the budget allocated to s. Each source node s has a capacity $c(s) \in \mathbb{Z}_+$ and probabilities $p_s^{(i)} \in [0,1]$ for $i = 1, \dots, c(s)$. Each source node s will be allocated a budget $b(s) \in \{0, 1, \dots, c(s)\}$ such that $\sum_{s \in S} b(s) \leq B$, where $B \in \mathbb{Z}_+$ denotes a *total budget*. If a source node s is allocated a budget of $b(s)$, the node s makes $b(s)$ independent trials to activate each neighboring target node t. The probability that t is activated by s in the ith trial is $p_s^{(i)}$. That is, the probability that t becomes active is

$$1 - \prod_{s \in \Gamma(t)} \prod_{i=1}^{b(s)} (1 - p_s^{(i)}), \tag{3.5}$$

where $\Gamma(t)$ denotes the set of source nodes that is adjacent to t. The objective of this model is to distribute the budget among the source nodes respecting the capacities of nodes, and to maximize the expected number of target nodes that become active.

Problem 3.2.1 (Optimal Budget Allocation [5])**. Given:** a bipartite graph $G = (S, T; E)$, a capacity vector $\boldsymbol{c} \in \mathbb{Z}_+^S$, probabilities $p_s^{(i)}$ $(i = 1, \dots, c(s))$ for each $s \in S$, and a total budget $B \in \mathbb{Z}_+$.

$$\begin{aligned} \text{maximize} \quad & \sum_{t \in T} \left[1 - \prod_{s \in \Gamma(t)} \prod_{i=1}^{b(s)} (1 - p_s^{(i)}) \right] \\ \text{subject to} \quad & \mathbf{0} \leq \boldsymbol{b} \leq \boldsymbol{c}, b(S) \leq B, \boldsymbol{b} \in \mathbb{Z}_+^S. \end{aligned}$$

In this problem, we have to decide *how much* budget should be set aside for each ad source, and hence set functions cannot capture the problem.

3.3 Beyond Set Functions: Submodularity over the Integer Lattice

The optimal budget allocation problem prompts us to generalize the submodularity and the diminishing return property to functions defined on the *integer lattice* $\mathbb{Z}_+^S$.

The most natural generalization of the diminishing return property to a function $f : \mathbb{Z}_+^S \to \mathbb{R}_+$ is the following inequality:

$$f(\boldsymbol{x} + \mathbf{e}_s) - f(\boldsymbol{x}) \geq f(\boldsymbol{y} + \mathbf{e}_s) - f(\boldsymbol{y}) \tag{3.6}$$

for $\boldsymbol{x} \leq \boldsymbol{y}$ and $s \in S$. If f satisfies (3.6), then f also satisfies the following *lattice submodular inequality*:

$$f(\boldsymbol{x}) + f(\boldsymbol{y}) \geq f(\boldsymbol{x} \vee \boldsymbol{y}) + f(\boldsymbol{x} \wedge \boldsymbol{y}) \tag{3.7}$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_+^S$, where $\vee$ and $\wedge$ are the coordinate-wise max and min operations, respectively. While the submodularity and the diminishing return property are equivalent for set functions, this is not the case for functions over the integer lattice; the diminishing return property (3.6) is stronger than the lattice submodular inequality (3.7). We say that f is *lattice submodular* if f satisfies (3.7), and if f further satisfies (3.6) we say that f is *diminishing return submodular* (*DR-submodular* for short).

Remark 3.3.1. Note that our definitions of submodularity on the integer lattice are not new. Indeed, the lattice submodularity is a special case of submodularity on a distributive lattice [66]. Also, it is not surprising that DR-submodular functions were also studied in various areas, due to its similarity to diminishing return property. In economics, Milgrom and Strulovici [120] discussed functions defined over the integer lattice, which they call "multi-unit valuations", and argued lattice submodularity and coordinate-wise concavity. DR-submodular functions include $\mathrm{M}^\natural$-concave functions in discrete convex analysis [124]. Shioura [149] studied maximization of $\mathrm{M}^\natural$-concave functions subject to certain constraints. Perhaps, the work most related to our interest here is one of Kaparalov, Post, and Vondrák [96], in which they call DR-submodular functions "diminishing return functions". They used DR-submodularity for proving approximation hardness for another submodular optimization problem and did not discuss maximization.

One can also define DR-submodular functions as lattice submodular functions with coordinate-wise concavity. We say a function $f : \mathbb{Z}_+^S \to \mathbb{R}$ to be *coordinate-wise concave* if

$$f(\boldsymbol{x} + 2\mathbf{e}_s) - f(\boldsymbol{x} + \mathbf{e}_s) \leq f(\boldsymbol{x} + \mathbf{e}_s) - f(\boldsymbol{x}) \tag{3.8}$$

for each $s \in S$.

Lemma 3.3.2. *A function $f : \mathbb{Z}^S \to \mathbb{R}$ is DR-submodular if and only if f is lattice submodular and coordinate-wise concave.*

Proof. (*If part*) Let us assume that f is lattice submodular and coordinate-wise concave. Let us take $\boldsymbol{x}$ and $\boldsymbol{y}$ with $\boldsymbol{x} \leq \boldsymbol{y}$, and fix $s \in S$. We can write $\boldsymbol{y} = \boldsymbol{x} + \sum_{t \in S^+} k_t \mathbf{e}_t$, where $S^+ := \mathrm{supp}^+(\boldsymbol{y} - \boldsymbol{x})$ and $k_t := y(t) - x(t)$ for each $t \in S^+$. If $s \notin S^+$, then

$$\begin{aligned}
&f(\boldsymbol{x} + \mathbf{e}_s) + f(\boldsymbol{y}) \\
&\quad \geq f((\boldsymbol{x} + \mathbf{e}_s) \vee \boldsymbol{y}) + f((\boldsymbol{x} + \mathbf{e}_s) \wedge \boldsymbol{y}) && \text{(by the lattice submodularity)} \\
&\quad = f(\boldsymbol{y} + \mathbf{e}_s) + f(\boldsymbol{x}), && \text{(since } \boldsymbol{x} \leq \boldsymbol{y} \text{ and } s \notin S^+\text{)}
\end{aligned}$$

which implies $f(\boldsymbol{x} + \mathbf{e}_s) - f(\boldsymbol{x}) \geq f(\boldsymbol{y} + \mathbf{e}_s) - f(\boldsymbol{y})$.

If $s \in S^+$, we have

$$\begin{aligned}
&f(\boldsymbol{x} + \mathbf{e}_s) - f(\boldsymbol{x}) \\
&\quad \geq f(\boldsymbol{x} + k_s\mathbf{e}_s + \mathbf{e}_s) - f(\boldsymbol{x} + k_s\mathbf{e}_s) && \text{(by the coordinate-wise concavity)} \\
&\quad \geq f\left(\boldsymbol{x} + \sum_{t \in S^+} k_t\mathbf{e}_t + \mathbf{e}_s\right) - f\left(\boldsymbol{x} + \sum_{t \in S^+} k_t\mathbf{e}_t\right) && \text{(by the lattice submodularity)} \\
&\quad = f(\boldsymbol{y} + \mathbf{e}_s) - f(\boldsymbol{y}).
\end{aligned}$$

(*Only if part*) Since the coordinate-wise concavity is immediate, we will show the lattice submodularity. Let us take $\boldsymbol{x}$ and $\boldsymbol{y}$ arbitrary, and let us write $\boldsymbol{y} = \boldsymbol{x} \wedge \boldsymbol{y} + \sum_{i=1}^{l} k_i \mathbf{e}_{s_i}$, where $s_1, \ldots, s_l$ are all the elements in $\mathrm{supp}^+(\boldsymbol{y} - \boldsymbol{x})$ and $k_i := y(s_i) - x(s_i)$ for $i = 1, \ldots, l$. Then, we have $\boldsymbol{x} \vee \boldsymbol{y} = \boldsymbol{x} + \sum_{i=1}^{l} k_i \mathbf{e}_{s_i}$. Using the DR-submodularity repeatedly, we obtain

$$f\left(\boldsymbol{x} + \sum_{i=1}^{j} k_i\mathbf{e}_{s_i}\right) - f\left(\boldsymbol{x} + \sum_{i=1}^{j-1} k_i\mathbf{e}_{s_i}\right) \leq f\left(\boldsymbol{x} \wedge \boldsymbol{y} + \sum_{i=1}^{j} k_i\mathbf{e}_{s_i}\right) - f\left(\boldsymbol{x} \wedge \boldsymbol{y} + \sum_{i=1}^{j-1} k_i\mathbf{e}_{s_i}\right)$$

for $j = 1, \ldots, l$. Summing up these inequalities yields $f(\boldsymbol{x} \vee \boldsymbol{y}) - f(\boldsymbol{x}) \leq f(\boldsymbol{y}) - f(\boldsymbol{x} \wedge \boldsymbol{y})$, which is the lattice submodular inequality. □

3.3.1 Submodularity in Optimal Budget Allocation

Let us go back to optimal budget allocation. We first show that the objective function of optimal budget allocation satisfies the lattice submodularity.

Lemma 3.3.3. *The objective function of optimal budget allocation satisfies the lattice submodularity.*

Proof. Let us define

$$g_t(\boldsymbol{b}) = \prod_{s \in \Gamma(t)} \prod_{i=1}^{b(s)} (1 - p_s^{(i)})$$

for each $t \in T$. Since the objective function equals $\sum_{t \in T}(1 - g_t(\boldsymbol{x}))$, it suffices to show g_t is lattice supermodular. Let

$$\alpha := g_t(\boldsymbol{x} \wedge \boldsymbol{y}), \beta := \frac{g_t(\boldsymbol{x})}{\alpha}, \gamma := \frac{g_t(\boldsymbol{y})}{\alpha}.$$

We can easily check that $\alpha, \beta, \gamma \in [0, 1]$ and $g_t(\boldsymbol{x} \vee \boldsymbol{y}) = \alpha\beta\gamma$. Then we obtain

$$g_t(\boldsymbol{x} \vee \boldsymbol{y}) + g_t(\boldsymbol{x} \wedge \boldsymbol{y}) - g_t(\boldsymbol{x}) - g_t(\boldsymbol{y}) = \alpha(\beta\gamma + 1 - \beta - \gamma) = \alpha(1 - \beta)(1 - \gamma) \geq 0. \qquad \square$$

In more realistic scenario, we may assume that the influence probabilities are nonincreasing; i.e., for each $s \in S$, we have $p_s^{(1)} \geq p_s^{(2)} \geq \cdots \geq p_s^{(c(s))}$. The property captures the real-world phenomena of marketing. In the budget allocation model, multiple trials to customers can be viewed as discrete-time steps. Thus it is natural to decrease effectiveness of an advertisement with time.

Lemma 3.3.4. *If the influence probabilities are nonincreasing, then the objective function is DR-submodular.*

Proof. By Lemmas 3.3.2 and 3.3.3, it suffices to show the coordinate-wise concavity. Simple algebra yields that

$$f(\boldsymbol{x} + \mathbf{e}_s) - f(\boldsymbol{x}) = p_s^{(x(s)+1)} \sum_{t \in \Gamma(s)} \prod_{v \in \Gamma(t)} \prod_{i=1}^{x(v)} (1 - p_v^{(i)}),$$

$$f(\boldsymbol{x} + 2\mathbf{e}_s) - f(\boldsymbol{x} + \mathbf{e}_s) = (1 - p_s^{(x(s)+1)}) p_s^{(x(s)+2)} \sum_{t \in \Gamma(s)} \prod_{v \in \Gamma(t)} \prod_{i=1}^{x(v)} (1 - p_v^{(i)}).$$

Let $\alpha := \sum_{t \in \Gamma(s)} \prod_{v \in \Gamma(t)} \prod_{i=1}^{x(v)} (1 - p_v^{(i)})$ for simplicity of notations. Then we have

$$\begin{aligned}
&f(\boldsymbol{x} + \mathbf{e}_s) - f(\boldsymbol{x}) - (f(\boldsymbol{x} + 2\mathbf{e}_s) - f(\boldsymbol{x} + \mathbf{e}_s)) \\
&= p_s^{(x(s)+1)} \alpha - p_s^{(x(s)+2)} (1 - p_s^{(x(s)+1)}) \alpha \\
&\geq \alpha (p_s^{(x(s)+2)} - p_s^{(x(s)+2)} (1 - p_s^{(x(s)+2)})) && (\text{since } p_s^{(x(s)+2)} \leq p_s^{(x(s)+1)}) \\
&= \alpha (p_s^{(x(s)+2)})^2 \geq 0 && \square
\end{aligned}$$

Note that the objective function in budget allocation is not DR-submodular in general. For example, if $p_s^{(1)} = 0$ and $p_s^{(2)} = 1$ for some $s \in S$ with nonzero out-degree, it holds that

$$f(2\mathbf{e}_s) - f(\mathbf{e}_s) > f(\mathbf{e}_s) - f(\mathbf{0}),$$

which violates the DR-submodularity (3.6).

3.4 Our Framework

In this dissertation, we consider the following generalized problems of monotone submodular function generalization and submodular cover.

Problem 3.4.1 (Monotone Submodular Function Maximization over the Integer Lattice)**. Given:** a monotone lattice submodular or DR-submodular function $f : \mathbb{Z}_+^S \to \mathbb{R}_+$ (with $f(\mathbf{0}) = 0$), $\boldsymbol{c} \in \mathbb{Z}_+^S$, and $F \subseteq \mathbb{R}_+^S$. .

$$\begin{aligned} &\text{maximize} && f(\boldsymbol{x}) \\ &\text{subject to} && \boldsymbol{x} \in [\mathbf{0}, \boldsymbol{c}]_{\mathbb{Z}} \cap F, \end{aligned}$$

where $[\mathbf{0}, \boldsymbol{c}]_{\mathbb{Z}} = \{\boldsymbol{x} \in \mathbb{Z}^S : \mathbf{0} \leq \boldsymbol{x} \leq \boldsymbol{c}\}$

Problem 3.4.2 (Diminishing Return Submodular Cover (DR-Submodular Cover for short))**. Given:** a subadditive function $c : \mathbb{Z}_+^S \to \mathbb{R}_+$ with $c(\mathbf{0}) = 0$, a DR-submodular function $f : \mathbb{Z}_+^S \to \mathbb{R}_+$ with $f(\mathbf{0}) = 0$, $r \in \mathbb{Z}_+^S$, and $\alpha > 0$.

$$\begin{aligned} &\text{minimize} && c(\boldsymbol{x}) \\ &\text{subject to} && f(\boldsymbol{x}) \geq \alpha, \mathbf{0} \leq \boldsymbol{x} \leq r\mathbf{1}. \end{aligned}$$

As in the set version of monotone submodular function maximization, we consider the following three constraints:

Cardinality Constraint $F = \{\boldsymbol{x} \in \mathbb{R}^S : x(S) \leq r\}$ for a given $r \in \mathbb{Z}_+$.

Polymatroid Constraint F is a polymatroid P.

Knapsack Constraint $F = \{\boldsymbol{x} \in \mathbb{R}_+^S : \boldsymbol{w}^\top \boldsymbol{x} \leq 1\}$ for a given weight vector $\boldsymbol{w} \in \mathbb{R}_+^S$.

By Lemma 3.3.3, optimal budget allocation problem is a special case of a cardinality constraint. A knapsack constraint enables us to consider more realistic scenario, for example, the budget allocation in which the unit costs of each ad source varies. This scenario is useful for allocating budget among ad sources of different scales, e.g., TV, newspaper, and online banner ads.

Before investigating machine learning problems boiling down to our framework, we summarize algorithmic aspects of monotone submodular optimization over the integer lattice below.

Our Algorithms for monotone submodular function maximization over the integer lattice Table 3.1 summarizes our algorithmic contributions for monotone submodular function maximization over the integer lattice.

Table 3.1: Our algorithms for monotone submodular function maximization over the integer lattice

	DR-submodular	lattice submodular
cardinality	$1 - 1/\mathrm{e} - \epsilon$ (polytime, deterministic)	$1 - 1/\mathrm{e} - \epsilon$ (polytime, deterministic)
polymatroid	$1 - 1/\mathrm{e} - \epsilon$ (polytime, expectation)	open
knapsack	$1 - 1/\mathrm{e} - \epsilon$ (polytime, expectation)	$1 - 1/\mathrm{e}$ (pseudopolytime, deterministic)

For the case in which f is DR-submodular (the first column), we devise polynomial time approximation algorithms for all the three constraints. For the case in which f is lattice submodular (the second column), we design a polynomial time approximation algorithm for a

cardinality constraint and a pseudopolynomial time approximation algorithm for a knapsack constraint. All the algorithms achieve approximation factor of $1 - 1/\mathrm{e}$ or $1 - 1/\mathrm{e} - \epsilon$ for any constant $\epsilon > 0$. Some algorithms are *deterministic* (indicated by "deterministic" in Table 3.1); i.e., they always return a feasible solution whose objective value is at least $1 - 1/\mathrm{e}$ or $1 - 1/\mathrm{e} - \epsilon$ fraction of the optimal value. Some algorithms only achieves approximation *in expectation* (indicated by "expectation" in Table 3.1); i.e., their output is a random feasible vector whose *expected* objective value is at least the guaranteed fraction of the optimal value.

Our Algorithm for DR-submodular cover On the other hand, we design a polynomial time bicriteria-approximation algorithm for DR-submodular cover. More precisely, our algorithm takes the additional parameters $0 < \epsilon, \delta < 1$. The output $\boldsymbol{x} \in \mathbb{Z}_+^S$ of our algorithm is guaranteed to satisfy that $c(\boldsymbol{x})$ is at most $(1 + 3\epsilon)\rho\left(1 + \log \frac{d}{\beta}\right)$ times the optimum and $f(\boldsymbol{x}) \geq (1-\delta)\alpha$, where ρ is the curvature of c (see Section 5.1 for the definition), $d = \max_s f(\mathbf{e}_s)$ is the maximum value of f over all the standard unit vectors, and β is the minimum value of the positive increments of f in the feasible region.

3.4.1 Budget Allocation Problem with a Competitor

Submodularity over the integer lattice enables us to extend the budget allocation problem with nonuniform costs to the two-player case. In the model, there is a competitor against an advertiser. The competitor allocates his/her budget to S in advance, and will try to influence target nodes at the same time as the advertiser's trials. Under this situation, the advertiser aims at allocating a budget to source nodes to maximize the expected number of target nodes influenced by his/her trials. We suppose that the trials of the two players are made in a discrete time step: At each time step i, first the competitor makes the ith trial, and then the advertiser makes the ith trial. The trials will be repeated until both budget allocations run out.

Thus each target node t of T has the following three states: inactive, positively active (influenced by an advertiser), and negatively active (influenced by a competitor). Since an advertiser is a follower, we assume that it has a chance to activate positively both an inactive node and a negatively activated node. Note that when a node is already influenced by the competitor, the probability to activate it should be smaller than one to activate an inactive node. In contrast, we suppose that the competitor can activate an inactive node, but not a positively active node. Thus the model is progressive with respect to positively active nodes.

More specifically, the model is defined as follows. For a bipartite graph $G = (S, T; E)$, an advertiser is given a capacity $c(s)$, a cost $w(s)$ and two probabilities $p_s^{(i)}$ and $q_s^{(i)}$ with $q_s^{(i)} \leq p_s^{(i)}$ for $i = 1, \ldots, c(s)$ for each $s \in S$. In addition, a competitor has already allocated his budget to source nodes, which is denoted by $\tilde{b}(s)$ for each source node $s \in S$. The competitor also has probabilities $\tilde{p}_s^{(i)}$ for $i = 1, \ldots, \tilde{b}(s)$ for each s in S. The probability that t is activated by s in the ith trial depends on the status of t as follows. If t is inactive, then the probabilities that t is positively/negatively activated are $p_s^{(i)}$ and $\tilde{p}_s^{(i)}$, respectively. If t is already negatively active, then the probability that t is positively activated is $q_s^{(i)}$. In this setting, we aim at maximizing the expected number of positively active nodes. For $t \in T$ and k with $1 \leq k \leq 1 + \max_{s \in S} \tilde{b}(s)$, let $E_{t,k}$ be the event that t becomes negatively active in the kth trial (when $k = 1 + \max_{s \in \Gamma(t)} \tilde{b}(s)$, $E_{t,k}$ means the event that t never become negatively active). Conditioned on $E_{t,k}$, the probability that t becomes positively active is

$$f_{t,k}(\boldsymbol{b}) = 1 - \prod_{s \in \Gamma(t)} \prod_{i=1}^{\min\{b(s), k-1\}} \left(1 - p_s^{(i)}\right) \prod_{i=k}^{b(s)} \left(1 - q_s^{(i)}\right).$$

The probability that t becomes positively active equals $\sum_k \lambda_{t,k} f_{t,k}(\boldsymbol{b})$, where $\lambda_{t,k} := \Pr(E_{t,k})$. Therefore, the expected number of positively active nodes, denoted by $f(\boldsymbol{b})$, is equal to $\sum_{t \in T} \sum_k \lambda_{t,k} f_{t,k}(\boldsymbol{b})$. Similarly to Lemma 3.3.3, $f_{t,k}$ is monotone and lattice submodular for any t and k. Since f is a nonnegative linear combination of $f_{t,k}$'s, f is also monotone and lattice submodular. Thus this problem can be reduced to the monotone submodular function maximization over the integer lattice.

3.4.2 Generalized Sensor Placement Model

We can also generalize the existing submodular models in machine learning in a similar way. Here we describe a generalization of a sensor placement model in Example 3.1.5. Suppose that we have several types of sensors with various energy levels, say, $0, \ldots, r$. Let us denote $x(s)$ the energy level of a sensor s and $\boldsymbol{x} \in \mathbb{Z}_+^S$ be the corresponding integral vector. In this generalized model, a sensor may fail to detect pollution. Precisely, when the pollution reaches a sensor s, the probability that we can detect it is $1 - (1-p)^{x(s)}$, where $p \in [0,1]$. In other words, by spending unit energy, we obtain an extra chance of detecting the pollution with probability p. For each event $e \in E$, let s_e be the first sensor where the pollution is detected in that injection event. Note that s_e is a random variable. Then, we define $f : \mathbb{Z}_+^S \to \mathbb{R}_+$ as follows:

$$f(\boldsymbol{x}) = \mathbf{E}_{e \in E} \mathbf{E}_{s_e}[z_\infty - z(s_e, e)],$$

where $z(s_e, e)$ is defined as z_∞ when there is no sensor that managed to detect the pollution.

Lemma 3.4.3. *The function $f : \mathbb{Z}_+^S \to \mathbb{R}_+$ is DR-submodular.*

Proof. Let us define $f_e : \mathbb{Z}_+^S \to \mathbb{R}_+$ as $f_e = \mathbf{E}_{s_e}[z_\infty - z(s_e, e)]$. It suffices to show that f_e is DR-submodular since $f = \mathbf{E}_{e \in E} f_e$.

Let $\tilde{S}$ be the set obtained by copying each sensor $s \in S$ exactly r times, and consider the function $\tilde{f}_e : 2^{\tilde{S}} \to \mathbb{R}_+$ given by $\tilde{f}(X) = f(\boldsymbol{x}^X)$, where $\boldsymbol{x}^X \in \mathbb{Z}_+^S$ be the integral vector such that $x^X(s)$ is the number of copies of s contained in X.

Then, we can regard $\tilde{f}(X)$ for $X \subseteq \tilde{S}$ as the expected cost when each sensor $s \in X$ tries to detect the pollution independently. Note that we have $f(\boldsymbol{x}) = \tilde{f}(X_{\boldsymbol{x}})$, where $X_{\boldsymbol{x}} \subseteq \tilde{S}$ contains $x(s)$ copies of each sensor $s \in S$. We note that $\tilde{f}$ satisfies the submodularity for set functions (see Example 3.1.5). Hence for $s \in S$ and $\boldsymbol{x} \leq \boldsymbol{y}$ with $y(s) \leq r - 1$, we have $f(\boldsymbol{x} + \mathbf{e}_s) - f(\boldsymbol{x}) = \tilde{f}(X_{\boldsymbol{x}+\mathbf{e}_s}) - \tilde{f}(X_{\boldsymbol{x}}) \geq \tilde{f}(X_{\boldsymbol{y}+\mathbf{e}_s}) - \tilde{f}(X_{\boldsymbol{y}}) = f(\boldsymbol{y} + \mathbf{e}_s) - f(\boldsymbol{y})$, which implies the DR-submodularity of f. □

3.4.3 Other Applications

We now present generalizations other than the budget allocation problems and sensor placement. Our framework can naturally extend various applications using submodular set functions into ones allowing multiple choices.

Maximum Coverage Let us first see that our framework includes a generalization of the maximum coverage problem (Example 3.1.6). Here consider covering functions $p_j : \mathbb{Z}_+ \to 2^T$ $(j = 1, \ldots, m)$, each of which is monotone, and we would like to maximize the number of the elements covered by p_j's, i.e., $|\cup_{j=1}^m p_j(x(j))|$, subject to $x(S) \leq k$. Note that the covering function p_j corresponds to the situation where choosing j multiple times makes the covered set larger. This problem clearly generalizes the maximum coverage problem, which corresponds to the case when we can only choose each p_j at most once. It is not difficult to see that the objective function is a monotone lattice submodular function.

Text Summarization We now extend the text summarization model (see Example 3.1.4) to the one that incorporates "confidence" of sentences, i.e., we can choose a sentence in various confidence level like "low", "mid" or "high" rather than just choose or not. As in the maximum coverage, let us introduce a monotone covering function p_j for each sentence j. Then the objective function of the extended model is defined to be the total credit $f(\boldsymbol{x}) = \sum c_i$, where the sum is taken over concept i covered by $\cup_j p_j(x(j))$ and c_i is coverage of a concept i. Again, this objective function is a monotone lattice submodular function.

Facility Location We are given a set S of facilities, and we aim at deciding how large facilities are opened up in order to serve a set of m customers, where we represent scale of facilities as integers $0, 1, \ldots, c$ ("0" means we do not open a facility). If we open up a facility j of scale $x(j)$, then it provides service of value $p_{i,j}(x(j))$ to customer i, where $p_{i,j} : \mathbb{Z}_+ \to \mathbb{R}\,(j = 1, \ldots, m)$ is a given monotone function. We suppose that each customer chooses the opened facility with highest value. That is, when we assign $x(j)$ to each facility j, the total value provided to all customers is given by $f(\boldsymbol{x}) = \sum_{i=1}^{m} \max_{j \in S} p_{i,j}(x(j))$. It turns out f is monotone and lattice submodular.

3.5 Facts on Submodular Function over the Integer Lattice

3.5.1 Useful Lemmas

We review some useful lemmas on submodular function over the integer lattice.

Lemma 3.5.1 (weak diminishing return)**.** *A function $f : \mathbb{Z}^S \to \mathbb{R}$ is monotone and lattice submodular if and only if*

$$f(\boldsymbol{x} \vee k\mathbf{e}_s) - f(\boldsymbol{x}) \geq f(\boldsymbol{y} \vee k\mathbf{e}_s) - f(\boldsymbol{y}), \tag{3.9}$$

for arbitrary $s \in S$, $k \in \mathbb{Z}_+$, $\boldsymbol{x}$ and $\boldsymbol{y}$ with $\boldsymbol{x} \leq \boldsymbol{y}$.

Proof. Suppose that f is monotone and lattice submodular. If $k \leq x(s)$ then both sides are 0. If $x(s) < k \leq y(s)$ then the inequality (3.9) is equivalent to $f(\boldsymbol{x} \vee k\mathbf{e}_s) - f(\boldsymbol{x}) \geq 0$, which is valid by monotonicity. Lastly, if $k > y(s)$ then we have

$$\begin{aligned} f(\boldsymbol{x} \vee k\mathbf{e}_s) + f(\boldsymbol{y}) &\geq f(\boldsymbol{y} \vee k\mathbf{e}_s) + f(\boldsymbol{x} \wedge y(s)\mathbf{e}_s) && \text{(by lattice submodularity)} \\ &\geq f(\boldsymbol{y} \vee k\mathbf{e}_s) + f(\boldsymbol{x}), && \text{(by monotonicity)} \end{aligned}$$

which directly implies (3.9).

Conversely, assume that f satisfies the weak diminishing return property (3.9). To see the monotonicity, it suffices to show $f(\boldsymbol{x} \vee k\mathbf{e}_s) \geq f(\boldsymbol{x})$ for any $\boldsymbol{x} \in \mathbb{Z}^S$, $k \in \mathbb{Z}_+$ and $s \in S$. However, from the weak diminishing return property (3.9), we have $f(\boldsymbol{x} \vee k\mathbf{e}_s) - f(\boldsymbol{x}) \geq f((\boldsymbol{x} \vee k\mathbf{e}_s) \vee k\mathbf{e}_s) - f(\boldsymbol{x} \vee k\mathbf{e}_s) = 0$.

Now we turn to the lattice submodularity. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}^S$ be vectors. We will show $f(\boldsymbol{x} \vee \boldsymbol{y}) - f(\boldsymbol{x}) \leq f(\boldsymbol{y}) - f(\boldsymbol{x} \wedge \boldsymbol{y})$, which implies the lattice submodularity. We can assume that $\mathrm{supp}^+(\boldsymbol{y} - \boldsymbol{x}) \neq \emptyset$, otherwise the lattice submodularity is trivial. Suppose that $\mathrm{supp}^+(\boldsymbol{y} - \boldsymbol{x}) = \{s_1, \ldots, s_\ell\}$. Let $\boldsymbol{x}_0 = \boldsymbol{x}$ and recursively define $\boldsymbol{x}_i := \boldsymbol{x}_{i-1} \vee y(s_i)\mathbf{e}_{s_i}$ $(i = 1, \ldots, \ell)$. Note that $\boldsymbol{x}_\ell = \boldsymbol{x} \vee \boldsymbol{y}$. By the weak diminishing return property (3.9),

$$f(\boldsymbol{x}_{i-1} \vee y(s_i)\mathbf{e}_{s_i}) - f(\boldsymbol{x}_{i-1}) \leq f((\boldsymbol{x}_{i-1} \wedge \boldsymbol{y}) \vee y(s_i)\mathbf{e}_{s_i}) - f(\boldsymbol{x}_{i-1} \wedge \boldsymbol{y})$$

for $i = 1, \ldots, \ell$. Since $(\boldsymbol{x}_{i-1} \wedge \boldsymbol{y}) \vee y(s_i)\mathbf{e}_{s_i} = \boldsymbol{x}_i \wedge \boldsymbol{y}$,

$$f(\boldsymbol{x}_i) - f(\boldsymbol{x}_{i-1}) \leq f(\boldsymbol{x}_i \wedge \boldsymbol{y}) - f(\boldsymbol{x}_{i-1} \wedge \boldsymbol{y}) \quad (i = 1, \ldots, \ell).$$

Summing up these inequalities, we obtain

$$f(\boldsymbol{x}_\ell) - f(\boldsymbol{x}_0) \leq f(\boldsymbol{x}_\ell \wedge \boldsymbol{y}) - f(\boldsymbol{x}_0 \wedge \boldsymbol{y}).$$

Putting $\boldsymbol{x}_0 = \boldsymbol{x}$ and $\boldsymbol{x}_\ell = \boldsymbol{x} \vee \boldsymbol{y}$ completes the proof. □

Lemma 3.5.2. *Let f be a lattice submodular function. For arbitrary $\boldsymbol{x}$ and $\boldsymbol{y}$, we have*

$$f(\boldsymbol{x} \vee \boldsymbol{y}) \leq f(\boldsymbol{x}) + \sum_{s \in \mathrm{supp}^+(\boldsymbol{y}-\boldsymbol{x})} \left(f(\boldsymbol{x} \vee y(s)\mathbf{e}_s) - f(\boldsymbol{x})\right). \tag{3.10}$$

Proof. We prove this lemma by induction on the size of $\mathrm{supp}^+(\boldsymbol{y} - \boldsymbol{x})$. If $|\mathrm{supp}^+(\boldsymbol{y} - \boldsymbol{x})| = 0$, that is, $\boldsymbol{x} \vee \boldsymbol{y} = \boldsymbol{x}$, then (3.10) is trivial. Suppose that there is an index $s \in \mathrm{supp}^+(\boldsymbol{y} - \boldsymbol{x})$. We define $\boldsymbol{y}' = \boldsymbol{y} - y(s)\mathbf{e}_s$. Then $\boldsymbol{y} = \boldsymbol{y}' \vee y(s)\mathbf{e}_s$ and $\boldsymbol{y}' \wedge y(s)\mathbf{e}_s = \mathbf{0}$. By lattice submodularity of $\boldsymbol{x} \vee \boldsymbol{y}'$ and $\boldsymbol{x} \vee y(s)\mathbf{e}_s$,

$$f(\boldsymbol{x} \vee \boldsymbol{y}') + f(\boldsymbol{x} \vee y(s)\mathbf{e}_s) \geq f(\boldsymbol{x} \vee \boldsymbol{y}) + f((\boldsymbol{x} \vee \boldsymbol{y}') \wedge (\boldsymbol{x} \vee y(s)\mathbf{e}_s)).$$

Since

$$(\boldsymbol{x} \vee \boldsymbol{y}') \wedge (\boldsymbol{x} \vee y(s)\mathbf{e}_s) = \boldsymbol{x} \vee (\boldsymbol{y}' \wedge y(s)\mathbf{e}_s) = \boldsymbol{x} \vee \mathbf{0} = \boldsymbol{x},$$

the above inequality implies that

$$f(\boldsymbol{x} \vee \boldsymbol{y}') + f(\boldsymbol{x} \vee y(s)\mathbf{e}_s) - f(\boldsymbol{x}) \geq f(\boldsymbol{x} \vee \boldsymbol{y}).$$

Therefore, applying the induction hypothesis to $\boldsymbol{x}$ and $\boldsymbol{y}'$, we obtain (3.10). □

Lemma 3.5.3. *For a DR-submodular function f,*

$$f(\boldsymbol{x} \vee \boldsymbol{y}) \leq f(\boldsymbol{x}) + \sum_{s \in \{\boldsymbol{y}\} \setminus \{\boldsymbol{x}\}} f(\mathbf{e}_s \mid \boldsymbol{x}) \tag{3.11}$$

for arbitrary $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}^S$, where $\{\boldsymbol{y}\} \setminus \{\boldsymbol{x}\}$ is the multiset such that $i \in S$ is contained $\max\{y(i) - x(i), 0\}$ times.

Proof. By Lemma 3.3.2, f is lattice submodular and coordinate-wise concave. For $s \in \mathrm{supp}^+(\boldsymbol{y} - \boldsymbol{x})$, $f(\boldsymbol{x} \vee y(s)\mathbf{e}_s) - f(\boldsymbol{x}) \leq (y(s) - x(s))f(\mathbf{e}_s \mid \boldsymbol{x})$ by coordinate-wise concavity. Then (3.11) is immediate from (3.10). □

3.5.2 Reducibility of DR-submodular Functions

Lastly we note that a DR-submodular function (defined in a bounded region) can be reduced to a submodular set function on an extended ground set. Let $f : [\mathbf{0}, r\mathbf{1}]_{\mathbb{Z}} \to \mathbb{R}$ be a DR-submodular function. For each $s \in S$, create r copies of s and let $\tilde{S}$ be the set of these copies. For $X \subseteq \tilde{S}$, define $\boldsymbol{x}^X \in \mathbb{Z}_+^S$ be the integral vector such that $x^X(s)$ is the number of copies of s contained in $\tilde{X}$. Then, $\tilde{f}(X) := f(\boldsymbol{x}^X)$ is a submodular set function.

Via this reduction, one can apply approximation algorithms for monotone submodular function maximization and submodular cover, if all the functions involved are DR-submodular. This naive reduction is sometimes useful, but it has a big drawback; the cardinality of $\tilde{S}$ is $r|S|$, which is pseudopolynomial in r. Therefore, the resulting algorithms are only pseudopolynomial time algorithms. Furthermore, this naive reduction does not work for lattice-submodular functions.

Chapter 4

Monotone Submodular Function Maximization over the Integer Lattice

4.1 Overview of this Chapter

In this chapter, we present approximation algorithms for monotone submodular function maximization: Given a monotone lattice submodular or DR-submodular function $f : \mathbb{Z}_+^S \to \mathbb{R}_+$ (with $f(\mathbf{0}) = 0$), $\boldsymbol{c} \in \mathbb{Z}_+^S$, and $F \subseteq \mathbb{R}_+^S$,

$$\begin{aligned} &\text{maximize} && f(\boldsymbol{x}) \\ &\text{subject to} && \boldsymbol{x} \in [\mathbf{0}, \boldsymbol{c}]_{\mathbb{Z}} \cap F. \end{aligned}$$

A technical overview of this chapter is as follows, where $n = |S|$.

Cardinality Constraint for DR-Submodular Function (Section 4.2): The objective is to maximize $f(\boldsymbol{x})$ subject to $\mathbf{0} \leq \boldsymbol{x} \leq \boldsymbol{c}$ and $x(S) \leq r$, where f is monotone DR-submodular, $\boldsymbol{c} \in \mathbb{Z}_+^S$, and $r \in \mathbb{Z}_+$. We design a deterministic $(1 - 1/\mathrm{e} - \epsilon)$-approximation algorithm whose running time is $O(\frac{n}{\epsilon} \log \|\boldsymbol{c}\|_\infty \log \frac{r}{\epsilon})$.

Cardinality Constraint for Lattice Submodular Function (Section 4.3): For cardinality constraints, we also show a $(1 - 1/\mathrm{e} - \epsilon)$-approximation algorithm for a monotone *lattice submodular* function f. This algorithm runs in $O(\frac{n}{\epsilon^2} \log \|\boldsymbol{c}\|_\infty \log \frac{r}{\epsilon} \log \tau)$ time, where τ is the ratio of the maximum value of f to the minimum positive value of f.

Polymatroid Constraint for DR-Submodular Function (Section 4.4): The objective is to maximize $f(\boldsymbol{x})$ subject to $\boldsymbol{x} \in P \cap \mathbb{Z}_+^S$, where f is monotone DR-submodular and P is a polymatroid given via an independence oracle. Our algorithm runs in $\widetilde{O}(\frac{n^3}{\epsilon^5} \log^2 r + n^8)$ time, where r is the maximum value of $x(S)$ for $\boldsymbol{x} \in P$.

Knapsack Constraint for DR-Submodular Function (Section 4.5): The objective is to maximize $f(\boldsymbol{x})$ subject to a single knapsack constraint $\boldsymbol{w}^\top \boldsymbol{x} \leq 1$, where f is monotone DR-submodular and $\boldsymbol{w} \in \mathbb{R}_+^S$. We devise an approximation algorithm with $\widetilde{O}(\frac{n^2}{\epsilon^{18}} \log \frac{1}{w})(\frac{1}{\epsilon})^{O(1/\epsilon^8)}$ running time, which is the first polynomial time algorithm for this problem.

Knapsack Constraint for Lattice-Submodular Function (Section 4.6): Also for a lattice submodular function f, we devise a $(1-1/\mathrm{e})$-approximation algorithm for this problem. The running time is $O(\|\boldsymbol{c}\|_\infty^5 n^4)$, a *pseudopolynomial* in $\boldsymbol{c}$.

4.1.1 Technical Contribution

Although our algorithms share some ideas with the algorithms of [13], [43], we achieve several improvements as well as new ideas, mainly due to the essential difference between set functions and functions over the integer lattice.

Binary Search in the Greedy Phase: In most previous algorithms, the greedy step works as follows; find the direction of maximum marginal gain and move the current solution along the direction with a *unit* step size. However, it turns out that a naive adaptation of this greedy step in the integer lattice only gives a pseudo-polynomial time algorithm. To circumvent this issue, we perform a binary search to determine the step size in the greedy phase. Combined with the decreasing threshold framework, this technique reduces the time complexity significantly.

New Continuous Extensions: To carry out the continuous greedy algorithm, we need a continuous extension of functions over the integer lattice. Note that the *multilinear extension* cannot be directly used since the domain of the multilinear extension is only the hypercube $[0,1]^S$. In this chapter, we propose two different kinds of new continuous extensions of a function over the integer lattice, one of which is for polymatroid constraints and the other is for knapsack constraints. These continuous extensions have similar properties to the multilinear extension when f is DR-submodular, and are carefully designed so that we can round fractional solutions without violating polymatroid or knapsack constraints. To the best of our knowledge, these continuous extensions in $\mathbb{R}_+^S$ have not been known.

Rounding without violating polymatroid and knapsack constraints: It is non-trivial how to round fractional solutions in $\mathbb{R}_+^S$ without violating polymatroid or knapsack constraints. For polymatroid constraints, we show that the rounding can be reduced to rounding in a matroid polytope, and therefore we can use existing rounding methods for a matroid polytope. For knapsack constraints, we design a new simple rounding method based on our continuous extension.

4.1.2 Related Work

Generalized forms of submodularity have been studied in various contexts. Fujishige [66] discusses submodular functions over a distributive lattice and its related polyhedra. In the theory of discrete convex analysis by Murota [124], a subclass of submodular functions over the integer lattice is considered. The maximization problem also has been studied for variants of submodular functions. Shioura [149] investigates the maximization of discrete convex functions. Although the present work focuses on monotone submodular functions, there are a large body of work on maximization of *non-monotone* submodular functions [26]–[28], [62]. Gottschalk and Peis [74] provided a 1/3-approximation algorithm for maximizing a lattice submodular function over (bounded) integer lattice. Recently, *bisubmodular* functions and *k-submodular* functions, other generalizations of submodular functions, have been studied, and approximation algorithms for maximizing these functions can be found in [92], [151], [168].

Algorithm 4.1 For DR-Submodular Function Subject to Cardinality Constraint

Input: $f : \mathbb{Z}_+^S \to \mathbb{R}_+$, $\boldsymbol{c} \in \mathbb{Z}_+^S$, $r \in \mathbb{Z}_+$, and $\epsilon > 0$.
Output: $\boldsymbol{y} \in \mathbb{Z}_+^S$.
1: $\boldsymbol{y} \leftarrow \boldsymbol{0}$.
2: $d \leftarrow \max_{s \in S} f(\mathbf{e}_s)$.
3: **for** $(\theta = d;\ \theta \geq \frac{\epsilon}{r} d;\ \theta \leftarrow \theta(1-\epsilon))$:
4: **for each** $s \in S$:
5: Find maximum $k \leq \min\{c(s) - y(s), r - y(S)\}$ with $f(k\mathbf{e}_s \mid \boldsymbol{y}) \geq k\theta$ by binary search.
6: **if** such k exists :
7: $\boldsymbol{y} \leftarrow \boldsymbol{y} + k\mathbf{e}_s$
8: **return** $\boldsymbol{y}$.

4.1.3 Preliminaries

In this chapter, we use the following form of Chernoff bound.

Lemma 4.1.1 (Relative+Additive Chernoff's bound [13]). *Let $Z_1, \ldots, Z_m$ be independent random variables such that for each i, $Z_i \in [0,1]$. Let $Z = \frac{1}{m}\sum Z_i$ and $\mu = \mathbf{E}[Z]$. Let $0 \leq \alpha < 1$ and $\beta \geq 0$. Then*

$$\Pr(Z > (1+\alpha)\mu + \beta) \leq \exp\left(-\frac{1}{3}m\alpha\beta\right),$$
$$\Pr(Z < (1-\alpha)\mu - \beta) \leq \exp\left(-\frac{1}{2}m\alpha\beta\right).$$

4.2 Cardinality Constraint for DR-Submodular Function

We start with the case of a DR-submodular function. Let $f : \mathbb{Z}_+^S \to \mathbb{R}_+$ be a monotone DR-submodular function. Let $\boldsymbol{c} \in \mathbb{Z}_+^S$ and $r \in \mathbb{Z}_+$. We want to maximize $f(\boldsymbol{x})$ under the constraints $0 \leq \boldsymbol{x} \leq \boldsymbol{c}$ and $x(S) \leq r$. The pseudocode description of our algorithm is shown in Algorithm 4.1.

Note that one can find the maximum k such that $f(k\mathbf{e}_s \mid \boldsymbol{y}) \geq k\theta$ by binary search, because $\frac{f(k\mathbf{e}_s|\boldsymbol{y})}{k}$ is nonincreasing with respect to k due to the DR-submodularity.

Remark 4.2.1. If $\boldsymbol{c} = \boldsymbol{1}$, we can drop the binary search from Algorithm 4.1. Actually the resulting algorithm coincides with the algorithm of [13] for a cardinality constraint.

Lemma 4.2.2. *Let $\boldsymbol{x}^*$ be an optimal solution. When adding $k\mathbf{e}_s$ to the current solution $\boldsymbol{y}$ in Line 7, the average gain satisfies the following.*

$$\frac{f(k\mathbf{e}_s \mid \boldsymbol{y})}{k} \geq \frac{1-\epsilon}{r} \sum_{t \in \{\boldsymbol{x}^*\} - \{\boldsymbol{y}\}} f(\mathbf{e}_t \mid \boldsymbol{y})$$

Proof. Due to DR-submodularity, the marginal values can only decrease as we add elements. If we are adding a vector $k\mathbf{e}_s$ and the current threshold value is θ, then it implies the following inequalities:

Claim 4.2.3. *$f(k\mathbf{e}_s \mid \boldsymbol{y}) \geq k\theta$, and $f(\mathbf{e}_t \mid \boldsymbol{y}) \leq \frac{\theta}{1-\epsilon}$ for any $t \in \{\boldsymbol{x}^*\} - \{\boldsymbol{y}\}$.*

Proof. The first inequality is trivial. The second inequality is also trivial by the DR-submodularity if $\theta = d$. Thus we assume that $\theta < d$, i.e., there was at least one threshold update. Let

$t \in \{\boldsymbol{x}^*\} - \{\boldsymbol{y}\}$, k' be the increment of tth entry in the previous threshold (i.e., $\frac{\theta}{1-\epsilon}$), and $\boldsymbol{y}'$ be the variable $\boldsymbol{y}$ at the moment. Suppose that $f(\mathbf{e}_t \mid \boldsymbol{y}) > \frac{\theta}{1-\epsilon}$. Then $f((k'+1)\mathbf{e}_t \mid \boldsymbol{y}') \geq f(\mathbf{e}_t \mid \boldsymbol{y}) + f(k'\mathbf{e}_t \mid \boldsymbol{y}') > \frac{\theta}{1-\epsilon} + \frac{k'\theta}{1-\epsilon} = \frac{(k'+1)\theta}{1-\epsilon}$, which contradicts the fact that k' is the largest value with $f(k'\mathbf{e}_t \mid \boldsymbol{y}') \geq \frac{k'\theta}{1-\epsilon}$. □

The above inequalities imply that $\frac{f(k\mathbf{e}_s \mid \boldsymbol{y})}{k} \geq (1-\epsilon) f(\mathbf{e}_t \mid \boldsymbol{y})$ for each $t \in \{\boldsymbol{x}^*\} - \{\boldsymbol{y}\}$. Taking the average over these inequalities we get

$$\frac{f(k\mathbf{e}_s \mid \boldsymbol{y})}{k} \geq \frac{1-\epsilon}{|\{\boldsymbol{x}^*\} - \{\boldsymbol{y}\}|} \sum_{t \in \{\boldsymbol{x}^*\} - \{\boldsymbol{y}\}} f(\mathbf{e}_t \mid \boldsymbol{y}) \geq \frac{1-\epsilon}{r} \sum_{t \in \{\boldsymbol{x}^*\} - \{\boldsymbol{y}\}} f(\mathbf{e}_t \mid \boldsymbol{y}) \qquad \square$$

Theorem 4.2.4. *Algorithm 4.1 achieves the approximation ratio of* $1-1/\mathrm{e}-\epsilon$ *in* $O(\frac{n}{\epsilon} \log \|\boldsymbol{c}\|_\infty \log \frac{r}{\epsilon})$ *time.*

Proof. Let $\boldsymbol{y}$ be the output of the algorithm. Without loss of generality, we can assume that $y(S) = r$. Otherwise, consider a modified version of the algorithm in which the threshold is updated until $y(S) = r$. Let $\boldsymbol{y}'$ be the output of this modified algorithm. Since marginal gain of increasing any coordinate of $\boldsymbol{y}$ by one is at most $\epsilon\frac{d}{r}$, we have $f(\boldsymbol{y}') - f(\boldsymbol{y}) \leq \epsilon d \leq \epsilon \mathrm{OPT}$. Therefore, if $\boldsymbol{y}'$ is a $(1-1/\mathrm{e}-\epsilon)$-approximate solution, then $\boldsymbol{y}$ is a $(1-1/\mathrm{e}-2\epsilon)$-approximate solution.

Let $\boldsymbol{y}_i$ be the vector after i steps. Let $k_i\mathbf{e}_{s_i}$ be the vector added in the ith step. That is, $\boldsymbol{y}_i = \sum_{j=1}^{i} k_j \mathbf{e}_{s_j}$. By Lemma 4.2.2, we have

$$\frac{f(k_{i+1}\mathbf{e}_{s_{i+1}} \mid \boldsymbol{y}_i)}{k_{i+1}} \geq \frac{1-\epsilon}{r} \sum_{t \in \{\boldsymbol{x}^*\} - \{\boldsymbol{y}_i\}} f(\mathbf{e}_t \mid \boldsymbol{y}_i).$$

By the diminishing return property, $\sum_{t \in \{\boldsymbol{x}^*\} - \{\boldsymbol{y}_i\}} f(\mathbf{e}_t \mid \boldsymbol{y}_i) \geq f(\boldsymbol{x}^* \vee \boldsymbol{y}_i) - f(\boldsymbol{y}_i)$ holds. Therefore by monotonicity, we have

$$\begin{aligned} f(\boldsymbol{y}_{i+1}) - f(\boldsymbol{y}_i) = f(k_{i+1}\mathbf{e}_{s_{i+1}} \mid \boldsymbol{y}_i) &\geq \frac{(1-\epsilon)k_{i+1}}{r}(f(\boldsymbol{x}^* \vee \boldsymbol{y}_i) - f(\boldsymbol{y}_i)) \\ &\geq \frac{(1-\epsilon)k_{i+1}}{r}(\mathrm{OPT} - f(\boldsymbol{y}_i)). \end{aligned}$$

Hence, we can show by induction that

$$f(\boldsymbol{y}) \geq \left(1 - \prod_i \left(1 - \frac{(1-\epsilon)k_i}{r}\right)\right) \mathrm{OPT}.$$

Since

$$\prod_i \left(1 - \frac{(1-\epsilon)k_i}{r}\right) \leq \prod_i \exp\left(-\frac{(1-\epsilon)k_i}{r}\right) = \exp\left(-\frac{(1-\epsilon)}{r}\sum_i k_i\right) = \mathrm{e}^{-(1-\epsilon)} \leq \frac{1}{\mathrm{e}} + \epsilon,$$

we obtain $(1-1/\mathrm{e}-\epsilon)$-approximation. □

4.3 Cardinality Constraint for Lattice Submodular Function

We now consider the case of a lattice submodular function. Let $f : \mathbb{Z}_+^S \to \mathbb{R}_+$ be a monotone lattice submodular function, $\boldsymbol{c} \in \mathbb{Z}_+^S$, and $r \in \mathbb{Z}_+$. We want to maximize $f(\boldsymbol{x})$ under the constraints $0 \leq \boldsymbol{x} \leq \boldsymbol{c}$ and $x(S) \leq r$.

Algorithm 4.2

Input: $f : \mathbb{Z}_+^S \to \mathbb{R}_+$, $s \in S$, $\theta > 0$, $k_{\max} \in \mathbb{Z}_+$, $\epsilon > 0$.
Output: $0 \leq k \leq k_{\max}$ or **fail**.

Find $k_{\min}$ with $0 \leq k_{\min} \leq k_{\max}$ such that $f(k_{\min}\mathbf{e}_s) > 0$ by binary search.
if no such $k_{\min}$ exists **then fail**.
for $(h = f(k_{\max}\mathbf{e}_s);\ h \geq (1-\epsilon)f(k_{\min}\mathbf{e}_s);\ h = (1-\epsilon)h)$:
  Find the smallest k with $k_{\min} \leq k \leq k_{\max}$ such that $f(k\mathbf{e}_s) \geq h$ by binary search.
  if $f(k\mathbf{e}_s) \geq (1-\epsilon)k\theta$:
    return k
fail.

The main issue is that we cannot find k such that $f(k\chi_e \mid \boldsymbol{x}) \geq k\theta$ by naive binary search anymore. However, we can find k such that $f(k\chi_e \mid \boldsymbol{x}) \geq (1-\epsilon)k\theta$ in polynomial time (if exists). The idea is guessing the value of $f(k\chi_e)$ by iteratively decreasing the threshold and checking whether the desired k exists by binary search. See Algorithm 4.2 for the details.

Lemma 4.3.1. *Algorithm 4.2 satisfies the following:*

(1) Suppose that there exists $0 \leq k^* \leq k_{\max}$ such that $f(k^*\mathbf{e}_s) \geq k^*\theta$. Then, Algorithm 4.2 returns k with $k_{\min} \leq k \leq k_{\max}$ such that $f(k\mathbf{e}_s) \geq (1-\epsilon)k\theta$.

(2) Suppose that Algorithm 4.2 outputs $0 \leq k \leq k_{\max}$. Then, we have $f(k'\mathbf{e}_s) < \max\left\{\frac{f(k\mathbf{e}_s)}{1-\epsilon}, k'\theta\right\}$ for any $k < k' \leq k_{\max}$.

(3) If Algorithm 4.2 does not fail, then the output k satisfies $f(k\mathbf{e}_s) \geq (1-\epsilon)k\theta$.

(4) Let $k_{\min} = \min\{k \mid f(k\mathbf{e}_s) > 0\}$. Then, Algorithm 4.2 runs in $O\left(\frac{1}{\epsilon}\log\frac{f(k_{\max}\mathbf{e}_s)}{f(k_{\min}\mathbf{e}_s)} \cdot \log k_{\max}\right)$ time. If no such $k_{\min}$ exists, then Algorithm 4.2 runs in $O(\log k_{\max})$ time.

Proof. (1) Let $H := \{(1-\epsilon)^\ell f(k_{\max}\mathbf{e}_s) : \ell \in \mathbb{Z}_+, (1-\epsilon)^\ell f(k_{\max}\mathbf{e}_s) \geq (1-\epsilon)f(k_{\min}\mathbf{e}_s)\}$. Let h^* be the (unique) element in H such that $h^* \leq f(k^*\mathbf{e}_s) < \frac{h^*}{1-\epsilon}$. Let k be the minimum integer such that $f(k\mathbf{e}_s) \geq h^*$. Then, we have $k \leq k^*$ and $f(k^*\mathbf{e}_s) < \frac{f(k\mathbf{e}_s)}{1-\epsilon}$. Thus $k\theta \leq k^*\theta \leq f(k^*\mathbf{e}_s) \leq \frac{f(k\mathbf{e}_s)}{1-\epsilon}$, which means that $(1-\epsilon)k\theta \leq f(k\mathbf{e}_s)$.

(2) Let h and h' be the unique elements in H such that $h \leq f(k\mathbf{e}_s) < \frac{h}{1-\epsilon}$ and $h' \leq f(k'\mathbf{e}_s) < \frac{h'}{1-\epsilon}$, respectively. Note that $h \leq h'$. If $h = h'$ then $f(k'\mathbf{e}_s) \leq \frac{f(k\mathbf{e}_s)}{1-\epsilon}$. If $h < h'$, let k_1 be the minimum k_1 such that $f(k_1\mathbf{e}_s) \geq h'$. Then $\frac{f(k'\mathbf{e}_s)}{k'} \leq \frac{f(k_1\mathbf{e}_s)}{(1-\epsilon)k_1} < \theta$, where the last inequality follows from the fact k_1 is examined by the algorithm before k.

(3) and (4) are obvious. □

Basically replacing the binary search in Algorithm 4.1 with Algorithm 4.2 yields the same approximation guarantee. The pseudocode description of our algorithm for maximizing monotone lattice submodular functions under a cardinality constraint is shown in Algorithm 4.3.

Let θ_i be θ in the ith iteration of the outer loop. Let $k_{i,s}$ be k when handling $s \in S$ in the ith iteration of the outer loop. Let $\boldsymbol{y}_{i,s}$ be the vector $\boldsymbol{y}$ right before adding the vector $k_{i,s}\mathbf{e}_s$. For notational simplicity, we define $\boldsymbol{y}_{0,s} = \mathbf{0}$ and $k_{0,s} = 0$ for all $s \in S$. Let $\boldsymbol{\Delta}_{i,s} = (\boldsymbol{x}^* - \boldsymbol{y}_{i,s}) \vee \mathbf{0}$. The following lemma essentially shows that we have $f(\boldsymbol{\Delta}_{i,s}(a)\mathbf{e}_a \mid \boldsymbol{y}_{i,s}) \leq \boldsymbol{\Delta}_{i,s}(a)\theta_i$ for any $i \in \mathbb{N}$ and $s, a \in S$, which is a crucial property to guarantee the approximation ratio of Algorithm 4.3, except that we have several small error terms.

Lemma 4.3.2. *For any $i \in \mathbb{N}$ and $s, a \in S$, we have*

$$f(\boldsymbol{\Delta}_{i,s}(a)\mathbf{e}_a \mid \boldsymbol{y}_{i,s}) \leq \max\left\{\frac{\epsilon}{1-\epsilon}f(k_{i-1,a}\mathbf{e}_a \mid \boldsymbol{y}_{i-1,a}), (\epsilon k_{i-1,a} + \boldsymbol{\Delta}_{i,s}(a))\frac{\theta_i}{1-\epsilon}\right\}.$$

Algorithm 4.3 For Lattice Submodular Function Subject to Cardinality Constraint

Input: $f : \mathbb{Z}_+^S \to \mathbb{R}_+$, $\boldsymbol{c} \in \mathbb{Z}_+^S$, $r \in \mathbb{Z}_+$, $\epsilon > 0$.
Output: $\boldsymbol{y} \in \mathbb{Z}_+^S$.

$\boldsymbol{y} \leftarrow \boldsymbol{0}$.
$d_{\max} \leftarrow \max_{s \in S} f(c(s)\mathbf{e}_s)$.
for $(\theta = d_{\max}; \theta \geq \frac{\epsilon}{r} d_{\max}; \theta \leftarrow \theta(1-\epsilon))$:
  for each $s \in S$:
    Invoke Algorithm 4.2 with $f(\cdot \mid \boldsymbol{y})$, s, θ, $\min\{c(s) - y(s), r - y(S)\}$, and ϵ.
    if Algorithm 4.2 outputs $k \in \mathbb{Z}_+$:
      $\boldsymbol{y} \leftarrow \boldsymbol{y} + k\mathbf{e}_s$
return $\boldsymbol{y}$.

Proof. We define $\boldsymbol{\Delta} = \boldsymbol{\Delta}_{i,s}$, $k' = k_{i-1,a}$, $\boldsymbol{y} = \boldsymbol{y}_{i,a}$, $\boldsymbol{y}' = \boldsymbol{y}_{i-1,a}$, and $\theta' = \theta_{i-1}$ for notational simplicity. We assume $\boldsymbol{\Delta}(a) > 0$ as otherwise the statement is trivial. From (2) and (3) of Lemma 4.3.1, we have

$$f((k' + \boldsymbol{\Delta}(a))\mathbf{e}_a \mid \boldsymbol{y}') \leq \max\Big\{\frac{f(k'\mathbf{e}_a \mid \boldsymbol{y}')}{1-\epsilon}, (k' + \boldsymbol{\Delta}(a))\theta'\Big\}.$$

Since $f((k' + \boldsymbol{\Delta}(a))\mathbf{e}_a \mid \boldsymbol{y}') \geq f(\boldsymbol{\Delta}(a)\mathbf{e}_a \mid \boldsymbol{y}) + f(k'\mathbf{e}_a \mid \boldsymbol{y}')$ holds from lattice submodularity, it follows that

$$f(\boldsymbol{\Delta}(a)\mathbf{e}_a \mid \boldsymbol{y}) \leq \max\Big\{\frac{\epsilon}{1-\epsilon} f(k'\mathbf{e}_a \mid \boldsymbol{y}'), (k' + \boldsymbol{\Delta}(a))\theta' - f(k'\mathbf{e}_a \mid \boldsymbol{y}')\Big\}.$$

Since $f(k'\mathbf{e}_a \mid \boldsymbol{y}') \geq (1-\epsilon)k'\theta'$ from (1) of Lemma 4.3.1, we obtain the claim. □

By taking the sum over all $a \in S$, we have the following:

Corollary 4.3.3.

$$f(k_{i,s}\mathbf{e}_s \mid \boldsymbol{y}_{i,s}) \geq \frac{(1-\epsilon)^2 k_{i,s}}{(1+\epsilon)r}\Big(\sum_{a \in S} f(\boldsymbol{\Delta}_{i,s}(a)\mathbf{e}_a \mid \boldsymbol{y}_{i,s}) - \frac{\epsilon}{1-\epsilon} f(\boldsymbol{y}_{i,s})\Big)$$

Proof. Note that for any $i \in \mathbb{N}$ and $s \in S$, we have $f(k_{i,s}\mathbf{e}_s \mid \boldsymbol{y}_{i,s}) \geq (1-\epsilon)k_{i,s}\theta_i$ from (1) of Lemma 4.3.1. Hence, by summing up the inequality of Lemma 4.3.2 over all $a \in S$, we obtain

$$\begin{aligned}
\sum_{a \in S} f(\boldsymbol{\Delta}_{i,s}(a)\mathbf{e}_a \mid \boldsymbol{y}_{i,s}) &\leq \sum_{a \in S} (\epsilon k_{i-1,a} + \boldsymbol{\Delta}_{i,s}(a)) \frac{\theta_i}{1-\epsilon} + \sum_{a \in S} \frac{\epsilon}{1-\epsilon} f(k_{i-1,a}\mathbf{e}_a \mid \boldsymbol{y}_{i-1,a}) \\
&\leq (1+\epsilon)r \frac{\theta_i}{1-\epsilon} + \frac{\epsilon}{1-\epsilon} f(\boldsymbol{y}_{i,s}) \\
&\leq \frac{(1+\epsilon)r}{(1-\epsilon)^2 k_{i,s}} f(k_{i,s}\mathbf{e}_s \mid \boldsymbol{y}_{i,s}) + \frac{\epsilon}{1-\epsilon} f(\boldsymbol{y}_{i,s}).
\end{aligned}$$

We get the claim by rearranging the inequality. □

Theorem 4.3.4. *Algorithm 4.3 achieves an approximation ratio of* $1 - \frac{1}{\mathrm{e}} - O(\epsilon)$ *in* $O(\frac{n}{\epsilon^2} \log \|\boldsymbol{c}\|_\infty \log \frac{r}{\epsilon} \log \tau)$ *time, where*

$$\tau = \frac{\max_{s \in S} f(c(s)\mathbf{e}_s)}{\min\{f(\boldsymbol{x}) : \boldsymbol{0} \leq \boldsymbol{x} \leq \boldsymbol{c}, f(\boldsymbol{x}) > 0\}}. \tag{4.1}$$

Proof. Let $\boldsymbol{y}$ be the final output of Algorithm 4.3. We can assume that $y(S) = r$. To see this, consider the modified algorithm in which θ is updated until $y(S) = r$. The output $\boldsymbol{y}'$ of this modified algorithm satisfies $y'(S) = r$. By a similar argument as above, we can show that $\sum_{a \in E} f((y' - y)(a)\mathbf{e}_a \mid \boldsymbol{y}) \leq (1+\epsilon) r \frac{\theta_{\min}}{1-\epsilon} + \frac{\epsilon}{1-\epsilon} f(\boldsymbol{y})$, where $\theta_{\min} = \frac{\epsilon}{r} d_{\max}$. This yields $f(\boldsymbol{y}') \leq (1 + O(\epsilon)) f(\boldsymbol{y}) + O(\epsilon)\mathrm{OPT}$. Hence it suffices to show that $\boldsymbol{y}'$ gives $(1 - 1/e - \epsilon)$-approximation.

Let $\alpha = \frac{(1-\epsilon)^2}{1+\epsilon}$ and $\beta = \frac{\epsilon}{1-\epsilon}$. By Corollary 4.3.3 and lattice submodularity of f,

$$\begin{aligned}
f(k_{i,s}\mathbf{e}_s \mid \boldsymbol{y}_{i,s}) &\geq \frac{\alpha k_{i,s}}{r} \Big(\sum_{a \in S} f(\boldsymbol{\Delta}_{i,s}(a)\mathbf{e}_a \mid \boldsymbol{y}_{i,s}) - \beta f(\boldsymbol{y}_i) \Big) \\
&\geq \frac{\alpha k_{i,s}}{r} \Big(f(\boldsymbol{x}^* \vee \boldsymbol{y}_{i,s}) - f(\boldsymbol{y}_{i,s}) - \beta f(\boldsymbol{y}_{i,s}) \Big) \\
&\geq \frac{\alpha k_{i,s}}{r} \Big(\mathrm{OPT} - (1+\beta) f(\boldsymbol{y}_{i,s}) \Big) \\
&\geq \frac{\alpha(1+\beta) k_{i,s}}{r} \Big(\widetilde{\mathrm{OPT}} - f(\boldsymbol{y}_{i,s}) \Big) \\
&= \frac{(1 - O(\epsilon)) k_{i,s}}{r} \Big(\widetilde{\mathrm{OPT}} - f(\boldsymbol{y}_{i,s}) \Big),
\end{aligned}$$

where $\widetilde{\mathrm{OPT}} = \mathrm{OPT}/(1+\beta) = (1-\epsilon)\mathrm{OPT}$. As before, we can show that $f(\boldsymbol{y}) \geq (1 - 1/\mathrm{e} - O(\epsilon))\widetilde{\mathrm{OPT}} = (1 - 1/\mathrm{e} - O(\epsilon))\mathrm{OPT}$. The analysis of time complexity is straightforward. □

4.4 Polymatroid Constraint for DR-Submodular Function

Let P be a polymatroid with a ground set S and the rank function $\rho : 2^S \to \mathbb{Z}_+$. The objective is to maximize $f(\boldsymbol{x})$ subject to $\boldsymbol{x} \in P \cap \mathbb{Z}^S$. In what follows, we denote $\rho(S)$ by r. We assume that P is contained in the interval $[\mathbf{0}, \boldsymbol{c}] := \{\boldsymbol{x} \in \mathbb{R}_+^S : \mathbf{0} \leq \boldsymbol{x} \leq \boldsymbol{c}\}$.

4.4.1 Continuous Extension for Polymatroid Constraints

For $\boldsymbol{x} \in \mathbb{R}^S$, let $\lfloor \boldsymbol{x} \rfloor$ denote the vector obtained by rounding down each entry of $\boldsymbol{x}$. For $a \in \mathbb{R}$, let $\langle a \rangle$ denote the fractional part of a, that is, $\langle a \rangle := a - \lfloor a \rfloor$. For $\boldsymbol{x} \in \mathbb{R}^S$, we define $C(\boldsymbol{x}) := \{\boldsymbol{y} \in \mathbb{R}^S \mid \lfloor \boldsymbol{x} \rfloor \leq \boldsymbol{y} \leq \lfloor \boldsymbol{x} \rfloor + \mathbf{1}\}$ as the hypercube $\boldsymbol{x}$ belongs to.

For $\boldsymbol{x} \in \mathbb{R}^S$, we define $\mathcal{D}(\boldsymbol{x})$ as the distribution from which we sample $\bar{\boldsymbol{x}}$ such that $\bar{x}(s) = \lfloor x(s) \rfloor$ with probability $1 - \langle x(s) \rangle$ and $\bar{x}(s) = \lceil x(s) \rceil$ with probability $\langle x(s) \rangle$, for each $s \in S$. We define the continuous extension $F : \mathbb{R}_+^S \to \mathbb{R}_+$ of $f : \mathbb{Z}_+^S \to \mathbb{R}_+$ as follows. For each $\boldsymbol{x} \in \mathbb{R}_+^S$, we define

$$F(\boldsymbol{x}) := \mathbf{E}_{\bar{\boldsymbol{x}} \sim \mathcal{D}(\boldsymbol{x})}[f(\bar{\boldsymbol{x}})] = \sum_{X \subseteq S} f(\lfloor \boldsymbol{x} \rfloor + \mathbf{1}_X) \prod_{i \in X} \langle x(i) \rangle \prod_{i \notin X} (1 - \langle x(i) \rangle). \tag{4.2}$$

We call this type of continuous extensions *the continuous extension for polymatroid constraints*. Note that F is obtained by gluing the multilinear extension of f restricted to each hypercube. If $f : \{0,1\}^S \to \mathbb{R}_+$ is a monotone submodular function, then it is known that its multilinear extension is monotone and concave along positive directions. We can show similar properties for the continuous extension of a function $f : \mathbb{Z}_+^S \to \mathbb{R}_+$ if f is monotone and DR-submodular.

Lemma 4.4.1. *For a monotone DR-submodular function f, the continuous extension F for the polymatroid constraint is a nondecreasing concave function along any line of direction $\boldsymbol{d} \geq 0$.*

To prove this, we need the following useful fact from convex analysis.

Lemma 4.4.2 (Rockafellar [141], Theorem 24.2)**.** *Let $g : \mathbb{R} \to \mathbb{R}$ be a nonincreasing function and $a \in \mathbb{R}$. Then*

$$f(x) := \int_a^x g(t)\mathrm{d}t$$

is a concave function.

We will use the following notation from [141]; For a function $\phi : \mathbb{R} \to \mathbb{R}$ and $a \in \mathbb{R}$, let us define

$$\phi'_+(a) := \lim_{\epsilon \downarrow 0} \frac{\phi(a) - \phi(a+\epsilon)}{\epsilon} \quad \text{and} \quad \phi'_-(a) := \lim_{\epsilon \uparrow 0} \frac{\phi(a) - \phi(a+\epsilon)}{\epsilon},$$

if the limits exist. For a multivariable function $F : \mathbb{R}^S \to \mathbb{R}$, $\nabla_+ F$ and $\nabla_- F$ are defined in similar manner.

Proof of Lemma 4.4.1. Let $\boldsymbol{p} \in \mathbb{R}^S_+$ and $\phi(\xi) := f(\boldsymbol{p} + \xi \boldsymbol{d})$ for $\xi \geq 0$. To show that ϕ is a nondecreasing concave function, we will find a nonincreasing nonnegative function g such that $\phi(\xi) = \phi(0) + \int_0^\xi g(t)dt$. Note that if ϕ is differentiable for all $\xi \geq 0$, then $g := \phi'$ satisfies the condition. However, ϕ may be nondifferentiable at ξ if $\boldsymbol{p} + \xi \boldsymbol{d}$ contains an integral entry.

Let us denote $\boldsymbol{x} := \boldsymbol{p} + \xi_0 \boldsymbol{d}$ and suppose that $x(i) \in \mathbb{Z}$ for some i. Then one can check that $\phi'_-(\xi_0) = (\nabla F_-|_{\xi_0})^\top \boldsymbol{d}$ and $\phi'_+(\xi_0) = (\nabla F_+|_{\xi_0})^\top \boldsymbol{d}$. For each j,

$$\nabla F_-(j)|_{\xi_0} = \sum_{X : i \notin X} (f(\lfloor \boldsymbol{x} \rfloor - \mathbf{e}_i + \mathbf{1}_{X+i}) - f(\lfloor \boldsymbol{x} \rfloor - \mathbf{e}_i + \mathbf{1}_X)) \prod_{j \in X} \langle x(j) \rangle \prod_{j \notin X, j \neq i} (1 - \langle x(j) \rangle),$$

$$\nabla F_+(j)|_{\xi_0} = \sum_{X : i \notin X} (f(\lfloor \boldsymbol{x} \rfloor + \mathbf{1}_{X+i}) - f(\lfloor \boldsymbol{x} \rfloor + \mathbf{1}_X)) \prod_{j \in X} \langle x(j) \rangle \prod_{j \notin X, j \neq i} (1 - \langle x(j) \rangle).$$

On the other hand, from the diminishing return property, we have

$$f(\lfloor \boldsymbol{x} \rfloor - \mathbf{e}_i + \mathbf{1}_{X+i}) - f(\lfloor \boldsymbol{x} \rfloor - \mathbf{e}_i + \mathbf{1}_X) \geq f(\lfloor \boldsymbol{x} \rfloor + \mathbf{1}_{X+i}) - f(\lfloor \boldsymbol{x} \rfloor + \mathbf{1}_X).$$

Since $\boldsymbol{d} \geq \mathbf{0}$, we have $\phi'_-(\xi_0) \geq \phi'_+(\xi_0)$.

Let $g : \mathbb{R}_+ \to \mathbb{R}$ be an arbitrary function such that $\phi'_-(\xi) \geq g(\xi) \geq \phi'_+(\xi)$ for $\xi \geq 0$. Note that $g(\xi) = \phi'(\xi)$ if ϕ is differentiable at ξ. Then g is nondecreasing and nonnegative. Since $g(\xi) = \phi'(\xi)$ almost everywhere, we have $\phi(\xi) = \phi(0) + \int_0^\xi g(t)dt$. The lemma now directly follows from the nonnegativity of g and Lemma 4.4.2. □

4.4.2 Continuous Greedy Algorithm for Polymatroid Constraint

In this section, we describe our algorithm for the polymatroid constraint, whose pseudocode description is presented in Algorithm 4.6. For a continuous extension F for polymatroid constraints and $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^S_+$, we define $F(\boldsymbol{x} \mid \boldsymbol{y}) = F(\boldsymbol{x} + \boldsymbol{y}) - F(\boldsymbol{y})$.

Proposition 4.4.3. *If $\boldsymbol{x} \in \mathbb{Z}^S_+$, then $F(\boldsymbol{x} \mid \boldsymbol{y}) = \mathbf{E}_{\boldsymbol{z} \sim \mathcal{D}(\boldsymbol{y})}[f(\boldsymbol{x} \mid \boldsymbol{z})]$.*

Proof. $F(\boldsymbol{x} \mid \boldsymbol{y}) = \mathbf{E}_{\boldsymbol{z} \sim \mathcal{D}(\boldsymbol{x}+\boldsymbol{y})}[f(\boldsymbol{z})] - \mathbf{E}_{\boldsymbol{z} \sim \mathcal{D}(\boldsymbol{y})}[f(\boldsymbol{z})] = \mathbf{E}_{\boldsymbol{z} \sim \mathcal{D}(\boldsymbol{y})}[f(\boldsymbol{x} \mid \boldsymbol{z})]$. □

At a high level, our algorithm produces a sequence $\boldsymbol{x}^0 = \mathbf{0}, \ldots, \boldsymbol{x}^{1/\epsilon}$ in P. Given $\boldsymbol{x}^t$, Decreasing-Threshold determines an update direction $\boldsymbol{y}^t$ (here our continuous extension F comes into play), and we update as $\boldsymbol{x}^{t+1} := \boldsymbol{x}^t + \epsilon \boldsymbol{y}^t$. Finally, we perform a rounding algorithm to the fractional solution $\boldsymbol{x}^{1/\epsilon}$ and obtain an integral solution $\bar{\boldsymbol{x}}$.

Given $\boldsymbol{x} \in \mathbb{R}^S_+$, $s \in S$, and $\theta \in \mathbb{R}_+$, Algorithm 4.4 performs binary search to find the largest k such that $F(k\mathbf{e}_s \mid \boldsymbol{x}) \geq \theta$. However, since we can only estimate the value of $F(\cdot)$ in polynomial time, the output k of Algorithm 4.4 satisfies the following weaker conditions:

Algorithm 4.4

Input: $f : \mathbb{Z}_+^S \to \mathbb{R}_+$, $\boldsymbol{x} \in \mathbb{R}_+^S$, $s \in S$, $\theta \in \mathbb{R}_+$, $\alpha, \beta, \delta \in (0,1)$, $k_{\max} \in \mathbb{Z}_+$
Output: $k \in \mathbb{Z}_+$.

$\ell \leftarrow 1$, $u \leftarrow k_{\max}$.
while $\ell < u$:
  $m = \lfloor \frac{\ell+u}{2} \rfloor$.
  $\widetilde{F}(m\mathbf{e}_s \mid \boldsymbol{x}) \leftarrow$ Estimates of $F(m\mathbf{e}_s \mid \boldsymbol{x})$ by averaging $O\left(\frac{\log(k_{\max}/\delta)}{\alpha\beta}\right)$ random samples, respectively.
  if $\widetilde{F}(m\mathbf{e}_s \mid \boldsymbol{x}) \geq m\theta$:
    $\ell \leftarrow m + 1$.
  else
    $u \leftarrow m$.
$k \leftarrow \ell - 1$.
return k.

Algorithm 4.5 Decreasing-Threshold

Input: $f : \mathbb{Z}_+^S \to \mathbb{R}_+$, $\boldsymbol{x} \in \mathbb{R}_+^S$, $\epsilon \in [0,1]$, $P \subseteq \mathbb{R}_+^S$.
Output: A vector $\boldsymbol{y} \in P \cap \mathbb{Z}_+^S$.

$\boldsymbol{y} \leftarrow \mathbf{0}$.
$r \leftarrow \max\{z(S) \mid \boldsymbol{z} \in P\}$.
$d \leftarrow \max_{s \in S} f(\mathbf{e}_s)$.
$N \leftarrow$ the solution to $N = n\lceil \log_{1/1-\epsilon} \frac{N}{\epsilon} \rceil$. Note that $N = O(\frac{n}{\epsilon} \log \frac{n}{\epsilon})$.
for $(\theta = d;\ \theta \geq \frac{\epsilon d}{N};\ \theta \leftarrow \theta(1-\epsilon))$:
  for all $s \in S$:
    Invoke Algorithm 4.4 to find maximum $0 \leq k \leq k_{\max}$ such that $\widetilde{F}(k\mathbf{e}_s \mid \boldsymbol{x} + \epsilon\boldsymbol{y}) \geq k\theta$, where $k_{\max} = \max\{\boldsymbol{y} + k\mathbf{e}_s \in P\}$, with additive error $\alpha_{4.5}$, multiplicative error $\beta_{4.5}$, and failure probability $\delta_{4.5}$.
    if such k exists :
      $\boldsymbol{y} \leftarrow \boldsymbol{y} + k\mathbf{e}_s$.
return $\boldsymbol{y}$.

Lemma 4.4.4. *Algorithm 4.4 satisfies the following properties:*

(1) Suppose that Algorithm 4.4 outputs $k \in \mathbb{Z}_+$. Then, with probability at least $1 - \delta$,

$$F(k\mathbf{e}_s \mid \boldsymbol{x}) \geq (1-\alpha)k\theta - \beta f(k\mathbf{e}_s) \quad \text{and} \quad F((k+1)\mathbf{e}_s \mid \boldsymbol{x}) < (1+\alpha)(k+1)\theta + \beta f((k+1)\mathbf{e}_s).$$

(2) Suppose that Algorithm 4.4 outputs $k \in \mathbb{Z}_+$. Then, with probability at least $1 - \delta$, for any $k < k' \leq k_{\max}$,

$$F(k'\mathbf{e}_s \mid \boldsymbol{x}) < (1+\alpha)k'\theta + \beta f(k_{\max}\mathbf{e}_s).$$

(3) Algorithm 4.4 runs in $O\left(\log k_{\max} \cdot \frac{\log k_{\max}/\delta}{\alpha\beta}\right)$ time.

Proof. (1) For $k \in \mathbb{Z}$, let $\Delta_m = F(m\mathbf{e}_s \mid \boldsymbol{x}) = \mathbf{E}_{\boldsymbol{z} \sim \mathcal{D}(\boldsymbol{x})}[f(m\mathbf{e}_s \mid \boldsymbol{z})]$ and $\widetilde{\Delta}_m$ be its estimation. Note that $0 \leq f(m\mathbf{e}_s \mid \boldsymbol{z}) \leq f(m\mathbf{e}_s)$ for any $\boldsymbol{z}$ in the support of $\mathcal{D}(\boldsymbol{x})$. By Lemma 4.1.1 and the union bound, with probability at least $1 - \delta$, we have

$$|\widetilde{\Delta}_m - \Delta_m| \leq \alpha s_m + \beta f(m\mathbf{e}_s).$$

Algorithm 4.6 Polymatroid Constraint

Input: $f : \mathbb{Z}_+^S \to \mathbb{R}_+, P \subseteq \mathbb{R}_+^S$.
Output: A vector $\bar{\boldsymbol{x}} \in P \cap \mathbb{Z}_+^S$.
1: $\boldsymbol{x} \leftarrow \mathbf{0}$.
2: **for** $(t \leftarrow 1; t \leq \lfloor \frac{1}{\epsilon} \rfloor; t \leftarrow t+1)$:
3: $\quad \boldsymbol{y} \leftarrow$ Decreasing-Threshold$(f, \boldsymbol{x}, \epsilon, P)$.
4: $\quad \boldsymbol{x} \leftarrow \boldsymbol{x} + \epsilon \boldsymbol{y}$.
5: $\bar{\boldsymbol{x}} \leftarrow$ Rounding$(\boldsymbol{x}, P)$.
6: **return** $\bar{\boldsymbol{x}}$.

for every estimation in the process. In what follows, we suppose this happens.

Suppose $\Delta_m < (1-\alpha)(m\theta - \beta f(m\mathbf{e}_s))$. Then, $\widetilde{\Delta}_m < m\theta$ and we reach Line 8. On the other hand, Suppose $\Delta_m \geq (1+\alpha)(m\theta + \beta f(m\mathbf{e}_s))$. Then, $\widetilde{\Delta}_m \geq m\theta$ and we reach Line 6. Hence after the while loop we must have

$$\begin{aligned} \Delta_{\ell-1} &\geq (1-\alpha)((\ell-1)\theta - \beta f((\ell-1)\mathbf{e}_s)) \geq (1-\alpha)(\ell-1)\theta - \beta f((\ell-1)\mathbf{e}_s) \\ \Delta_\ell &< (1+\alpha)(\ell\theta + \beta f(\ell\mathbf{e}_s)) \leq (1+\alpha)\ell\theta + \beta f(\ell\mathbf{e}_s)), \end{aligned}$$

which implies (1).

(2) holds because of (1) and the concavity of F, and (3) is obvious. □

Lemma 4.4.5. *Let $\boldsymbol{x}^*$ be an optimal solution. Given a fractional solution $\boldsymbol{x}$, Decreasing-Threshold produces a new fractional solution $\boldsymbol{y}$ such that $\boldsymbol{x}' := \boldsymbol{x} + \boldsymbol{y}$ satisfies*

$$F(\boldsymbol{x}') - F(\boldsymbol{x}) \geq \epsilon((1-5\epsilon)f(\boldsymbol{x}^*) - F(\boldsymbol{x}'))$$

with probability at least $1 - \epsilon/3$.

Proof. We choose $\alpha_{4.5} = \epsilon$, $\beta_{4.5} = \frac{\epsilon}{N(n+1)}$, and $\delta_{4.5} = \frac{\epsilon}{3N}$. With probability at least $1 - \epsilon/3$, all the binary searches succeed. In what follows, we assume this has happened.

Assume for now that Decreasing-Threshold returns $\boldsymbol{y}$ with $y(S) = r$. If not, we add dummy elements of value 0 so that $y(S) = r$. Let $\boldsymbol{y}_i$ be the vector $\boldsymbol{y}$ after the ith update in the execution of Decreasing-Threshold. Let b_i and k_i be the element of S and the step size chosen in the ith update, respectively; i.e., $\boldsymbol{y}_i = \boldsymbol{y}_{i-1} + k_i \mathbf{e}_{b_i}$. Applying Lemma 2.2.5 to $\boldsymbol{y}$ and $\boldsymbol{x}^*$, we obtain a mapping $\phi : I(\boldsymbol{y}) \to \mathrm{supp}^+(\boldsymbol{x}^*)$. Let $b^*_{i,j} := \phi(b_i, \boldsymbol{y}_{i-1}(b_i) + j)$ for $j = 1, \ldots, k_i$. Then, by construction, $\boldsymbol{y}_{i-1} - \mathbf{e}_{b_i} + \mathbf{e}_{b^*_{i,j}} \in P$ for $j = 1, \ldots, k_i$.

For each i, let θ_i be the threshold used in the ith update and let $\boldsymbol{x}_i = \boldsymbol{x} + \boldsymbol{y}_i$. By (1) of Lemma 4.4.4, we have $F(k_i \mathbf{e}_{b_i} \mid \boldsymbol{x}_{i-1}) \geq (1-\epsilon)k_i\theta_i - \beta_{4.5} f(k_i \mathbf{e}_{b_i}) - \frac{\epsilon d}{N}$, where the last term $\frac{\epsilon d}{N}$ comes from the case that b_i is a dummy element.

For each i and $s \in S$, we define $k^*_{i,s} \in \mathbb{Z}_+$ as the number of times that s appeared in the sequence $b^*_{i,1}, \ldots, b^*_{i,k_i}$. When the vector $k_i \mathbf{e}_{b_i}$ is added, any $s \in S$ have been an candidate element when the threshold was θ_{i-1}. Thus by (2) of Lemma 4.4.4, we have for each $s \in S$,

$$F(k^*_{i,s}\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) \leq (1+\epsilon)k^*_{i,s}\theta_{i-1} + \beta_{4.5} f(k_{\max}\mathbf{e}_s) \leq \frac{k^*_{i,s}}{(1-\epsilon)^2}\theta_i + \beta_{4.5} f(k_{\max}\mathbf{e}_s).$$

Rephrasing this yields

$$\theta_i \geq \frac{(1-\epsilon)^2}{k^*_{i,s}}(F(k^*_{i,s}\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) - \beta_{4.5} f(k_{\max}\mathbf{e}_s)).$$

By averaging on $s \in S$, we have

$$\theta_i \geq \frac{(1-\epsilon)^2}{k_i} \sum_s (F(k^*_{i,s}\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) - \beta_{4.5} f(k_{\max}\mathbf{e}_s)).$$

Combining these inequalities and the fact that $f(\boldsymbol{x}^*) \geq d$, we get

$$\begin{aligned}
F(k_i\mathbf{e}_{b_i} \mid \boldsymbol{x}_{i-1}) &\geq (1-\epsilon)^3 \sum_s (F(k^*_{i,s}\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) - \beta_{4.5} f(k_{\max}\mathbf{e}_s)) - \beta_{4.5} f(k_i\mathbf{e}_{b_i}) - \frac{\epsilon d}{N} \\
&= (1-\epsilon)^3 \sum_s F(k^*_{i,s}\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) - \beta_{4.5}\Big(f(k_i\mathbf{e}_{b_i}) + \sum_s f(k_{\max}\mathbf{e}_s)\Big) - \frac{\epsilon d}{N} \\
&\geq (1-\epsilon)^3 \sum_s F(k^*_{i,s}\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) - \beta_{4.5}\Big(f(\boldsymbol{x}^*) + \sum_s f(\boldsymbol{x}^*)\Big) - \frac{\epsilon d}{N} \\
&\geq (1-\epsilon)^3 \sum_s F(k^*_{i,s}\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) - \frac{2\epsilon}{N} f(\boldsymbol{x}^*)
\end{aligned}$$

Since F is concave along non-negative directions, we have for any $\epsilon > 0$

$$F(\epsilon k_i\mathbf{e}_{b_i} \mid \boldsymbol{x}_{i-1}) \geq \epsilon(1-\epsilon)^3 \sum_s F(k^*_{i,s}\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) - \frac{2\epsilon^2}{N} f(\boldsymbol{x}^*). \tag{4.3}$$

Using the above inequality and the fact that N is an upper bound on the total number of updates, we bound the improvement at each time step as follows:

$$\begin{aligned}
F(\boldsymbol{x}') - F(\boldsymbol{x}) &= \sum_i (F(\boldsymbol{x} + \epsilon\boldsymbol{y}_i) - F(\boldsymbol{x} + \epsilon\boldsymbol{y}_{i-1})) = \sum_i F(\epsilon k_i\mathbf{e}_{b_i} \mid \boldsymbol{x}_{i-1}) \\
&\geq \sum_i \left(\epsilon(1-\epsilon)^3 \sum_s F(k^*_{i,s}\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) - \frac{2\epsilon^2}{N} f(\boldsymbol{x}^*)\right) \\
&= \epsilon(1-\epsilon)^3 \sum_i \sum_s \mathbf{E}_{\boldsymbol{z}\sim\mathcal{D}(\boldsymbol{x}_{i-1})}[f(\boldsymbol{z} + k^*_{i,s}\mathbf{e}_s) - f(\boldsymbol{z})] - \sum_i \frac{2\epsilon^2}{N} f(\boldsymbol{x}^*) \\
&= \epsilon(1-\epsilon)^3 \sum_i \sum_s \mathbf{E}_{\boldsymbol{z}\sim\mathcal{D}(\boldsymbol{x}_{i-1})}[f(\boldsymbol{z} + k^*_{i,s}\mathbf{e}_s) - f(\boldsymbol{z})] - 2\epsilon^2 f(\boldsymbol{x}^*) \\
&\geq \epsilon(1-\epsilon)^3\, \mathbf{E}_{\boldsymbol{z}\sim\mathcal{D}(\boldsymbol{x}')}[f(\boldsymbol{z} \vee \boldsymbol{x}^*) - f(\boldsymbol{z})] - 2\epsilon^2 f(\boldsymbol{x}^*) \qquad \text{(from DR-submodularity)} \\
&\geq \epsilon((1-5\epsilon) f(\boldsymbol{x}^*) - F(\boldsymbol{x}')). \qquad \text{(from monotonicity)}
\end{aligned}$$

□

Lemma 4.4.6. *Decreasing-Threshold runs in time* $O(\frac{n^2 N}{\epsilon^3} \log\frac{n}{\epsilon} \log r \log\frac{rN}{\epsilon})$.

Proof. Algorithm 4.4 takes $O(\frac{nN}{\epsilon^2} \log r \log\frac{rN}{\epsilon})$ time. The outer loop iterates for $O(\frac{1}{\epsilon}\log\frac{n}{\epsilon})$ times, while the inner loop contains the execution of Algorithm 4.4 for n times. □

Claim 4.4.7. *In the end of Algorithm 4.6, we have* $F(\boldsymbol{x}) \geq (1-1/\mathrm{e}-O(\epsilon))\mathrm{OPT}$ *with probability at least* $2/3$ *in* $O(\frac{n^3}{\epsilon^5}\log^3\frac{n}{\epsilon}\log^2 r)$ *time.*

Proof. Define $\Omega := (1-5\epsilon)f(\boldsymbol{x}^*) = (1-5\epsilon)\mathrm{OPT}$. Let $\boldsymbol{x}^t$ be the variable $\boldsymbol{x}$ after the tth iteration. Substituting this in the result of Lemma 4.4.5, we get

$$F(\boldsymbol{x}^{t+1}) - F(\boldsymbol{x}^t) \geq \epsilon(\Omega - F(\boldsymbol{x}^{t+1})).$$

Rephrasing the equation we get

$$\Omega - F(\boldsymbol{x}^{t+1}) \leq \frac{\Omega - F(\boldsymbol{x}^t)}{1+\epsilon}.$$

Now applying induction to this equation, we obtain

$$\Omega - F(\boldsymbol{x}^t) \leq \frac{\Omega}{(1+\epsilon)^t}.$$

Substituting $t = 1/\epsilon$ and rewriting the equation we get the desired approximation ratio:

$$F(\boldsymbol{x}) \geq \left(1 - \frac{1}{(1+\epsilon)^{1/\epsilon}}\right)\Omega \geq (1-1/\mathrm{e})(1-5\epsilon)\mathrm{OPT} = (1-1/\mathrm{e}-O(\epsilon))\mathrm{OPT}.$$

The time complexity is $O(\frac{n^2 N}{\epsilon^4}\log\frac{n}{\epsilon}\log r\log\frac{rN}{\epsilon}) = O(\frac{n^3}{\epsilon^5}\log^3\frac{n}{\epsilon}\log^2 r)$. □

4.4.3 Rounding

We need a rounding procedure that takes a real vector $\boldsymbol{x}$ as the input and returns an integral vector $\bar{\boldsymbol{x}}$ such that $\mathbf{E}[f(\bar{\boldsymbol{x}})] \geq F(\boldsymbol{x})$. There are several rounding algorithm in the $\{0,1\}^S$ case (see Section 3.1.1). However, generalizing these rounding algorithms over integer lattice is a nontrivial task. Here, we show that rounding in the integer lattice can be reduced to rounding in the $\{0,1\}^S$ case.

Suppose that we have a fractional solution $\boldsymbol{x}$. The following lemma implies that we can round $\boldsymbol{x}$ by considering the corresponding matroid polytope.

Lemma 4.4.8. *For $\boldsymbol{x} \in P$, $P \cap C(\boldsymbol{x})$ is a translation of a matroid polytope.*

Proof. Let us consider a polytope $P' := \{\langle \boldsymbol{z}\rangle : \boldsymbol{z} \in P \cap C(\boldsymbol{x})\}$. We can check that P' can be obtained by translating $P \cap C(\boldsymbol{x})$ by $-\lfloor \boldsymbol{x}\rfloor$ and restricting to $[0,1]^S$. Therefore, $P' = \{\boldsymbol{z} \in [0,1]^S : \boldsymbol{z}(X) \leq \rho'(X) \quad \forall X \subseteq S\}$, where $\rho'(X) := \min_{Y\subseteq X}\{(\rho - \lfloor \boldsymbol{x}\rfloor)(Y) + \mathbf{1}(S \setminus Y)\}$. Then we can show that ρ' is the rank function of a matroid by checking the axiom. □

The independence oracle of the corresponding matroid is just the independence oracle of P restricted to $C(\boldsymbol{x})$. Thus, the pipage rounding algorithm for P' yields an integral solution $\bar{\boldsymbol{x}}$ with $\mathbf{E}[f(\bar{\boldsymbol{x}})] \geq F(\boldsymbol{x})$ in strongly polynomial time.

Slightly faster rounding can be achieved by swap rounding. Swap rounding requires that the given fractional solution $\boldsymbol{x}$ is represented by a convex combination of extreme points of the matroid polytope. In our setting, we can represent $\boldsymbol{x}$ as a convex combination of extreme points of $P \cap C(\boldsymbol{x})$, using the algorithm of Cunningham [50]. Then we run the swap rounding algorithm for the convex combination and $P \cap C(\boldsymbol{x})$. The running time of this rounding algorithm is dominated by the complexity of finding a convex combination for $\boldsymbol{x}$, which is $O(n^8)$ time.

Theorem 4.4.9. *Algorithm 4.6 finds an $(1-1/\mathrm{e}-\epsilon)$-approximate solution (in expectation) with high probability in $\widetilde{O}(\frac{n^3}{\epsilon^5}\log^2 r + n^8)$ time.*

4.5 Knapsack Constrant for DR-Submodular Function

In this section, we give an efficient algorithm for the submodular function maximization over the integer lattice under knapsack constraints. The main difficulty of this case is that we cannot obtain a feasible solution by applying the swap rounding or the pipage rounding to a fractional solution of the multilinear extension considered in Section 4.4. To overcome this issue, we introduce another multilinear extension in Section 4.5.1. Then, we describe our continuous greedy algorithm using this extension in Section 4.5.2.

4.5.1 Multilinear extension for knapsack constraints

Let $\boldsymbol{X} : I \to \mathbb{R}_+^S$ be a multiset of vectors (indexed by a set I). We say that $\boldsymbol{X}$ is a *multi-vector* if, for every $i \in I$, $\boldsymbol{X}(i)$ is of the form $k\mathbf{e}_s$, where $k \in \mathbb{R}_+$ and $s \in S$. For $J \subseteq I$, let $\boldsymbol{X}(J) := \sum_{j \in J} \boldsymbol{X}(j) \in \mathbb{R}_+^S$. Let $\boldsymbol{p} \in [0,1]^I$. Then we define

$$F_{\boldsymbol{p}}(\boldsymbol{X}) = \sum_{J \subseteq I} \prod_{i \in J} p(i) \prod_{i \notin J} (1 - p(i)) f(\boldsymbol{X}(J)). \tag{4.4}$$

In this section, $\oplus$ denotes the concatenation. In our setting, we often consider $F_{\boldsymbol{p}}$ for $\boldsymbol{p} = \epsilon\mathbf{1}$, i.e, each element of $\boldsymbol{X}$ is independently sampled in the same probability ϵ. In this case, we will use the shorthand notation $F_\epsilon(\boldsymbol{X}) := F_{\epsilon\mathbf{1}}(\boldsymbol{X})$. In addition, we define $F_\epsilon(k\mathbf{e}_s \mid \boldsymbol{X}) := F_\epsilon(\boldsymbol{X} \oplus k\mathbf{e}_s) - F_\epsilon(\boldsymbol{X})$. Also, we define $F_\epsilon^{\oplus 1}(k\mathbf{e}_s \mid \boldsymbol{X}) := F_{\epsilon\mathbf{1}\oplus 1}(\boldsymbol{X} \oplus k\mathbf{e}_s) - F_\epsilon(\boldsymbol{X})$, where $\epsilon\mathbf{1}$ corresponds to $\boldsymbol{X}$ and 1 corresponds to $k\mathbf{e}_s$. For a multi-vector $\boldsymbol{X}$ and a vector $\boldsymbol{w} \in \mathbb{R}^S$, we define $w(\boldsymbol{X}) = \sum_{k\mathbf{e}_s \in \boldsymbol{X}} kw(s)$ as the sum of weights of vectors in $\boldsymbol{X}$.

Lemma 4.5.1. *Let $f : \mathbb{Z}_+^S \to \mathbb{R}_+$ be a monotone DR-submodular function. Fix a multi-vector $\boldsymbol{X} : I \to \mathbb{Z}_+^S$. Then, the function $F_{\boldsymbol{p}}(\boldsymbol{X})$, as a function of $\boldsymbol{p}$, is a monotone concave function along any line of direction $\boldsymbol{d} \geq 0$.*

Proof. The proof is exactly the same as Lemma 4.4.1 and we omit. □

Lemma 4.5.2. *Let $f : \mathbb{Z}_+^S \to \mathbb{R}_+$ be a monotone DR-submodular function. Then,*

$$F_\epsilon(\boldsymbol{Y} \mid \boldsymbol{X}) \geq F_\epsilon(\boldsymbol{Y} \mid \boldsymbol{X} \oplus \boldsymbol{X}')$$

for any multi-vectors $\boldsymbol{X}$, $\boldsymbol{X}'$, and $\boldsymbol{Y}$.

Proof. Obvious from the DR-submodularity of f. □

We denote by $R_{\boldsymbol{p}}(\boldsymbol{X})$ the (random) vector $\boldsymbol{X}(J)$, where J is obtained by independently adding i with probability $p(i)$ for each $i \in I$. Note that $F_{\boldsymbol{p}}(\boldsymbol{X}) = \mathbf{E}[f(R_{\boldsymbol{p}}(\boldsymbol{X}))]$. We can see $R_{\boldsymbol{p}}(\boldsymbol{X})$ as a randomized rounding of $\boldsymbol{X}$.

For a partition $\Pi = \{I_1, \ldots, I_k\}$ of I and a vector $\boldsymbol{p}$ with $p(I_i) \leq 1$ for each $i \in [k]$, we introduce another randomized rounding $U_{\boldsymbol{p}}(\boldsymbol{X}; \Pi)$ as follows. Namely, we define $U_{\boldsymbol{p}}(\boldsymbol{X}; \Pi)$ as the vector $\sum_{i \in [k]} \boldsymbol{X}(J_i)$, where $J_i = \{j\}$ with probability $\boldsymbol{p}(j)$ for each $j \in I_i$ and $J_i = \emptyset$ with probability $1 - p(I_i)$. The rounding $U_{\boldsymbol{p}}(\boldsymbol{X}; \Pi)$ is as well as $R_{\boldsymbol{p}}(\boldsymbol{X})$:

Lemma 4.5.3. *Let $f : \mathbb{Z}^S \to \mathbb{R}_+$ be a monotone DR-submodular function. Let Π and $\boldsymbol{p}$ as above. Then, for any multi-vector $\boldsymbol{X} : I \to \mathbb{R}_+^S$,*

$$\mathbf{E}[f(U_{\boldsymbol{p}}(\boldsymbol{X}; \Pi))] \geq F_{\boldsymbol{p}}(\boldsymbol{X}).$$

Proof. Let $\boldsymbol{r} = R_{\boldsymbol{p}}(\boldsymbol{X})$ and $\boldsymbol{u} = U_{\boldsymbol{p}}(\boldsymbol{X}; \Pi)$.

We proceed by induction on k. First, assume that $k = 1$ and $I = I_1$. Then,

$$\begin{aligned}
F_{\boldsymbol{p}}(\boldsymbol{X}) = \mathbf{E}[f(\boldsymbol{r})] &= \sum_{J \subseteq I_1} \Pr[J \text{ is chosen}] f(\boldsymbol{X}(J)) \\
&\leq \sum_{J \subseteq I_1} \Pr[J \text{ is chosen}] \sum_{j \in J} f(\boldsymbol{X}(j)) \qquad \text{(by the DR-submodularity of } f\text{)} \\
&= \sum_{j \in I_1} \Pr_J[j \in J] f(\boldsymbol{X}(j)) = \sum_{j \in I_1} p(j) f(\boldsymbol{X}(j)) = \mathbf{E}[f(\boldsymbol{u})]
\end{aligned}$$

Now let $k > 1$. We define $\boldsymbol{r}' = \boldsymbol{r}_{I_1 \cup \cdots \cup I_{k-1}}$ and $\boldsymbol{r}'' = \boldsymbol{r}_{I_k}$. Define $\boldsymbol{u}'$ and $\boldsymbol{u}''$ similarly. Then,

$$\mathbf{E}[f(\boldsymbol{u})] = \mathbf{E}[f(\boldsymbol{u}') + f(\boldsymbol{u}'' \mid \boldsymbol{u}')] \geq \mathbf{E}[f(\boldsymbol{u}') + f(\boldsymbol{r}'' \mid \boldsymbol{u}')] = \mathbf{E}[f(\boldsymbol{u}' + \boldsymbol{r}'')]$$

using the base case for $f(\cdot \mid \boldsymbol{u}')$ on I_k. Then,

$$\mathbf{E}[f(\boldsymbol{u}' + \boldsymbol{r}'')] = \mathbf{E}[f(\boldsymbol{r}'') + f(\boldsymbol{u}' \mid \boldsymbol{r}'')] \geq \mathbf{E}[f(\boldsymbol{r}'') + f(\boldsymbol{r}' \mid \boldsymbol{r}'')] = \mathbf{E}[f(\boldsymbol{r})]$$

using the induction hypothesis for $f(\cdot \mid \boldsymbol{r}'')$ on $I_1 \cup \cdots \cup I_{k-1}$. □

4.5.2 Algorithm

We now describe the sketch of our algorithm for knapsack constraints. Let $\boldsymbol{x}^*$ be an optimal solution. We call an item s *small* if it satisfies

- $x^*(s) f(\mathbf{e}_s) \leq \epsilon^6 \mathrm{OPT}$ and
- $w(s) \leq \epsilon^4$.

(In order to obtain the upper bound on the number of copies to be included, we only need an estimate for OPT. So we can discretize the values in between $\max_{s \in S} \frac{f(\mathbf{e}_s)}{w(s)}$ and $n \max_{s \in S} \frac{f(\mathbf{e}_s)}{w(s)}$ into $\frac{\log n}{\epsilon}$ values and check each one of them.) Other items are called *large*. Let $\ell \leq \frac{1}{\epsilon^6} + \frac{1}{\epsilon^4} = O(\frac{1}{\epsilon^6})$ be the number of large items. Let us decompose $\boldsymbol{x}^* = \boldsymbol{x}^*_L + \boldsymbol{x}^*_{\overline{L}}$ where $L \subseteq S$ is the set of large items.

Our overall approach is a variation of the continuous greedy method. However, we deal with large items and small items separately. This is because, when there are large items, we cannot round a fractional solution to a feasible solution without losing significantly in the approximation ratio. To get around this issue, we guess value sets taken by these large sets.

In Section 4.5.3, we describe the subroutine for handling small items, and then we give the entire algorithm in Section 4.5.4.

4.5.3 Subroutine for handling small items

We give a sketch of the subroutine for handling small items. For notational simplicity, this subroutine is called at each time step, and it updates the current multi-vector $\boldsymbol{X}$ by adding a multi-vector $\boldsymbol{S}$ consisting of vectors in $\mathbb{R}^S_+$ with the following properties.

- $F_\epsilon(\boldsymbol{S} \mid \boldsymbol{X}) \geq \epsilon(1 - O(\epsilon)) F_\epsilon^{\oplus \mathbf{1}}(\boldsymbol{x}^*_{\overline{L}} \mid \boldsymbol{X})$,
- $w(\boldsymbol{S}) \leq \boldsymbol{w}^\top \boldsymbol{x}^*_{\overline{L}}$.

At a high level, by iteratively decreasing a threshold θ by multiplicative factor of $1 - \epsilon$, we add elements if its (estimated) marginal value / weight ratio is higher than θ. We stop the process, when the total marginal value is sufficiently large compared to the guess value g_{small} of $\epsilon F_\epsilon^{\oplus 1}(\boldsymbol{x}^*_{\overline{L}} \mid \boldsymbol{X})$, which is sent to the subroutine. A formal implementation can be found in Algorithm 4.7. The parameters β and δ mean that an additive error of $\beta f(\boldsymbol{x}^*)$ is allowed and the algorithm must succeed with probability at least $1 - \delta$, respectively. We choose parameters $\alpha_{4.7} = O(\epsilon)$, $\beta_{4.7} = O(\beta\epsilon)$, and $\delta_{4.7} = O(\frac{\delta}{n \log_{1+\epsilon}(n/\beta_{4.7})}) = O(\frac{\epsilon\delta}{n \log(n/\beta\epsilon)})$ in Algorithm 4.7, which are determined later.

We use the following binary search algorithm. We omit the algorithm description as it is almost the same as Algorithm 4.4.

Lemma 4.5.4. *Let $\boldsymbol{X}'$ be a multi-vector, $s \in S$, and $\alpha, \beta, \delta \in (0,1)$. By querying $O(\frac{n \log(1/w(s)) \log(1/\delta)}{\alpha\beta})$ times to f, we can find $k \in \mathbb{Z}_+$ such that*

$$\frac{F_\epsilon^{\oplus 1}(k\mathbf{e}_s \mid \boldsymbol{X}')}{w(s)} \geq (1-\alpha)k\theta - \frac{\beta f(\boldsymbol{x}^*)}{nw(s)}.$$
$$\frac{F_\epsilon^{\oplus 1}((k+1)\mathbf{e}_s \mid \boldsymbol{X}')}{w(s)} < (1+\alpha)(k+1)\theta + \frac{\beta f(\boldsymbol{x}^*)}{nw(s)}.$$

with probability at least $1-\delta$.

Algorithm 4.7 Small-Items

Input: A multi-vector $\boldsymbol{X}$, a guess marginal value g_{small}, and $\beta, \delta \in (0,1)$.
Output: A multi-vector $\boldsymbol{S}$.

Let $\boldsymbol{S}$ be the empty multi-vector.
$d \leftarrow \max_{s \in S} \frac{f(\mathbf{e}_s)}{w(s)}$
for $(\theta = d;\ \theta \geq \frac{\beta_{4.7} d}{n};\ \theta \leftarrow (1-\epsilon)\theta)$:
  for each small item s :
    Perform binary search (Lemma 4.5.4) to find maximum k such that $\frac{F_\epsilon^{\oplus 1}(k\mathbf{e}_s \mid \boldsymbol{X} \oplus \boldsymbol{S})}{w(s)} \geq k\theta$ and $kf(\mathbf{e}_s) \leq \beta\epsilon f(\boldsymbol{x}^*)$ with multiplicative error $\alpha_{4.7}$, additive error $\frac{\beta_{4.7} f(\boldsymbol{x}^*)}{n}$, and the failure probability at most $\delta_{4.7}$.
    if such k exists :
      $\boldsymbol{S} \leftarrow \boldsymbol{S} \oplus k\mathbf{e}_s$.
    $\widetilde{F}_\epsilon(\boldsymbol{S} \mid \boldsymbol{X}) \leftarrow$ the guessed value of $F_\epsilon(\boldsymbol{S} \mid \boldsymbol{X})$ with multiplicative error $\alpha_{4.7}$ and additive error $\frac{\beta_{4.7} f(\boldsymbol{x}^*)}{n}$.
    if $\widetilde{F}_\epsilon(\boldsymbol{S} \mid \boldsymbol{X}) \geq (1-10\epsilon) g_{\text{small}}$:
      return $\boldsymbol{S}$.
return $\boldsymbol{S}$.

Lemma 4.5.5. *Suppose $(1+\epsilon) g_{\text{small}} \geq \epsilon F_\epsilon^{\oplus \mathbf{1}}(\boldsymbol{x}^*_L \mid \boldsymbol{X}) \geq g_{\text{small}}$ and $g_{\text{small}} \geq \beta f(\boldsymbol{x}^*)$. With probability at least $1-\delta/2$, Algorithm 4.7 returns $\boldsymbol{S}$ such that $w(\boldsymbol{S}) \leq \boldsymbol{w}^\top \boldsymbol{x}^*_L$.*

Proof. For the notational simplicity, we denote g_{small} by g.

From the choice of δ, all the binary searches and the estimation of the marginal values succeed with probability at least $1-\delta$ by the union bound. In what follows, we assume this happens.

We first observe an upper bound on $F_\epsilon(\boldsymbol{S} \mid \boldsymbol{X})$. Note that, by the stopping condition, we stop when $\widetilde{F}_\epsilon(\boldsymbol{S} \mid \boldsymbol{X}) \geq (1-10\epsilon)g$. Hence, $\widetilde{F}_\epsilon(\boldsymbol{S} \mid \boldsymbol{X}) \leq (1-10\epsilon)g + \beta\epsilon f(\boldsymbol{x}^*) \leq (1-9\epsilon)g$. Now we have that

$$F_\epsilon(\boldsymbol{S} \mid \boldsymbol{X}) \leq (1+\alpha_{4.7})\widetilde{F}_\epsilon(\boldsymbol{S} \mid \boldsymbol{X}) + \frac{\beta_{4.7} f(\boldsymbol{x}^*)}{n} \leq (1+O(\epsilon))(1-9\epsilon)g + O(\epsilon g) \leq (1-8\epsilon)g$$

by choosing the hidden constants in $\alpha_{4.7}$ and $\beta_{4.7}$ small enough.

Now we prove $w(\boldsymbol{S}) \leq \boldsymbol{w}^\top \boldsymbol{x}^*_L$. Assume for contrary that $w(\boldsymbol{S}) > \boldsymbol{w}^\top \boldsymbol{x}^*_L$. Then, we will show that $F_\epsilon(\boldsymbol{X}_S \mid \boldsymbol{X}) \geq (1-3\epsilon)g$, which is a contradiction.

Let $\{k_1\mathbf{e}_{s_1}, \ldots, k_\ell \mathbf{e}_{s_\ell}\}$ be the sequence of vectors in the order they are chosen and $\boldsymbol{X}' = \boldsymbol{X} \oplus \boldsymbol{S}$. Suppose $\boldsymbol{x}^*_L$ is of the form $\sum_{i=1}^m \mathbf{e}_{s_i^*}$, where $s_1^*, \ldots, s_m^*$ are in the order of decreasing $F_\epsilon^{\oplus 1}(\mathbf{e}_{s_i^*} \mid \boldsymbol{X}')/w(s_i^*)$. Note that the same element can appear twice or more. For $0 < \zeta < \boldsymbol{w}^\top \boldsymbol{x}^*_L$, let $N(\zeta) := \min\{j : \sum_{i=1}^j k_i w(s_i) \geq \zeta\}$ and $M(\zeta) = \min\{j : \sum_{i=1}^j w(s_i^*) \geq \zeta\}$. Also, let $\boldsymbol{X}_i$

be the multi-vector obtained by concatenating $k_{s_1}\mathbf{e}_{s_1}, \dots, k_{s_i}\mathbf{e}_{s_i}$ to $\boldsymbol{X}$. Note that $\boldsymbol{X}_0 = \boldsymbol{X}$ and $\boldsymbol{X}_\ell = \boldsymbol{X}'$.

Suppose that $(k_{N(\zeta)}, s_{N(\zeta)})$ is chosen at some step. Let $\widetilde{F}_\epsilon$ denote the estimate done during performing the binary search. Since some element from $\{s_1^*, \dots, s_{M(\zeta)}^*\}$ is not chosen in the previous step, that is, when θ was larger by the factor of $\frac{1}{1-\epsilon}$, we have that

$$\frac{\widetilde{F}_\epsilon^{\oplus 1}(k_{N(\zeta)}\mathbf{e}_{s_{N(\zeta)}} \mid \boldsymbol{X}_{N(\zeta)-1})}{w(s_{N(\zeta)})} \geq k_{N(\zeta)}\theta, \qquad \frac{\widetilde{F}_\epsilon^{\oplus 1}(\mathbf{e}_{s^*_{M(\zeta)}} \mid \boldsymbol{X}_{N(\zeta)-1})}{w(s^*_{M(\zeta)})} < \frac{\theta}{1-\epsilon}.$$

By Lemma 4.5.4, this implies

$$\begin{aligned}\frac{F_\epsilon^{\oplus 1}(k_{N(\zeta)}\mathbf{e}_{s_{N(\zeta)}} \mid \boldsymbol{X}_{N(\zeta)-1})}{w(s_{N(\zeta)})} &\geq (1-\alpha_{4.7})k_{N(\zeta)}\theta - \frac{\beta_{4.7} f(\boldsymbol{x}^*)}{n w(s_{N(\zeta)})}.\\ \frac{F_\epsilon^{\oplus 1}(\mathbf{e}_{s^*_{M(\zeta)}} \mid \boldsymbol{X}_{N(\zeta)-1})}{w(s^*_{M(\zeta)})} &< (1+\alpha_{4.7})\frac{\theta}{1-\epsilon} + \frac{\beta_{4.7} f(\boldsymbol{x}^*)}{n w(s^*_{M(\zeta)})}.\end{aligned}$$

Hence we have for all ζ that

$$\begin{aligned}\frac{F_\epsilon^{\oplus 1}(k_{N(\zeta)}\mathbf{e}_{s_{N(\zeta)}} \mid \boldsymbol{X}_{N(\zeta)-1})}{k_{N(\zeta)} w(s_{N(\zeta)})} &\geq \frac{(1-\epsilon)(1-\alpha_{4.7})}{1+\alpha_{4.7}} \frac{F_\epsilon^{\oplus 1}(\mathbf{e}_{s^*_{M(\zeta)}} \mid \boldsymbol{X}_{N(\zeta)-1}) - \beta_{4.7} f(\boldsymbol{x}^*)/n}{w(s^*_{M(\zeta)})} - \frac{\beta_{4.7} f(\boldsymbol{x}^*)}{n k_{N(\zeta)} w(s_{N(\zeta)})}\\ &\geq \frac{(1-\epsilon)(1-\alpha_{4.7})}{1+\alpha_{4.7}} \frac{F_\epsilon^{\oplus 1}(\mathbf{e}_{s^*_{M(\zeta)}} \mid \boldsymbol{X}') - \beta_{4.7} f(\boldsymbol{x}^*)/n}{w(s^*_{M(\zeta)})} - \frac{\beta_{4.7} f(\boldsymbol{x}^*)}{n k_{N(\zeta)} w(s_{N(\zeta)})}.\end{aligned}$$

Now we can bound the total gain from $\boldsymbol{S}$.

$$\begin{aligned}
&F_\epsilon(\boldsymbol{S} \mid \boldsymbol{X}) = \sum_{i=1}^{\ell} F_\epsilon(k_{s_i}\mathbf{e}_{s_i} \mid \boldsymbol{X}_{i-1})\\
&\geq \epsilon \sum_{i=1}^{\ell} F_\epsilon^{\oplus 1}(k_{s_i}\mathbf{e}_{s_i} \mid \boldsymbol{X}_{i-1}) && \text{(By the concavity of } F\text{)}\\
&\geq \epsilon \int_0^{w(\boldsymbol{S})} \frac{F_\epsilon^{\oplus 1}(k_{N(\zeta)}\mathbf{e}_{s_{N(\zeta)}} \mid \boldsymbol{X}_{N(\zeta)-1})}{k_{N(\zeta)} w(s_{N(\zeta)})} \mathrm{d}\zeta\\
&\geq \epsilon \int_0^{\boldsymbol{w}^\top \boldsymbol{x}^*_L} \left[\frac{(1-\epsilon)(1-\alpha_{4.7})}{1+\alpha_{4.7}} \frac{F_\epsilon^{\oplus 1}(\mathbf{e}_{s^*_{M(\zeta)}} \mid \boldsymbol{x}') - \beta_{4.7} f(\boldsymbol{x}^*)/n}{w(s^*_{M(\zeta)})} - \frac{\beta_{4.7} f(\boldsymbol{x}^*)}{n k_{N(\zeta)} w(s_{N(\zeta)})} \right] \mathrm{d}\zeta\\
&\geq \epsilon \frac{(1-\epsilon)(1-\alpha_{4.7})}{1+\alpha_{4.7}} \sum_{i=1}^{m} F_\epsilon^{\oplus 1}(\mathbf{e}_{s_i^*} \mid \boldsymbol{X}') - 2\beta_{4.7}\epsilon f(\boldsymbol{x}^*)\\
&\geq \epsilon \frac{(1-\epsilon)(1-\alpha_{4.7})}{1+\alpha_{4.7}} F_\epsilon^{\oplus \mathbf{1}}(\boldsymbol{x}^*_L \mid \boldsymbol{X}') - 2\beta_{4.7}\epsilon f(\boldsymbol{x}^*) && \text{(By the DR-submodularity of } f\text{)}\\
&\geq \frac{(1-\epsilon)(1-\alpha_{4.7})}{1+\alpha_{4.7}} \left[g - \epsilon(F_\epsilon(\boldsymbol{X}') - F_\epsilon(\boldsymbol{X})) \right] - 2\frac{\beta_{4.7}\epsilon}{\beta} g.
\end{aligned}$$

By the choice of $\alpha_{4.7}$ and $\beta_{4.7}$, we get $F_\epsilon(\boldsymbol{X}') - F_\epsilon(\boldsymbol{X}) \geq (1-3\epsilon)g$, which is a contradiction. □

Lemma 4.5.6. *Suppose* $(1+\epsilon)g_{\mathrm{small}} \geq \epsilon F_\epsilon^{\oplus \mathbf{1}}(\boldsymbol{x}^*_L \mid \boldsymbol{X}) \geq g_{\mathrm{small}}$ *and* $g_{\mathrm{small}} \geq \beta f(\boldsymbol{x}^*)$. *With probability at least* $1 - \delta/2$, *Algorithm 4.7 returns* $\boldsymbol{S}$ *such that* $F_\epsilon(\boldsymbol{S} \mid \boldsymbol{X}) \geq (1 - O(\epsilon))g_{\mathrm{small}}$.

Proof. Similarly to the proof of Lemma 4.5.5, we assume that all the binary searches and the estimation of the marginal values have succeeded.

Recall that $w(s) \leq \epsilon^4$ for all small item s. Hence, if we run Algorithm 4.7 ignoring the stopping condition (Line 9), either $w(\boldsymbol{X}_S) \geq 1 - \epsilon^4$ holds or

$$\frac{F_\epsilon^{\oplus 1}(\mathbf{e}_s \mid \boldsymbol{X} \oplus \boldsymbol{S})}{w(s)} < \frac{(1+\alpha_{4.7})\beta_{4.7} d}{n} + \frac{\beta_{4.7} f(\boldsymbol{x}^*)}{n}$$

holds for all $s \in S$.

Suppose that the former case happens. In this case, we can use a similar argument to the proof of Lemma 4.5.5. There, we have assumed that $w(\boldsymbol{S}) > \boldsymbol{w}^\top \boldsymbol{x}^*_{\overline{L}}$. Here, we have $w(\boldsymbol{S}) \geq 1 - \epsilon^4 \geq \boldsymbol{w}^\top \boldsymbol{x}^*_{\overline{L}} - \epsilon^4$. Regarding that $F_\epsilon^{\oplus 1}(\mathbf{e}_{s^*_{M(\zeta)}} \mid \boldsymbol{X}')$ is a decreasing function of ζ, we have that $F_\epsilon(\boldsymbol{S} \mid \boldsymbol{X}) \geq (1-3\epsilon) g_{\mathrm{small}}$.

Suppose that the latter case happens. In this case, the total loss caused by ignoring subsequent items is at most

$$\sum_{s \in S} \frac{(1+\alpha_{4.7})\beta_{4.7} d + \beta_{4.7} f(\boldsymbol{x}^*)}{n} \leq O(\beta\epsilon d + \beta\epsilon f(\boldsymbol{x}^*)) = O(\epsilon g_{\mathrm{small}}).$$

Hence, we have $F_\epsilon(\boldsymbol{X}_S \mid \boldsymbol{X}) \geq (1-3\epsilon) g_{\mathrm{small}} - O(\epsilon g_{\mathrm{small}}) = (1 - O(\epsilon)) g_{\mathrm{small}}$. □

Lemma 4.5.7. *The running time of Algorithm 4.7 is* $O(\frac{n^2}{\beta\epsilon^3} \log \frac{n}{\beta\epsilon} \log \frac{n \log(n/\beta\epsilon)}{\epsilon\delta} \log \frac{1}{\min_s w(s)})$.

Proof. The running time is dominated by the time for performing binary searches. The number of binary searches is $n \log_{1+\epsilon} \frac{n}{\beta_{4.7}} = O(\frac{n}{\epsilon} \log \frac{n}{\beta\epsilon})$. Each binary search takes

$$\frac{n}{\alpha_{4.7}\beta_{4.7}} \log \frac{n}{\delta_{4.7}} \log \frac{1}{\min_s w(s)} = O\left(\frac{n^2}{\beta\epsilon^2} \log \frac{n \log(n/\beta\epsilon)}{\epsilon\delta} \log \frac{1}{\min_s w(s)} \right)$$

time. Hence, the lemma holds. □

4.5.4 Complete algorithm

Now we describe the complete algorithm for knapsack constraints (Algorithm 4.8). First, it guesses a set of values and sends it to Guessing-Continuous-Greedy (Algorithm 4.9). Algorithm 4.9 at each time step finds step sizes and large items $(k_1, s_1), (k_2, s_2), \ldots, (k_\ell, s_\ell)$ and a multi-vector $\boldsymbol{S}$ consisting of small items, and then update the current solution $\boldsymbol{X}$ by concatenating these items. Finally, Algorithm 4.9 rounds the obtained multi-vector $\boldsymbol{X}$ using Rounding (Algorithm 4.10), which will be explained in Section 4.5.4.

To guarantee the approximation ratio, the items chosen by Algorithm 4.9 at each time step should satisfy the following properties.

- $F_\epsilon^{\oplus 1}(k_i \mathbf{e}_{s_i} \mid \boldsymbol{X}) \geq (1 - o(1)) F_\epsilon^{\oplus 1}(\boldsymbol{x}^* \mid \boldsymbol{X}) - o(f(\boldsymbol{x}^*))$.
- $k_i w(s_i) \leq k^*_i w(s^*_i)$ (Recall that $\boldsymbol{x}^*_L = \sum_{i=1}^{\ell} k^*_i \mathbf{e}_{s^*_i}$).
- $F_\epsilon(\boldsymbol{S} \mid \boldsymbol{X}) \geq \epsilon(1 - o(1)) F_\epsilon^{\oplus 1}(\boldsymbol{x}^*_{\overline{L}} \mid \boldsymbol{X}) - o(f(\boldsymbol{x}^*))$.
- $w(\boldsymbol{X}_S) \leq \boldsymbol{w}^\top \boldsymbol{x}^*_{\overline{L}}$

We can achieve the latter two properties by using the algorithm described in Section 4.5.3.

We want to take a set of small items and k large items from different domains so that we can apply Lemma 4.5.3 later. To this end, we let $S_0, S_1, \ldots, S_\ell$ be copies of the item set S, and consider a function $f' : \mathbb{Z}_+^{S_0 \cup \cdots \cup S_\ell} \to \mathbb{R}_+$ defined as $f'(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_\ell) = f(\boldsymbol{x}_0 + \cdots + \boldsymbol{x}_\ell)$. We show that various properties of f are preserved in f'.

Algorithm 4.8 Knapsack Constraint

Input: $f : \mathbb{Z}_+^S \to \mathbb{R}_+$, weight vector $\boldsymbol{w} \in \mathbb{R}_+^S$.
Output: A vector $\boldsymbol{x} \in \mathbb{Z}_+^S$ satisfying $\boldsymbol{w}^\top \boldsymbol{x} \leq 1$.

$d \leftarrow \max_{s \in S} \frac{f(\mathbf{e}_s)}{w(s)}$.
GuessSet $\leftarrow \{d, (1-\epsilon)d, (1-\epsilon)^2 d, \ldots, \beta_{4.8} d\} \cup \{0\}$.
TimeSet $\leftarrow \{1, 2, \ldots, 1/\epsilon\}$.
$\mathcal{X} = \emptyset$.
for all $G = \{g_1^{(t)}, g_2^{(t)}, \ldots, g_\ell^{(t)}, g_{\mathrm{small}}^{(t)} \in \text{GuessSet} \mid t \in \text{Time-Set}\}$:
  $\boldsymbol{x} \leftarrow$ Guessing-Continuous-Greedy$(f, \boldsymbol{w}, G)$
  $\mathcal{X} \leftarrow \mathcal{X} \cup \{\boldsymbol{x}\}$.
return $\max_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$.

Lemma 4.5.8. *Suppose $f : \mathbb{Z}_+^S \to \mathbb{R}_+$ is a monotone DR-submodular function. Let f' be the function defined as above. Then, f' is a monotone DR-submodular function.*

Proof. We prove the case that the domain of f' is $\mathbb{Z}_+^{S_1 \cup S_2}$. The general case is achieved by induction on the number of copies of S. The monotonicity of f' is clear.

For the DR-submodularity, let $\boldsymbol{x}_i, \boldsymbol{y}_i \in \mathbb{Z}_+^{S_i}$ be vectors such that $\boldsymbol{x}_i \leq \boldsymbol{y}_i$ for $i = 1, 2$. Let us take $s \in S_1$. Then, we have

$$\begin{aligned} &f'(\boldsymbol{x}_1 + \mathbf{e}_s, \boldsymbol{x}_2) - f'(\boldsymbol{x}_1, \boldsymbol{x}_2) = f(\boldsymbol{x}_1 + \mathbf{e}_s + \boldsymbol{x}_2) - f(\boldsymbol{x}_1 + \boldsymbol{x}_2) \\ &\geq f(\boldsymbol{y}_1 + \mathbf{e}_s + \boldsymbol{y}_2) - f(\boldsymbol{y}_1 + \boldsymbol{y}_2) \qquad \text{(By the DR-submodularity of } f\text{)} \\ &= f'(\boldsymbol{y}_1 + \mathbf{e}_s, \boldsymbol{y}_2) - f'(\boldsymbol{y}_1, \boldsymbol{y}_2) \end{aligned}$$

The case of $s \in S_2$ can be handled similarly. □

Lemma 4.5.9. *For a some set G of guess values, Algorithm 4.9 computes a multi-vector $\boldsymbol{X}$ such that $F_\epsilon(\boldsymbol{X}) \geq (1 - 1/\mathrm{e} - O(\epsilon)) f(\boldsymbol{x}^*)$ with probability at least $\frac{5}{6}$.*

Proof. We will determine G later. Suppose that, for this particular G, every binary search in Algorithm 4.9 and the call of Small-Items have succeeded. By setting $\delta_{4.9} = \frac{\epsilon}{6\ell n} = O(\frac{\epsilon^7}{n})$, this happens with probability at least $\frac{5}{6}$ by the union bound. Let $\widetilde{F}_\epsilon(\cdot)$ denote the estimation of $F_\epsilon(\cdot)$ calculated when performing the binary searches.

Let $\boldsymbol{X}^t$ be the multi-vector at the end of time step t. We want to show the following.

Claim 4.5.10. *For some choice of G, we have*

$$F_\epsilon(\boldsymbol{X}^{t+1}) - F_\epsilon(\boldsymbol{X}^t) \geq \epsilon(1 - O(\epsilon))(f(\boldsymbol{x}^*) - F_\epsilon(\boldsymbol{X}^{t+1})) - O(\epsilon^2 f(\boldsymbol{x}^*)).$$

for every t.

Proof. Fix t, and we write $\boldsymbol{X}' = \boldsymbol{X}^{t+1}$ and $\boldsymbol{X} = \boldsymbol{X}^t$. For $i \in [\ell]$, define $\boldsymbol{X}_i = \boldsymbol{X} \oplus \bigoplus_{j=1}^{i} k_j \mathbf{e}_{s_j}$. For each step, suppose that we have chosen g_i as the largest value in the Guess-Set smaller than $\widetilde{F}_\epsilon^{\oplus 1}(k_i^* \mathbf{e}_{s_i^*} \mid \boldsymbol{X}_{i-1})$. Then, we have the following two properties.

- $\widetilde{F}_\epsilon^{\oplus 1}(k_i \mathbf{e}_{s_i} \mid \boldsymbol{X}_{i-1}) \geq g_i \geq (1-\epsilon)\widetilde{F}_\epsilon^{\oplus 1}(k_i \mathbf{e}_{s_i} \mid \boldsymbol{X}_{i-1}) - \beta_{4.8} d$. This implies

$$F_\epsilon^{\oplus 1}(k_i \mathbf{e}_{s_i} \mid \boldsymbol{X}_{i-1}) \geq \frac{(1-\epsilon)(1-\alpha_{4.9})}{(1+\alpha_{4.9})} F_\epsilon^{\oplus 1}(k_i^* \mathbf{e}_{s_i^*} \mid \boldsymbol{X}_{i-1})) - \beta_{4.8} d - 2\beta_{4.9} f(\boldsymbol{x}^*).$$

Algorithm 4.9 Guessing-Continuous-Greedy

Input: $f : \mathbb{Z}_+^S \to \mathbb{R}_+$, weight vector $\boldsymbol{w} \in \mathbb{R}_+^S$, a set of guess values G
Output: An integral vector $\boldsymbol{z} \in \mathbb{Z}_+^S$

$S_0, S_1, \ldots, S_\ell \leftarrow S$ (Duplicate the items)
Define $f' : \mathbb{Z}_+^{S_0 \cup S_1 \cup \cdots \cup S_\ell} \to \mathbb{R}_+$ as $f'(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell) = f(\boldsymbol{x}_0 + \sum_{i=1}^{\ell} \boldsymbol{x}_i)$.
Let F_ϵ be the continuous extension (4.4) of f'.
$\boldsymbol{X} \leftarrow$ an empty multi-vector.
for $(t \leftarrow 1; t \leq \frac{1}{\epsilon}; t \leftarrow t+1)$:
  for $i = 1, \ldots, \ell$:
    For each $s \in S_i$, perform binary search (Lemma 4.5.4) to find minimum k_s such that $F_\epsilon^{\oplus 1}(k_s \mathbf{e}_s \mid \boldsymbol{X})/w(s) \geq g_i^{(t)}$ with multiplicative error $\alpha_{4.9}$, additive error $\beta_{4.9} f(\boldsymbol{x}^*)/w(s)$ and failure probability $\delta_{4.9}$.
    $s \leftarrow \operatorname{argmin}\{k_s w(s) : s \in S_i\}$.
    $\boldsymbol{X} \leftarrow \boldsymbol{X} \oplus k_s \mathbf{e}_s$.
  $\boldsymbol{S} \leftarrow$ Small-Items$(f', \boldsymbol{X}, g_0^{(t)})$ with additive error $\beta_{4.9} f(\boldsymbol{x}^*)$ and failure probability $\delta_{4.9}$.
  $\boldsymbol{X} \leftarrow \boldsymbol{X} \oplus \boldsymbol{S}$.
$\boldsymbol{z} \leftarrow$ Rounding$(\boldsymbol{X}, \epsilon \mathbf{1})$.
if $\boldsymbol{w}^\top \boldsymbol{z} > 1$:
  $\boldsymbol{z} \leftarrow \mathbf{0}$.
return $\boldsymbol{z}$.

- $k_{s_i} w(s_i) \leq k_i^* w(s_i^*)$. This is because $k_i^* \mathbf{e}_{s_i^*}$ should have been a candidate item when choosing $k_i \mathbf{e}_{s_i}$ as $\widetilde{F}_\epsilon^{\oplus 1}(k_{s_i} \mathbf{e}_{s_i} \mid \boldsymbol{X}_{i-1}) \geq g_i$.

Suppose that we have chosen g_0 so that $(1+\epsilon) g_0 \geq \epsilon F_\epsilon^{\oplus \mathbf{1}}(\boldsymbol{x}_L^* \mid \boldsymbol{X}) \geq g_0$. By Lemma 4.5.6, it holds that

- $F_\epsilon(\boldsymbol{S} \mid \boldsymbol{X}_\ell) \geq \epsilon(1 - O(\epsilon)) F_\epsilon^{\oplus 1}(\boldsymbol{x}_L^* \mid \boldsymbol{X}_\ell) - \beta_{4.9} f(\boldsymbol{x}^*)$.
- $w(\boldsymbol{S}) \leq \boldsymbol{w}^\top \boldsymbol{x}_L^*$

For this choice of $\boldsymbol{g}$, we can bound the increase at each time step as follows.

$$
\begin{aligned}
&F_\epsilon(\boldsymbol{X}') - F_\epsilon(\boldsymbol{X}) \\
&= \sum_{i=1}^{\ell} F_\epsilon(k_i \mathbf{e}_{s_i} \mid \boldsymbol{X}_i) + F_\epsilon(\boldsymbol{X}') - F_\epsilon(\boldsymbol{X}_\ell) \\
&\geq \epsilon \sum_{i=1}^{\ell} F_\epsilon^{\oplus 1}(k_i \mathbf{e}_{s_i} \mid \boldsymbol{X}_{i-1}) + F_\epsilon(\boldsymbol{X}') - F_\epsilon(\boldsymbol{X}_\ell) && \text{(By Lemma 4.5.1)} \\
&\geq \epsilon \frac{(1-\epsilon)(1-\alpha_{4.9})}{(1+\alpha_{4.9})} \sum_{i=1}^{\ell} \Big(F_\epsilon^{\oplus 1}(k_i^* s_i^* \mid \boldsymbol{X}_{i-1}) - \beta_{4.8} d - 2\beta_{4.9} f(\boldsymbol{x}^*) \Big) + \epsilon(1 - O(\epsilon)) F_\epsilon^{\oplus \mathbf{1}}(\boldsymbol{x}_L^* \mid \boldsymbol{X}_\ell) - \beta_{4.9} f(\boldsymbol{x}^*) \\
&\geq \epsilon \frac{(1-\epsilon)(1-\alpha_{4.9})}{(1+\alpha_{4.9})} \sum_{i=1}^{\ell} \Big(F_\epsilon^{\oplus 1}(k_i^* s_i^* \mid \boldsymbol{X}' \oplus \boldsymbol{x}_L^*) \Big) + \epsilon(1 - O(\epsilon)) F_\epsilon^{\oplus \mathbf{1}}(\boldsymbol{x}_L^* \mid \boldsymbol{X}') - \beta_{4.8} \ell d - \beta_{4.9}(2\ell+1) f(\boldsymbol{x}^*) \\
&\qquad && \text{(By Lemma 4.5.2)} \\
&\geq \epsilon \frac{(1-\epsilon)(1-\alpha_{4.9})}{(1+\alpha_{4.9})} F_\epsilon^{\oplus 1}(\boldsymbol{x}_L^* \mid \boldsymbol{X}' \oplus \boldsymbol{x}_L^*) + \epsilon(1 - O(\epsilon)) F_\epsilon^{\oplus \mathbf{1}}(\boldsymbol{x}_L^* \mid \boldsymbol{X}') - \beta_{4.8} \ell d - \beta_{4.9}(2\ell+1) f(\boldsymbol{x}^*) \\
&\geq \epsilon(1 - O(\epsilon) - 2\alpha_{4.9}) F_\epsilon^{\oplus 1}(\boldsymbol{x}^* \mid \boldsymbol{X}') - \beta_{4.8} \ell d - \beta_{4.9}(2\ell+1) f(\boldsymbol{x}^*) \\
&\geq \epsilon(1 - O(\epsilon) - 2\alpha_{4.9})(f(\boldsymbol{x}^*) - F_\epsilon(\boldsymbol{X}')) - \beta_{4.8} \ell d - \beta_{4.9}(2\ell+1) f(\boldsymbol{x}^*).
\end{aligned}
$$

By choosing $\alpha_{4.9} = O(\epsilon)$ and $\beta_{4.8} = \beta_{4.9} = O(\frac{\epsilon^2}{\ell})$, this quantity is at least $\epsilon(1 - O(\epsilon))(f(\boldsymbol{x}^*) - F_\epsilon(\boldsymbol{X}')) - O(\epsilon^2 f(\boldsymbol{x}^*))$. □

Now we proceed similarly to the analysis of continuous greedy for the polymatroid constraint. Rephrasing the inequality in the claim above we get

$$(1-\epsilon')f(\boldsymbol{x}^*) - F_\epsilon(\boldsymbol{X}^{t+1}) \leq \frac{(1-\epsilon')f(\boldsymbol{x}^*) - F_\epsilon(\boldsymbol{X}^t)}{1+\epsilon(1-O(\epsilon))}.$$

for a certain $\epsilon' = O(\epsilon)$. Then, by induction we can prove

$$(1-\epsilon')f(\boldsymbol{x}^*) - F_\epsilon(\boldsymbol{X}^t) \leq \left(\frac{1}{(1+\epsilon(1-O(\epsilon))^t)}\right)(1-\epsilon')f(\boldsymbol{x}^*).$$

Substituting $t = 1/\epsilon$ and rephrasing once again,

$$F_\epsilon(\boldsymbol{X}^{1/\epsilon}) \geq (1 - \frac{1}{\mathrm{e}} - O(\epsilon))(1-\epsilon')f(\boldsymbol{x}^*) = (1 - \frac{1}{\mathrm{e}} - O(\epsilon))f(\boldsymbol{x}^*).$$ □

Lemma 4.5.11. *The running time of Algorithm 4.8 is* $O(\frac{n^2}{\epsilon^{18}} \log^3 \frac{n}{\epsilon} \log \frac{1}{\min_s w(s)})(\frac{1}{\epsilon})^{O(1/\epsilon^8)}$.

Proof. The number of choices for t and G is $\frac{1}{\epsilon} \cdot (\log_{1+\epsilon}(\beta_{4.8}))^\ell = (\frac{1}{\epsilon})^{O(1/\epsilon^8)}$. The running time of Algorithm 4.7 is

$$O\left(\frac{\ell}{\epsilon}\frac{n^2}{\beta_{4.9}\epsilon^3}\log\frac{n}{\beta_{4.9}\epsilon}\log\frac{n\log(n/\beta_{4.9}\epsilon)}{\epsilon\delta_{4.9}}\log\frac{1}{\min_s w(s)}\right) = O\left(\frac{n^2}{\epsilon^{18}}\log^3\frac{n}{\epsilon}\log\frac{1}{\min_s w(s)}\right)$$

. The total number of queries to f caused by the binary search is

$$O\left(\frac{n\ell}{\epsilon}\frac{1}{\alpha_{4.9}\beta_{4.9}}\log\frac{1}{\delta_{4.9}}\log\frac{1}{\min_s w(s)}\right) = O\left(\frac{n}{\epsilon^{16}}\log\frac{n}{\epsilon}\log\frac{1}{\min_s w(s)}\right)$$

Hence, the lemma holds. □

Rounding

In this subsection we show how to convert a multi-vector produced in Algorithm 4.9 to an integral vector.

At time t, suppose that $(k_i^{(t)}, s_i^t)$ is chosen for $i \in [\ell]$. Let $\boldsymbol{X}_L^t = \bigoplus k_i \mathbf{e}_{s_i}$ and $\boldsymbol{X}_L = \bigoplus_t \boldsymbol{X}_L^t$. Let $\boldsymbol{S}^t$ be the multi-vector obtained by Guessing-Continuous-Greedy at time t and $\boldsymbol{X}_{\overline{L}} = \bigoplus_t \boldsymbol{S}^t$. We restrict our attention to the multi-vector $\boldsymbol{X}$ corresponding to the set of guess values G for which we have the following properties.

- $F_\epsilon(\boldsymbol{X}) \geq (1 - 1/\mathrm{e} - O(\epsilon))f(\boldsymbol{x}^*)$.
- For each time t and each $i \in [\ell]$, we have $k_i^{(t)} w(s_i^t) \leq k_i^* w(s_i^*)$.
- For each time t we have that $w(\boldsymbol{S}^t) \leq \boldsymbol{w}^\top \boldsymbol{x}^*_{\overline{L}}$.

Our rounding method is described in Algorithm 4.10. Basically, we apply Lemma 4.5.3 to $\boldsymbol{X}_L$ and the independent rounding to $\boldsymbol{S}$.

Lemma 4.5.12. *Algorithm 4.10 rounds a multi-vector* $\boldsymbol{X}$ *to a vector* $\boldsymbol{z} \in \mathbb{Z}_+^S$ *such that* $\mathbf{E}[f(\boldsymbol{z})] \geq (1 - O(\epsilon))F_\epsilon(\boldsymbol{X})$.

Algorithm 4.10 Rounding

Input: A multi-vector $\boldsymbol{X} = \boldsymbol{X}_{\overline{L}} \oplus \boldsymbol{X}_L$
Output: A vector $\boldsymbol{z} \in \mathbb{Z}_+^S$ satisfying $\boldsymbol{w}^\top \boldsymbol{z} \leq 1$ and $\mathbf{E}[f(\boldsymbol{z})] \geq (1 - O(\epsilon))F_\epsilon(\boldsymbol{X})$

$\boldsymbol{z}_L \leftarrow U_{\epsilon(1-\epsilon)}(\boldsymbol{X}_L; \Pi)$
$\boldsymbol{z}_{\overline{L}} \leftarrow \mathbf{0}$.
for $k\mathbf{e}_s \in \boldsymbol{X}_{\overline{L}}$:
  if $kw(s) \leq \epsilon^3 w(S \setminus L)$:
    Add $k\mathbf{e}_s$ to $\boldsymbol{z}_{\overline{L}}$ with probability $\epsilon(1-\epsilon)$.
$\boldsymbol{z} = \boldsymbol{z}_L + \boldsymbol{z}_{\overline{L}}$.
return $\boldsymbol{z}$.

Proof. Let $\boldsymbol{X}'_{\overline{L}}$ consist of all $k\mathbf{e}_s \in \boldsymbol{S}$ such that $kw(s) \leq \epsilon^3 w(\boldsymbol{X}_{\overline{L}})$. Let $\boldsymbol{X}''_{\overline{L}}$ be the rest, that is, $\boldsymbol{X}'_{\overline{L}} \oplus \boldsymbol{X}''_{\overline{L}} = \boldsymbol{X}_{\overline{L}}$. First let us relate the value of $\boldsymbol{X} = \boldsymbol{X}_{\overline{L}} \oplus \boldsymbol{X}_L$ to that of $\boldsymbol{X}' = \boldsymbol{X}'_{\overline{L}} \oplus \boldsymbol{X}_L$.

$$\begin{aligned} F_\epsilon(\boldsymbol{X}') = F_\epsilon(\boldsymbol{X}'_{\overline{L}} \oplus \boldsymbol{X}_L) &= \mathbf{E}[f(R_\epsilon(\boldsymbol{X}'_{\overline{L}} \oplus \boldsymbol{X}_L)] \geq \mathbf{E}[f(R_\epsilon(\boldsymbol{X}_{\overline{L}} \oplus \boldsymbol{X}_L))] - \mathbf{E}[f(R_\epsilon(\boldsymbol{X}''_{\overline{L}}))] \\ &\geq F_\epsilon(\boldsymbol{X}_{\overline{L}} \oplus \boldsymbol{X}_L) - \epsilon \sum_{k\mathbf{e}_s \in \boldsymbol{X}''_S} f(k\mathbf{e}_s) \geq F_\epsilon(\boldsymbol{X}) - \epsilon \max_{k\mathbf{e}_s \in \boldsymbol{X}''_S} f(k\mathbf{e}_s) \sum_{k\mathbf{e}_s \in \boldsymbol{X}''_S} 1 \\ &\geq F_\epsilon(\boldsymbol{X}) - \epsilon^7 f(\boldsymbol{x}^*)\frac{1}{\epsilon^3} \geq F_\epsilon(\boldsymbol{X}) - \epsilon^4 f(\boldsymbol{x}^*) \geq (1-\epsilon)F_\epsilon(\boldsymbol{X}). \end{aligned}$$

Next we note that we get $\boldsymbol{z}$ by selecting exactly one random item from each S_i, and sampling independently from $S_{\overline{L}}$. Hence, we have

$$\begin{aligned} \mathbf{E}[f(\boldsymbol{z})] &\geq \mathbf{E}[f(U_{\epsilon(1-\epsilon)}(\boldsymbol{X}_L; \Pi) \mid R_{\epsilon(1-\epsilon)}(\boldsymbol{X}'_{\overline{L}}))] + \mathbf{E}[f(R_{\epsilon(1-\epsilon)}(\boldsymbol{X}'_{\overline{L}}))] \\ &\geq \mathbf{E}[f(R_{\epsilon(1-\epsilon)}(\boldsymbol{X}_L) \mid R_{\epsilon(1-\epsilon)}(\boldsymbol{X}'_{\overline{L}}))] + \mathbf{E}[f(R_{\epsilon(1-\epsilon)}(\boldsymbol{X}'_{\overline{L}}))] \qquad \text{(By Lemma 4.5.3)} \\ &\geq F_{\epsilon(1-\epsilon)}(\boldsymbol{X}') \geq (1-\epsilon)F_\epsilon(\boldsymbol{X}') \geq (1 - O(\epsilon))F_\epsilon(\boldsymbol{X}). \qquad \square \end{aligned}$$

Lemma 4.5.13. *Algorithm 4.10 rounds a multi-vector $\boldsymbol{X}$ to a vector $\boldsymbol{z} \in \mathbb{Z}_+^S$ such that $\boldsymbol{w}^\top \boldsymbol{z} \leq 1$ with probability at least $\frac{5}{6}$.*

Proof. Let $\boldsymbol{u}$ be a vector sampled from $U_{\epsilon(1-\epsilon)}(\boldsymbol{X}_L; \Pi)$. Then, $\boldsymbol{u}$ contains at most one element from the set $\{k_i^{(t)} \mathbf{e}_{s_i^t}\}_{t \in [\frac{1}{\epsilon}]}$ for each $i \in [\ell]$. Hence, we have $\boldsymbol{w}^\top \boldsymbol{u} \leq \boldsymbol{w}^\top \boldsymbol{x}^*_L$.

Let $\boldsymbol{r}'_{\epsilon(1-\epsilon)}$ be sampled from $R_{\epsilon(1-\epsilon)}(\boldsymbol{X}'_{\overline{L}})$. Then it is enough to prove that with high probability $\boldsymbol{w}^\top \boldsymbol{r}'_{\epsilon(1-\epsilon)} \leq \boldsymbol{w}^\top \boldsymbol{x}^*_{\overline{L}}$. Let $\boldsymbol{r}_{\epsilon(1-\epsilon)}$ and $\boldsymbol{r}_\epsilon$ be sampled from $R_{\epsilon(1-\epsilon)}(\boldsymbol{X}'_{\overline{L}})$ and $R_\epsilon(\boldsymbol{X}'_{\overline{L}})$, respectively. Then we have

$$\mathbf{E}[\boldsymbol{w}^\top \boldsymbol{r}'_{\epsilon(1-\epsilon)}] = (1-\epsilon)\,\mathbf{E}[\boldsymbol{w}^\top \boldsymbol{r}_{\epsilon(1-\epsilon)}] \leq (1-\epsilon)\,\mathbf{E}[\boldsymbol{w}^\top \boldsymbol{r}_\epsilon] \leq (1-\epsilon)w(S \setminus L) \leq (1-\epsilon)\boldsymbol{w}^\top \boldsymbol{x}^*_{\overline{L}}$$

Additionally we have that for each $k\mathbf{e}_s \in \boldsymbol{X}'_{\overline{L}}$, we have that $kw(s) \leq \epsilon^3 w(S \setminus L) \leq \epsilon^3 \boldsymbol{w}^\top \boldsymbol{x}^*_{\overline{L}}$. Hence, we can bound the probability that $\boldsymbol{w}^\top \boldsymbol{r}'_{\epsilon(1-\epsilon)} > \boldsymbol{w}^\top \boldsymbol{x}^*_{\overline{L}}$ by a simple Chernoff bound to show this happens with very low probability. $\square$

Combining Lemmas 4.5.9 and 4.5.13, we obtain the following.

Theorem 4.5.14. *Algorithm 4.8 outputs a $(1 - 1/\mathrm{e} - O(\epsilon))$-approximate feasible solution with probability at least $\frac{2}{3}$.*

Algorithm 4.11

Input: a feasible solution $\boldsymbol{x}_0$

Let $\boldsymbol{x} := \boldsymbol{x}_0$.
Let $Q := \{(s,k) : s \in S \text{ and } 1 \leq k \leq c(s) - x(s)\}$.
while $\boldsymbol{w}^\top \boldsymbol{x} < 1$ and $Q \neq \emptyset$:
  Find s and k maximizing $\frac{f(k\mathbf{e}_s|\boldsymbol{x})}{kw(s)}$ among $(s,k) \in Q$.
  if $w \cdot b + w(s)k \leq 1$:
    Let $x(s) := x(s) + k$.
    Remove all pairs (s,l) such that $x(s) + l > c(s)$ from Q.
  else
    Remove (s,k) from Q.
return $\boldsymbol{x}$

4.6 Knapsack Constrant for Lattice Submodular Function

We now describe our approximation algorithm for maximizing a monotone lattice submodular function subject to a knapsack constraint. In our algorithm, we first enumerate every feasible solution $\boldsymbol{x}_0$ such that $|\mathrm{supp}^+(\boldsymbol{x}_0)| \leq 3$. The number of such solutions is $O(\|\boldsymbol{c}\|_\infty^3 |S|^3)$. For each feasible solution $\boldsymbol{x}_0$, the algorithm increases each component of $\boldsymbol{x}_0$ in Algorithm 4.11, and obtains a feasible solution $\boldsymbol{x}$. Finally, the algorithm returns the best one among the obtained solutions. Since Algorithm 4.11 requires $O(\|\boldsymbol{c}\|_\infty^2 |S|)$ time, the total complexity is $O(\|\boldsymbol{c}\|_\infty^5 |S|^4)$.

Remarks Let us leave some remarks on our algorithm. Although the worst case complexity of our algorithm is quite expensive, Algorithm 4.11 can be accelerated using a "lazy evaluation" technique given in [121], and thus it runs more efficiently in practice. Also, there is another approach to reduce the time complexity at the expense of a good approximation ratio. We can apply ideas similar to those for maximizing a submodular set function to make practical simpler algorithms with somewhat worse approximation factors [111]. For example, as in [5], if we enumerate solutions with only one positive support before performing Algorithm 4.11, then we can find a $\frac{1-1/\mathrm{e}}{2}$-approximate solution in $O(\|\boldsymbol{c}\|_\infty^2 |S|^2)$ time.

Theorem 4.6.1. *For monotone lattice submodular function maximization subject to a knapsack constraint, we can find a* $(1 - 1/\mathrm{e})$*-approximate solution in* $O(\|\boldsymbol{c}\|_\infty^5 |S|^4)$ *time.*

Let $\boldsymbol{x}^*$ be an optimal solution for a given instance of the problem. We first examine properties of Algorithm 4.11 that hold for *arbitrary* $\boldsymbol{x}_0$. We then show that there exists *some* $\boldsymbol{x}_0$ in the enumeration step such that Algorithm 4.11 returns $\boldsymbol{x}$ with $f(\boldsymbol{x}) \geq (1 - 1/\mathrm{e}) f(\boldsymbol{x}^*)$, which proves Theorem 4.6.1.

Let us fix an initial solution $\boldsymbol{x}_0$, and analyze behavior of Algorithm 4.11 with input $\boldsymbol{x}_0$. We denote by $\boldsymbol{x}_i$ the tentative solution $\boldsymbol{x}$ at the beginning of the ith iteration and denote by s_i and k_i s and k chosen in the ith iteration, respectively. Assume that Algorithm 4.11 first has not updated the tentative solution $\boldsymbol{x}$ in the Lth trial. Equivalently, let L be the minimum number such that $\boldsymbol{x}_L = \boldsymbol{x}_{L+1}$ and $\boldsymbol{x}_i < \boldsymbol{x}_{i+1}$ for $i = 1, \ldots, L-1$. Note that if such a situation never happens during the execution of Algorithm 4.11, define L to be the number of iterations.

Lemma 4.6.2. *Without loss of generality, we may assume that* $x(s_L) + k_L \leq x^*(s_L)$.

Proof. Suppose that $x(s_L) + k_L > x^*(s_L)$. Let us consider a modified instance in which the capacity of s_L is decreased to $x(s_L) + k_L - 1$. The optimal value is unchanged by this modification because x^* is still feasible and optimal. Furthermore, Algorithm 4.11 returns the same solution

(with respect to same $\boldsymbol{x}_0$). Thus it suffices to analyze the algorithm in the modified instance. Repeating this argument completes the proof of this lemma. □

Consider the ith iteration of the algorithm. For simplicity, we denote

$$\Delta_i = f(k_i \mathbf{e}_{s_i} \mid \boldsymbol{x}), \quad \delta_i = \frac{\Delta_i}{k_i w(s_i)}, \quad w_i = w(s_i).$$

Note that $f(\boldsymbol{x}_i) = f(\boldsymbol{x}_0) + \sum_{j=1}^{i-1} \Delta_i$ for $i = 1, \ldots, L$. Let $B' := 1 - \boldsymbol{w}^\top \boldsymbol{x}_0$.

Lemma 4.6.3. *For $i = 1, \ldots, L$, we have $\Delta_i \geq \dfrac{w_i k_i}{B'} \left(f(\boldsymbol{x}^*) - f(\boldsymbol{x}_i) \right)$.*

Proof. Let us denote $\boldsymbol{x}_i \vee \boldsymbol{x}^* = \boldsymbol{x}_i + \sum \alpha_s \mathbf{e}_s$, where the sum is taken over s in $\mathrm{supp}^+(\boldsymbol{x}^* - \boldsymbol{x}_i)$ and $\alpha_s := x^*(s) - x_i(s)$. Since $\boldsymbol{x}_i \vee \boldsymbol{x}^*(s)\mathbf{e}_s = \boldsymbol{x}_i + \alpha_s \mathbf{e}_s$ for each $s \in \mathrm{supp}^+(\boldsymbol{x}^* - \boldsymbol{x}_i)$, (3.10) implies

$$\begin{aligned} f(\boldsymbol{x}_i \vee \boldsymbol{x}^*) &\leq f(\boldsymbol{x}_i) + \sum_{s \in \mathrm{supp}^+(\boldsymbol{x}^* - \boldsymbol{x}_i)} \left(f(\boldsymbol{x}_i + \alpha_s \mathbf{e}_s) - f(\boldsymbol{x}_i) \right) \\ &\leq f(\boldsymbol{x}_i) + \sum_{s \in \mathrm{supp}^+(\boldsymbol{x}^* - \boldsymbol{x}_i)} w(s) \alpha_s \delta_i, \end{aligned}$$

where the last inequality follows from the fact that $(f(\boldsymbol{x}_i + k\mathbf{e}_s) - f(\boldsymbol{x}_i))/(w(s)k) \leq \delta_i$ for all $s \in S$ and k. Since $\sum w(s)\alpha_s$ is at most B' and $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}_i \vee \boldsymbol{x}^*)$ by the monotonicity of f, it holds that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}_i) + \delta_i B'$. Therefore, we obtain $\delta_i \geq (f(\boldsymbol{x}^*) - f(\boldsymbol{x}_i))/B'$. □

Applying Lemma 4.6.3 repeatedly, we have the following lemmas.

Lemma 4.6.4. *For $i = 1, \ldots, L$, we have*

$$f(\boldsymbol{x}^*) - f(\boldsymbol{x}_i) \leq \left(f(\boldsymbol{x}^*) - f(\boldsymbol{x}_0) \right) \cdot \prod_{j=1}^{i} \left(1 - w_j k_j / B' \right).$$

Proof. We prove this lemma by induction on i. For $i = 1$, then the inequality holds because $\Delta_1 \geq w_1 k_1 (f(\boldsymbol{x}^*) - f(\boldsymbol{x}_0))/B'$ by Lemma 4.6.3. For $i > 1$, we have

$$\begin{aligned} &f(\boldsymbol{x}^*) - f(\boldsymbol{x}_0) - \sum_{j=1}^{i} \Delta_j = f(\boldsymbol{x}^*) - f(\boldsymbol{x}_0) - \sum_{j=1}^{i-1} \Delta_j - \Delta_i \\ &\leq f(\boldsymbol{x}^*) - f(\boldsymbol{x}_0) - \sum_{j=1}^{i-1} \Delta_j - \frac{w_i k_i}{B'} \left(f(\boldsymbol{x}^*) - f(\boldsymbol{x}_0) - \sum_{j=1}^{i-1} \Delta_j \right) && \text{(by Lemma 4.6.3)} \\ &= \left(1 - \frac{w_i k_i}{B'} \right) \left(f(\boldsymbol{x}^*) - f(\boldsymbol{x}_0) - \sum_{j=1}^{i-1} \Delta_j \right) \\ &\leq \left(1 - \frac{w_i k_i}{B'} \right) \cdot \left(f(\boldsymbol{x}^*) - f(\boldsymbol{x}_0) \right) \cdot \prod_{j=1}^{i-1} \left(1 - \frac{w_j k_j}{B'} \right), && \text{(by the induction hypothesis)} \end{aligned}$$

which completes the proof. □

Lemma 4.6.5. *It holds that $f(\boldsymbol{x}^*) - f(\boldsymbol{x}_i) \leq (f(\boldsymbol{x}^*) - f(\boldsymbol{x}_0))/\mathrm{e}$.*

Proof. We have

$$\prod_{i=1}^{L}\left(1-\frac{w_i k_i}{B'}\right) \leq \exp\left(-\sum_{i=1}^{L}\frac{w_i k_i}{B'}\right) \leq \frac{1}{\mathrm{e}}, \tag{4.5}$$

where the last inequality follows the fact that $\sum_{i=1}^{L} w_i k_i \geq B'$. Combining this fact and Lemma 4.6.4 completes the proof. □

We now move to proving that the greedy procedure returns a $(1-1/\mathrm{e})$-approximate solution for some $\boldsymbol{x}_0$ in the enumeration step. If the optimal solution $\boldsymbol{x}^*$ satisfies $|\mathrm{supp}^+(\boldsymbol{x}^*)| \leq 3$, it can be found in the enumeration step. Therefore, we assume that all the optimal solutions have more than three positive components. Let s_1^*, s_2^*, s_3^* be elements in S satisfying:

$$s_i^* \in \operatorname*{argmax}_{s \in S \setminus \{s_1^*,\ldots,s_{i-1}^*\}} f\left(x^*(s)\mathbf{e}_s \mid \vee_{j=1}^{i-1} x^*(s_j^*)\mathbf{e}_{s_j^*}\right)$$

for $i = 1, 2, 3$. Since the size of the support of $\boldsymbol{x}^*$ is more than three, such s_1^*, s_2^* and s_3^* clearly exist.

Lemma 4.6.6. *For the feasible solution $\boldsymbol{x}_0$ with the support $\{s_1^*, s_2^*, s_3^*\}$ satisfying $x_0(s_i^*) = x^*(s_i^*)$ for $1 \leq i \leq 3$, we have $\Delta_L \leq f(\boldsymbol{x}_0)/3$.*

Proof. It follows that $\Delta_L = f(k_L \mathbf{e}_{s_L}) - f(\boldsymbol{x}_L) \leq f(\boldsymbol{x}_L \vee x^*(s_L)\mathbf{e}_{s_L}) - f(\boldsymbol{x}_L)$ since $x_L(s_L) + k_L \leq x^*(s_L)$. By Lemma 3.5.1, Δ_L is at most $f(x^*(s_L)\mathbf{e}_{s_L}) - f(\mathbf{0}) = f(x^*(s_L)\mathbf{e}_{s_L})$, and hence $\Delta_L \leq f(x^*(s_1^*)\mathbf{e}_{s_1^*})$ by the choice of s_1^*. Similarly, by the choices of s_2^* and s_3^*, we have $\Delta_L \leq f(x^*(s_2^*)\mathbf{e}_{s_2^*} \mid x^*(s_1^*)\mathbf{e}_{s_1^*})$ and $\Delta_L \leq f(x^*(s_3^*)\mathbf{e}_{s_3^*} \mid x^*(s_1^*)\mathbf{e}_{s_1^*} + x^*(s_2^*)\mathbf{e}_{s_2^*})$. Adding these inequalities, we obtain $\Delta_L \leq f(\boldsymbol{x}_0)/3$. □

We are now ready to prove Theorem 4.6.1. Since $f(\boldsymbol{x}) \geq f(\boldsymbol{x}_0) + \sum_{i=1}^{L} \Delta_i - \Delta_L$, Lemmas 4.6.5 and 4.6.6 imply that

$$\begin{aligned} f(\boldsymbol{x}) &\geq (1-1/3)f(\boldsymbol{x}_0) + (1-1/\mathrm{e})(f(\boldsymbol{x}^*) - f(\boldsymbol{x}_0)) \\ &\geq (1-1/\mathrm{e})f(\boldsymbol{x}^*) \end{aligned}$$

for the initial solution $\boldsymbol{x}_0$ described in Lemma 4.6.6. This completes the proof of Theorem 4.6.1.

Chapter 5

The Diminishing Return Submodular Cover Problem

In this chapter, we consider the following generalization of the submodular cover problem for set functions: Given a monotone DR-submodular function $f : \mathbb{Z}_+^S \to \mathbb{R}_+$, a subadditive function $c : \mathbb{Z}_+^S \to \mathbb{R}_+$, $\alpha > 0$, and $r \in \mathbb{Z}_+$, we are to

$$\text{minimize } c(\boldsymbol{x}) \qquad \text{subject to } f(\boldsymbol{x}) \geq \alpha, \quad \mathbf{0} \leq \boldsymbol{x} \leq r\mathbf{1}, \tag{5.1}$$

where we say that c is *subadditive* if $c(\boldsymbol{x} + \boldsymbol{y}) \leq c(\boldsymbol{x}) + c(\boldsymbol{y})$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_+^S$. We call problem (5.1) the *DR-submodular cover problem*.

For the problem (5.1), we devise a *bicriteria* approximation algorithm based on the decreasing threshold technique of [13]. More precisely, our algorithm takes the additional parameters $0 < \epsilon, \delta < 1$. The output $\boldsymbol{x} \in \mathbb{Z}_+^S$ of our algorithm is guaranteed to satisfy that $c(\boldsymbol{x})$ is at most $(1+3\epsilon)\rho\left(1 + \log \frac{d}{\beta}\right)$ times the optimum and $f(\boldsymbol{x}) \geq (1-\delta)\alpha$, where ρ is the curvature of c (see Section 5.1 for the definition), $d = \max_s f(\mathbf{e}_s)$ is the maximum value of f over all standard unit vectors, and β is the minimum value of the positive increments of f in the feasible region.

Running Time (dependency on r): An important feature of our algorithm is that the running time depends on the bit length of r only *polynomially* whereas the naive reduction algorithms depend on it *exponentially* as mentioned above. More precisely, the running time of our algorithm is $O(\frac{n}{\epsilon} \log \frac{nrc_{\max}}{\delta c_{\min}} \log r)$, which is polynomial in the input size, whereas the naive algorithm is only psuedo-polynomial time algorithm. In fact, our experiments using real and synthetic datasets show that our algorithm is considerably faster than naive algorithms. Furthermore, in terms of the objective value (that is, the cost of the output), our algorithm also exhibits comparable performance.

Approximation Guarantee: Our approximation guarantee on the cost is almost tight. Note that the DR submodular cover problem (5.1) includes the set cover problem, in which we are given a collection of sets, and we want to find a minimum number of sets that covers all the elements. In our context, S corresponds to the collection of sets, the cost c is the number of chosen sets, and f is the number of covered elements. It is known that we cannot obtain an $o(\log m)$-approximation unless NP $\subseteq$ DTIME($n^{O(\log \log n)}$), where m is the number of elements [61]. However, since for the set cover problem we have $\rho = 1$, $d = O(m)$, and $\beta = 1$, our approximation guarantee is $O(\log m)$.

Related Work Our result can be compared with several results in the literature for the submodular cover problem for set functions. It is shown by Wolsey [173] that if $c(X) = |X|$, a simple greedy algorithm yields $(1 + \log \frac{d}{\beta})$-approximation, which coincides with our approximation ratio except for the $(1 + 3\epsilon)$ factor. Note that $\rho = 1$ when $c(X) = |X|$, or more generally, when c is modular. Recently, Wan et al. [167] discussed a slightly different setting, in which c is also submodular and both f and c are integer valued. They proved that the greedy algorithm achieves $\rho H(d)$-approximation, where $H(d) = 1 + 1/2 + \cdots + 1/d$ is the dth harmonic number. Again, their ratio asymptotically coincides with our approximation ratio (Note that $\beta \geq 1$ when f is integer valued).

We note that the submodular cover problem and the submodular maximization problem are somewhat dual to each other. Indeed, Iyer and Bilmes [93] showed that a bicriteria algorithm of one of these problems yields a bicriteria algorithm for the other. We note that our result is not implied by the results of previous chapters via the duality of [93].

5.1 Algorithm for the DR-submodular Cover

Recall the DR-submodular cover problem (5.1). Let $f : \mathbb{Z}_+^S \to \mathbb{R}_+$ be a monotone DR-submodular function and let $c : \mathbb{Z}_+^S \to \mathbb{R}_+$ be a subadditive cost function. The objective is to minimize $c(\boldsymbol{x})$ subject to $f(\boldsymbol{x}) \geq \alpha$ and $\mathbf{0} \leq \boldsymbol{x} \leq r\mathbf{1}$, where $\alpha > 0$ and $r \in \mathbb{Z}_+$ are the given constants. Without loss of generality, we can assume that $\max\{f(\boldsymbol{x}) : \mathbf{0} \leq \boldsymbol{x} \leq r\mathbf{1}\} = \alpha$ (otherwise, we can consider $\widehat{f}(\boldsymbol{x}) := \min\{f(\boldsymbol{x}), \alpha\}$ instead of f). Furthermore, we can assume $c(\boldsymbol{x}) > 0$ for any $\boldsymbol{x} \in \mathbb{Z}_+^S$.

A pseudocode description of our algorithm is presented in Algorithm 5.1. The algorithm can be viewed as a modified version of the greedy algorithm and works as follows: We start with the initial solution $\boldsymbol{x} = \mathbf{0}$ and increase each coordinate of $\boldsymbol{x}$ gradually. To determine the amount of increments, the algorithm maintains a threshold θ that is initialized to be sufficiently large enough. For each $s \in S$, the algorithm finds the largest integer step size $0 < k \leq r - x(s)$ such that the marginal cost-gain ratio $\frac{f(k\mathbf{e}_s|\boldsymbol{x})}{kc(\mathbf{e}_s)}$ is above the threshold θ. If such k exists, the algorithm updates $\boldsymbol{x}$ to $\boldsymbol{x} + k\mathbf{e}_s$. After repeating this for each $s \in S$, the algorithm decreases the threshold θ by a factor of $(1 - \epsilon)$. If $\boldsymbol{x}$ becomes feasible, the algorithm returns the current $\boldsymbol{x}$. Even if $\boldsymbol{x}$ does not become feasible, the final $\boldsymbol{x}$ satisfies $f(\boldsymbol{x}) \geq (1 - \delta)\alpha$ if we iterate until θ gets sufficiently small.

Algorithm 5.1 Decreasing Threshold for the DR-Submodular Cover Problem

Input: $f : \mathbb{Z}_+^S \to \mathbb{R}_+$, $c : \mathbb{Z}_+^S \to \mathbb{R}_+$, $r \in \mathbb{N}$, $\alpha > 0$, $\epsilon > 0$, $\delta > 0$.
Output: $\mathbf{0} \leq \boldsymbol{x} \leq r\mathbf{1}$ such that $f(\boldsymbol{x}) \geq \alpha$.

$\boldsymbol{x} \leftarrow \mathbf{0}$, $d \leftarrow \max_{s \in S} f(\mathbf{e}_s)$, $c_{\min} \leftarrow \min_{s \in S} c(\mathbf{e}_s)$, $c_{\max} \leftarrow \max_{s \in S} c(\mathbf{e}_s)$
for $(\theta = \frac{d}{c_{\min}}; \theta \geq \frac{\delta}{nc_{\max}r}d; \theta \leftarrow \theta(1 - \epsilon))$:
  for all $s \in S$:
    Find maximum integer $0 < k \leq r - x(s)$ such that $\frac{f(k\mathbf{e}_s|\boldsymbol{x})}{kc(\mathbf{e}_s)} \geq \theta$ with binary search.
    if such k exists :
      $\boldsymbol{x} \leftarrow \boldsymbol{x} + k\mathbf{e}_s$
    if $f(\boldsymbol{x}) \geq \alpha$:
      Break the outer **for** loop.
return $\boldsymbol{x}$

Before we claim the theorem, we need to define several parameters on f and c. Let $\beta := \min\{f(\mathbf{e}_s \mid \boldsymbol{x}) : s \in S, \boldsymbol{x} \in \mathbb{Z}_+^S, f(\mathbf{e}_s \mid \boldsymbol{x}) > 0\}$ and $d := \max_s f(\mathbf{e}_s)$. Let $c_{\max} := \max_s c(\mathbf{e}_s)$ and

$c_{\min} := \min_s c(\mathbf{e}_s)$. Define the *curvature* of c to be

$$\rho := \min_{\boldsymbol{x}^*:\text{optimal solution}} \frac{\sum_{s\in\{\boldsymbol{x}^*\}} c(\mathbf{e}_s)}{c(\boldsymbol{x}^*)}. \tag{5.2}$$

Definition 5.1.1. For $\gamma \geq 1$ and $0 < \delta < 1$, a vector $\boldsymbol{x} \in \mathbb{Z}_+^S$ is a (γ, δ)-*bicriteria approximate* solution if $c(\boldsymbol{x}) \leq \gamma \cdot c(\boldsymbol{x}^*)$, $f(\boldsymbol{x}) \geq (1-\delta)\alpha$, and $\mathbf{0} \leq \boldsymbol{x} \leq r\mathbf{1}$.

Our main theorem is described below.

Theorem 5.1.2. *Algorithm 5.1 outputs a* $\left((1+3\epsilon)\rho\left(1+\log\frac{d}{\beta}\right), \delta\right)$*-bicriteria approximate solution in* $O\left(\frac{n}{\epsilon}\log\frac{nrc_{\max}}{\delta c_{\min}}\log r\right)$ *time.*

First, we introduce a notation. Let us assume that $\boldsymbol{x}$ is updated L times in the algorithm. Let $\boldsymbol{x}_i$ be the variable $\boldsymbol{x}$ after the ith update ($i = 0, \ldots, L$). Note that $\boldsymbol{x}_0 = \mathbf{0}$ and $\boldsymbol{x}_L$ is the final output of the algorithm. Let $s_i \in S$ and $k_i \in \mathbb{Z}_+$ be the pair used in the ith update for $i = 1, \ldots, L$; that is, $\boldsymbol{x}_i = \boldsymbol{x}_{i-1} + k_i\mathbf{e}_{s_i}$ for $i = 1, \ldots, L$. Let $\mu_0 := 0$ and $\mu_i := \frac{k_i c(\mathbf{e}_{s_i})}{f(k_i\mathbf{e}_{s_i} | \boldsymbol{x}_{i-1})}$ for $i = 1, \ldots, L$. Let $\hat{\mu}_0 := 0$ and $\hat{\mu}_i := \theta_i^{-1}$ for $i = 1, \ldots, L$, where θ_i is the threshold value on the ith update. Note that $\hat{\mu}_{i-1} \leq \hat{\mu}_i$ for $i = 1, \ldots, L$. Let $\boldsymbol{x}^*$ be an optimal solution such that $\rho \cdot c(\boldsymbol{x}^*) = \sum_{s\in\{\boldsymbol{x}^*\}} c(\mathbf{e}_s)$.

We regard that in the ith update, the elements of $\{\boldsymbol{x}^*\}$ are charged by the value of $\mu_i(f(\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) - f(\mathbf{e}_s \mid \boldsymbol{x}_i))$. Then, the total charge on $\{\boldsymbol{x}^*\}$ is defined as

$$T(\boldsymbol{x}, f) := \sum_{s\in\{\boldsymbol{x}^*\}} \sum_{i=1}^{L} \mu_i(f(\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) - f(\mathbf{e}_s \mid \boldsymbol{x}_i)).$$

Claim 5.1.3. *Let us fix $1 \leq i \leq L$ arbitrary and let θ be the threshold value on the ith update. Then, we have*

$$\frac{f(k_i\mathbf{e}_{s_i} \mid \boldsymbol{x}_{i-1})}{k_i c(\mathbf{e}_{s_i})} \geq \theta \quad \textit{and} \quad \frac{f(\mathbf{e}_s \mid \boldsymbol{x}_{i-1})}{c(\mathbf{e}_s)} \leq \frac{\theta}{1-\epsilon} \quad (s \in S).$$

Proof. The first inequality is immediate from the fact that (k_i, s_i) is chosen by the algorithm. For the second inequality, if $\theta = d/c_{\min}$ then it is trivial. Thus, we can assume $\theta < d/c_{\min}$; that is, there were at least one threshold update. Suppose to the contrary that $\frac{f(\mathbf{e}_s|\boldsymbol{x}_{i-1})}{c(\mathbf{e}_s)} > \frac{\theta}{1-\epsilon}$ for some $s \in S$. Let k be the chosen step size for the previous update on the sth component and let $\boldsymbol{x}'$ be the variable $\boldsymbol{x}$ at that time. Then, $f((k+1)\mathbf{e}_s \mid \boldsymbol{x}') \geq f(k\mathbf{e}_s \mid \boldsymbol{x}') + f(\mathbf{e}_s \mid \boldsymbol{x}) > \frac{\theta}{1-\epsilon}kc(\mathbf{e}_s) + \frac{\theta}{1-\epsilon}c(\mathbf{e}_s) = \frac{\theta}{1-\epsilon}(k+1)c(\mathbf{e}_s)$, which contradicts the choice of k. □

Eliminating θ from the inequalities in Claim 5.1.3, we obtain

$$\frac{k_i c(\mathbf{e}_{s_i})}{f(k_i\mathbf{e}_{s_i} \mid \boldsymbol{x}_{i-1})} \leq \frac{1}{1-\epsilon}\frac{c(\mathbf{e}_s)}{f(\mathbf{e}_s \mid \boldsymbol{x}_{i-1})} \quad (i = 1, \ldots, L, \quad s \in S) \tag{5.3}$$

Furthermore, we have $\mu_i \leq \hat{\mu}_i \leq \frac{1}{1-\epsilon}\mu_i$ for $i = 1, \ldots, L$.

Claim 5.1.4. $c(\boldsymbol{x}) \leq \frac{1}{1-\epsilon}T(\boldsymbol{x}, f)$.

Proof.

$$
\begin{aligned}
c(\boldsymbol{x}) &\le \sum_{i=1}^{L} k_i c(\mathbf{e}_{s_i}) && \text{(since } c \text{ is subadditive)}\\
&= \sum_{i=1}^{L} \mu_i f(k_i \mathbf{e}_{s_i} \mid \boldsymbol{x}_{i-1}) = \sum_{i=1}^{L} \mu_i (f(\boldsymbol{x}_i) - f(\boldsymbol{x}_{i-1}))\\
&\le \sum_{i=1}^{L} \hat{\mu}_i (f(\boldsymbol{x}_i) - f(\boldsymbol{x}_{i-1}))\\
&= \sum_{i=1}^{L} \hat{\mu}_i (f(\boldsymbol{x}^*) - f(\boldsymbol{x}_{i-1})) - \sum_{i=1}^{L} \hat{\mu}_i (f(\boldsymbol{x}^*) - f(\boldsymbol{x}_i))\\
&= \sum_{i=1}^{L} \hat{\mu}_i (f(\boldsymbol{x}^*) - f(\boldsymbol{x}_{i-1})) - \sum_{i=1}^{L} \hat{\mu}_{i-1} (f(\boldsymbol{x}^*) - f(\boldsymbol{x}_{i-1}))\\
& && \text{(since } \hat{\mu}_0 = 0 \text{ and } f(\boldsymbol{x}^*) = f(\boldsymbol{x}_L))\\
&= \sum_{i=1}^{L} (\hat{\mu}_i - \hat{\mu}_{i-1})(f(\boldsymbol{x}^*) - f(\boldsymbol{x}_{i-1})) \le \sum_{i=1}^{L} (\hat{\mu}_i - \hat{\mu}_{i-1}) \sum_{s \in \{\boldsymbol{x}^*\}} f(\mathbf{e}_s \mid \boldsymbol{x}_{i-1})\\
&= \sum_{i=1}^{L} (\hat{\mu}_i - \hat{\mu}_{i-1}) \sum_{s \in \{\boldsymbol{x}^*\}} f(\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) = \sum_{s \in \{\boldsymbol{x}^*\}} \sum_{i=1}^{L} (\hat{\mu}_i - \hat{\mu}_{i-1}) f(\mathbf{e}_s \mid \boldsymbol{x}_{i-1})\\
&= \sum_{s \in \{\boldsymbol{x}^*\}} \Big(\sum_{i=1}^{L} \hat{\mu}_i f(\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) - \sum_{i=1}^{L} \hat{\mu}_{i-1} f(\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) \Big)\\
&= \sum_{s \in \{\boldsymbol{x}^*\}} \sum_{i=1}^{L} \hat{\mu}_i \Big(f(\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) - f(\mathbf{e}_s \mid \boldsymbol{x}_i) \Big) && \text{(since } \hat{\mu}_0 = 0 \text{ and } f(\boldsymbol{x}^*) = f(\boldsymbol{x}_L))\\
&= (1-\epsilon)^{-1} T(\boldsymbol{x}, f) && \text{(since } \hat{\mu}_i \le (1-\epsilon)^{-1} \mu_i)
\end{aligned}
$$

□

Claim 5.1.5. *For each $s \in \{\boldsymbol{x}^*\}$, the total charge on s is at most $\frac{1}{1-\epsilon}(1 + \log(d/\beta))c(\mathbf{e}_s)$.*

Proof. Let us fix $s \in \{\boldsymbol{x}^*\}$ and let l be the minimum i such that $f(\mathbf{e}_s \mid \boldsymbol{x}_i) = 0$. By (5.3), we have

$$
\mu_i = \frac{k_i c(\mathbf{e}_{s_i})}{f(k_i \mathbf{e}_{s_i} \mid \boldsymbol{x}_{i-1})} \le \frac{1}{1-\epsilon} \cdot \frac{c(\mathbf{e}_s)}{f(\mathbf{e}_s \mid \boldsymbol{x}_{i-1})}. \qquad (i = 1, \dots, l)
$$

Then, we have

$$
\begin{aligned}
&\sum_{i=1}^{L} \mu_i (f(\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) - f(\mathbf{e}_s \mid \boldsymbol{x}_i)) = \sum_{i=1}^{l-1} \mu_i (f(\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) - f(\mathbf{e}_s \mid \boldsymbol{x}_i)) + \mu_l f(\mathbf{e}_s \mid \boldsymbol{x}_{l-1})\\
&\le \frac{1}{1-\epsilon} c(\mathbf{e}_s) \Big(\sum_{i=1}^{l-1} \frac{(f(\mathbf{e}_s \mid \boldsymbol{x}_{i-1}) - f(\mathbf{e}_s \mid \boldsymbol{x}_i))}{f(\mathbf{e}_s \mid \boldsymbol{x}_{i-1})} + \frac{f(\mathbf{e}_s \mid \boldsymbol{x}_{l-1})}{f(\mathbf{e}_s \mid \boldsymbol{x}_{l-1})} \Big)\\
&\le \frac{1}{1-\epsilon} c(\mathbf{e}_s) \Big(1 + \sum_{i=1}^{l-1} \Big(1 - \frac{f(\mathbf{e}_s \mid \boldsymbol{x}_i)}{f(\mathbf{e}_s \mid \boldsymbol{x}_{i-1})} \Big) \Big)\\
&\le \frac{1}{1-\epsilon} c(\mathbf{e}_s) \Big(1 + \sum_{i=1}^{l-1} \log \frac{f(\mathbf{e}_s \mid \boldsymbol{x}_{i-1})}{f(\mathbf{e}_s \mid \boldsymbol{x}_i)} \Big) && \text{(since } 1 - 1/x \le \log x \text{ for } x \ge 1)
\end{aligned}
$$

$$= \frac{1}{1-\epsilon} c(\mathbf{e}_s)\Big(1 + \log \frac{f(\mathbf{e}_s \mid \boldsymbol{x}_0)}{f(\mathbf{e}_s \mid \boldsymbol{x}_{l-1})}\Big) \leq \frac{1}{1-\epsilon}\Big(1 + \log \frac{d}{\beta}\Big) c(\mathbf{e}_s) \qquad \square$$

Proof of Theorem 5.1.2. Combining these claims, we have

$$c(\boldsymbol{x}) \leq \frac{1}{1-\epsilon} \cdot T(\boldsymbol{x}, f) \leq \frac{1}{(1-\epsilon)^2} \cdot \left(1 + \log \frac{d}{\beta}\right) \cdot \sum_{s \in \{\boldsymbol{x}^*\}} c(\mathbf{e}_s) \leq (1+3\epsilon) \cdot \left(1 + \log \frac{d}{\beta}\right) \cdot \rho c(\boldsymbol{x}^*).$$

Thus, $\boldsymbol{x}$ is an approximate solution with the desired ratio.

Let us see that $\boldsymbol{x}$ approximately satisfies the constraint; that is, $f(\boldsymbol{x}) \geq (1-\delta)\alpha$. We will now consider a slightly modified version of the algorithm; in the modified algorithm, the threshold is updated until $f(\boldsymbol{x}) = \alpha$. Let $\boldsymbol{x}'$ be the output of the modified algorithm. Then, we have

$$f(\boldsymbol{x}') - f(\boldsymbol{x}) \leq \sum_{s \in \{\boldsymbol{x}'\}} f(\mathbf{e}_s \mid \boldsymbol{x}) \leq \sum_{s \in \{\boldsymbol{x}'\}} \frac{\delta c(\mathbf{e}_s)}{c_{\max} nr} d \leq \delta d \leq \delta \alpha$$

The second inequality holds since $c(\mathbf{e}_s) \leq c_{\max}$ and $|\{\boldsymbol{x}'\}| \leq nr$. Therefore, $f(\boldsymbol{x}) \geq f(\boldsymbol{x}') - \delta\alpha = (1-\delta)\alpha$. $\square$

5.2 Discussion

Integer-valued Case. Let us make a simple remark on the case that f is integer valued. Without loss of generality, we can assume $\alpha \in \mathbb{Z}_+$. Then, Algorithm 5.1 always returns a feasible solution for any $0 < \delta < 1/\alpha$. Therefore, our algorithm can be easily modified to an approximation algorithm if f is integer valued.

Definition of Curvature. Several authors [93], [160] use a different notion of curvature called the *total curvature*, whose natural extension for a function over the integer lattice is as follows: The total curvature κ of $c : \mathbb{Z}_+^S \to \mathbb{R}_+$ is defined as $\kappa := 1 - \min_{s \in S} \frac{c(\mathbf{e}_s \mid r\mathbf{1} - \mathbf{e}_s)}{c(\mathbf{e}_s)}$. Note that $\kappa = 0$ if c is modular, while $\rho = 1$ if c is modular. For example, Iyer and Bilmes [93] devised a bicriteria approximation algorithm whose approximation guarantee is roughly $O((1-\kappa)^{-1} \log \frac{\beta}{d})$.

Let us investigate the relation between ρ and κ for DR-submodular functions. One can show that $1 - \kappa \leq \rho \leq (1-\kappa)^{-1}$, which means that our bound in terms of ρ is tighter than one in terms of $(1-\kappa)^{-1}$.

Lemma 5.2.1. *Let $c : \mathbb{Z}_+ \to \mathbb{R}_+$ be a DR-submodular function, and let ρ and κ be the curvature and the total curvature of c, respectively. Then, we have $1 - \kappa \leq \rho \leq (1-\kappa)^{-1}$.*

Fact 5.2.2. For $a_1, b_1, a_2, b_2, \ldots, a_n, b_n > 0$, $\frac{a_1 + a_2 + \cdots + a_n}{b_1 + b_2 + \cdots + b_n} \geq \min\left\{\frac{a_1}{b_1}, \frac{a_2}{b_2}, \ldots, \frac{a_n}{b_n}\right\}$.

Proof. Let $\boldsymbol{x}^*$ be the optimal solution of the submodular cover problem that attains the minimum of (5.2)

$$\begin{aligned}
\rho &= \frac{\sum_{s \in \{\boldsymbol{x}^*\}} c(\mathbf{e}_s)}{c(\boldsymbol{x}^*)} \\
&\geq \frac{\sum_{s \in \{\boldsymbol{x}^*\}} c(\mathbf{e}_s \mid r\mathbf{1} - \mathbf{e}_s)}{\sum_{s \in \{\boldsymbol{x}^*\}} c(\mathbf{e}_s)} && \text{(since } c(\mathbf{e}_s) \geq c(\mathbf{e}_s \mid r\mathbf{1} - \mathbf{e}_s) \text{ and } c(\boldsymbol{x}^*) \leq \textstyle\sum_{s \in \{\boldsymbol{x}^*\}} c(\mathbf{e}_s)\text{)} \\
&\geq \min_{s \in \{\boldsymbol{x}^*\}} \frac{c(\mathbf{e}_s \mid r\mathbf{1} - \mathbf{e}_s)}{c(\mathbf{e}_s)} && \text{(by Fact 5.2.2)} \\
&\geq \min_{s \in S} \frac{c(\mathbf{e}_s \mid r\mathbf{1} - \mathbf{e}_s)}{c(\mathbf{e}_s)} = 1 - \kappa.
\end{aligned}$$

On the other hand, let $\boldsymbol{x}^* = \mathbf{e}_{s_1} + \cdots + \mathbf{e}_{s_k}$, where $s_1, \ldots, s_k$ are the elements of the support of $\boldsymbol{x}^*$, $\boldsymbol{x}_i := \mathbf{e}_{s_1} + \cdots + \mathbf{e}_{s_i}$ $(i = 1, \ldots, k)$, and $\boldsymbol{x}_0 := \mathbf{0}$. Similar argument shows that

$$\begin{aligned}
\frac{1}{\rho} &= \frac{c(\boldsymbol{x}^*)}{\sum_{s \in \{\boldsymbol{x}^*\}} c(\mathbf{e}_s)} \\
&= \frac{\sum_{i=1}^{k} c(\mathbf{e}_{s_i} \mid \boldsymbol{x}_{i-1})}{\sum_{s \in \{\boldsymbol{x}^*\}} c(\mathbf{e}_s)} \\
&\geq \frac{\sum_{i=1}^{k} c(\mathbf{e}_{s_i} \mid r\mathbf{1} - \mathbf{e}_{s_i})}{\sum_{s \in \{\boldsymbol{x}^*\}} c(\mathbf{e}_s)} && \text{(since } \boldsymbol{x}_{i-1} \leq r\mathbf{1} - \mathbf{e}_{s_i}\text{)} \\
&= \frac{\sum_{s \in \{\boldsymbol{x}^*\}} c(\mathbf{e}_s \mid r\mathbf{1} - \mathbf{e}_s)}{\sum_{s \in \{\boldsymbol{x}^*\}} c(\mathbf{e}_s)} \geq 1 - \kappa.
\end{aligned}$$

Thus $\rho^{-1} \geq 1 - \kappa$, which is equivalent to $\rho \leq (1 - \kappa)^{-1}$. □

Lazy Evaluation. We finally note that we can combine the lazy evaluation technique [121], which significantly reduces runtime in practice, with our algorithm. Specifically, we first push all the elements in S to a max-based priority queue. Here, the key of an element $s \in S$ is $\frac{f(\mathbf{e}_s)}{c(\mathbf{e}_s)}$. Then the inner loop of Algorithm 5.1 is modified as follows: Instead of checking all the elements in S, we pop elements whose keys are at least θ. For each popped element $s \in S$, we find k such that $0 < k \leq r - x(s)$ with $\frac{f(k\mathbf{e}_s|\boldsymbol{x})}{kc(\mathbf{e}_s)} \geq \theta$ with binary search. If there is such k, we update $\boldsymbol{x}$ with $\boldsymbol{x} + k\mathbf{e}_s$. Finally, we push s again with the key $\frac{f(\mathbf{e}_s|\boldsymbol{x})}{c(\mathbf{e}_s)}$ if $x(s) < r$.

The correctness of this technique is obvious because of the DR-submodularity of f. In particular, the key of each element $s \in S$ in the queue is always at least $\frac{f(\mathbf{e}_s|\boldsymbol{x})}{c(\mathbf{e}_s)}$, where $\boldsymbol{x}$ is the current vector. Hence, we never miss $s \in S$ with $\frac{f(k\mathbf{e}_s|\boldsymbol{x})}{kc(\mathbf{e}_s)} \geq \theta$.

5.3 Experiments

In the rest of this chapter, we show several experimental results of our method.

5.3.1 Experimental Setting

We conducted experiments on a Linux server with an Intel Xeon E5-2690 (2.90 GHz) processor and 256 GB of main memory. The experiments required, at most, 4 GB of memory. All the algorithms were implemented in C++ and compiled with g++ 4.6.3.

In our experiments, the cost function $\boldsymbol{c} : \mathbb{Z}_+^S \to \mathbb{R}_+$ is always chosen as $\boldsymbol{c}(\boldsymbol{x}) = \|\boldsymbol{x}\|_1 := \sum_{s \in S} x(s)$. Let $f : \mathbb{Z}_+^S \to \mathbb{R}_+$ be a submodular function and α be the worst quality guarantee. We implemented the following four methods:

- Decreasing-threshold is our method with the lazy evaluation technique. We chose $\delta = 0.01$ as stated otherwise.
- Greedy is a method in which, starting from $\boldsymbol{x} = \mathbf{0}$, we iteratively increment $x(s)$ for $s \in S$ that maximizes $f(\boldsymbol{x} + \mathbf{e}_s) - f(\boldsymbol{x})$ until we get $f(\boldsymbol{x}) \geq \alpha$. We also implemented the lazy evaluation technique [108].
- Degree is a method in which we assign $x(s)$ a value proportional to the marginal $f(\mathbf{e}_s) - f(\mathbf{0})$, where $\|\boldsymbol{x}\|_1$ is determined by binary search so that $f(\boldsymbol{x}) \geq \alpha$. Precisely speaking, $x(s)$ is approximately proportional to the marginal since $x(s)$ must be an integer.

- Uniform is a method that returns $k\mathbf{1}$ for minimum $k \in \mathbb{Z}_+$ such that $f(k\mathbf{1}) \geq \alpha$.

We use the following real-world and synthetic datasets to confirm the accuracy and efficiency of our method against other methods. We set $r = 100,000$ for both problems.

Sensor placement. Our first experiment is the generalized sensor placement scenario described in Example 3.1.5 and Section 3.4.2 with a real-world dataset. We used a dataset acquired by running simulations on a 129-vertex sensor network used in Battle of the Water Sensor Networks (BWSN) [134]. We used the "bwsn-utilities" [163] program to simulate 3000 random injection events to this network for a duration of 96 hours. Let S and E be the set of the 129 sensors in the network and the set of the 3000 events, respectively. For each sensor $s \in S$ and event $e \in E$, a value $z(s, e)$ is provided, which denotes the time, in minutes, the pollution has reached s after the injection time.[1]

We define a function $f : \mathbb{Z}_+^S \to \mathbb{R}_+$ as follows: Let $\boldsymbol{x} \in \mathbb{Z}_+^S$ be a vector, where we regard $x(s)$ as the energy level of the sensor s. Suppose that when the pollution reaches a sensor s, the probability that we can detect it is $1 - (1-p)^{x(s)}$, where $p = 0.0001$. In other words, by spending unit energy, we obtain an extra chance of detecting the pollution with probability p. For each event $e \in E$, let s_e be the first sensor where the pollution is detected in that injection event. Note that s_e is a random variable. Let $z_\infty = \max_{e \in E, s \in S} z(s, e)$. Then, we define f as follows:

$$f(\boldsymbol{x}) = \mathbf{E}_{e \in E} \mathbf{E}_{s_e}[z_\infty - z(s_e, e)],$$

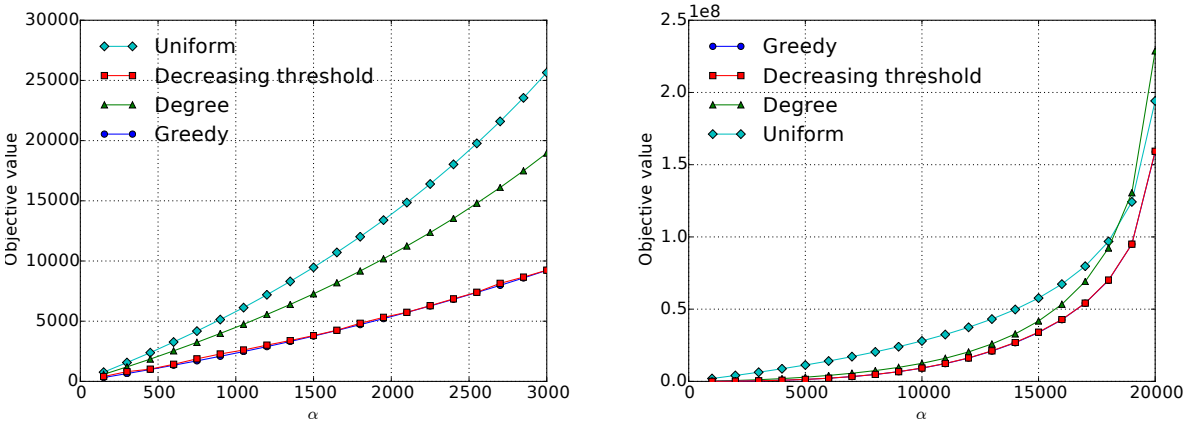
where $z(s_e, e)$ is defined as z_∞ when there is no sensor that managed to detect the pollution. Intuitively speaking, $\mathbf{E}_{s_e}[z_\infty - z(s_e, e)]$ expresses how much time we managed to save in the event e on average. Then, we take the average over all the events. A similar function was also used in [108] to measure the performance of a sensor allocation although they only considered the case $p = 1$. This corresponds to the case that by spending unit energy at a sensor s, we can always detect the pollution that has reached s. We note that $f(\boldsymbol{x})$ is DR-submodular (see Lemma 3.4.3 for the proof).

Budget allocation problem. In order to observe the behavior of our algorithm for large-scale instances, we created a synthetic instance of the budget allocation problem (see Section 3.2) as follows: The instance can be represented as a bipartite graph $(S, T; E)$, where S is a set of 5,000 vertices and T is a set of 50,000 vertices. We regard a vertex in S as an ad source, and a vertex in T as a person. Then, we fix the degrees of vertices in S so that their distribution obeys the power law of $\gamma := 2.5$; that is, the fraction of ad sources with out-degree d is proportional to $d^{-\gamma}$. For a vertex $s \in S$ of the supposed degree d, we choose d vertices in T uniformly at random and connect them to s with edges. We define a function $f : \mathbb{Z}_+^S \to \mathbb{R}_+$ as

$$f(\boldsymbol{x}) = \sum_{t \in T} \Big(1 - \prod_{s \in \Gamma(t)} (1-p)^{x(s)}\Big), \tag{5.4}$$

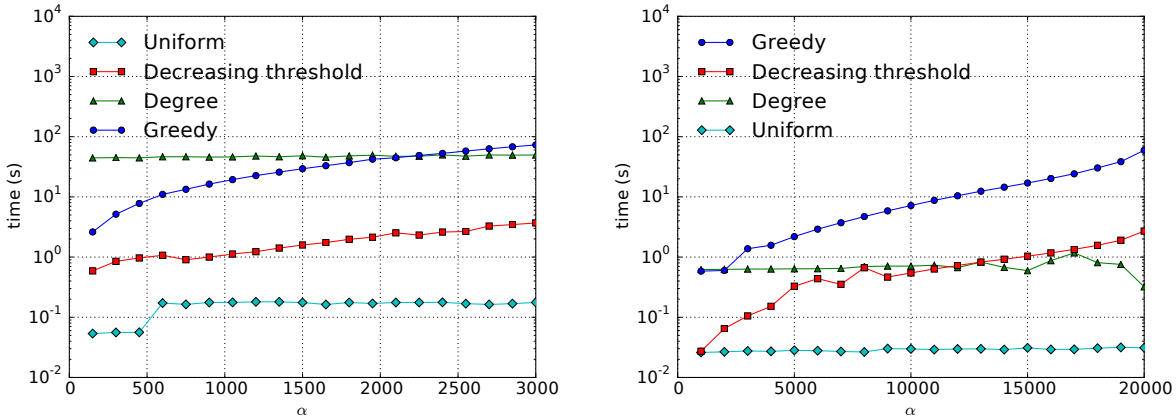
where $\Gamma(t)$ is the set of vertices connected to t and $p = 0.0001$. Here, we suppose that, by investing a unit cost to an ad source $s \in S$, we have an extra chance of influencing a person $t \in T$ with $s \in \Gamma(t)$ with probability p. Then, $f(\boldsymbol{x})$ can be seen as the expected number of people influenced by ad sources. By Lemma 3.3.4, f is a monotone DR-submodular function.

[1] Although three other values are provided, they showed similar empirical results and we omit them.



(a) Sensor placement (BWSN) (b) Budget allocation (synthetic)

Figure 5.1: Objective values



(a) Sensor placement (BWSN) (b) Budget allocation (synthetic)

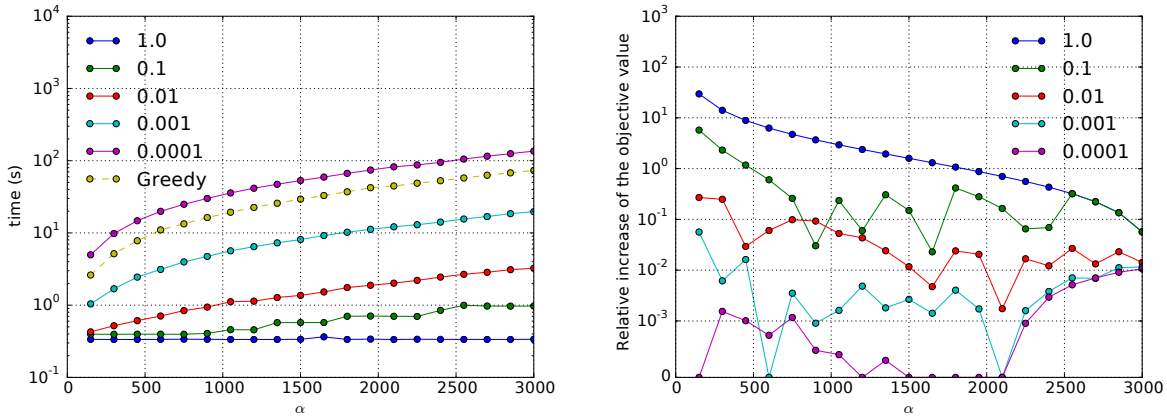
Figure 5.2: Runtime

5.3.2 Experimental Results

Figure 5.1 illustrates the obtained objective value $\|\boldsymbol{x}\|_1$ for various choices of the worst quality guarantee α on each dataset. We chose $\epsilon = 0.01$ in Decreasing threshold. We can observe that Decreasing threshold attains almost the same objective value as Greedy, and it outperforms Degree and Uniform.

Figure 5.2 illustrates the runtime for various choices of the worst quality guarantee α on each dataset. We chose $\epsilon = 0.01$ in Decreasing threshold. We can observe that the runtime growth of Decreasing threshold is significantly slower than that of Greedy.

Figures 5.3a and 5.3b show the relative increase of the objective value and the runtime, respectively, of our method against Greedy on the BWSN dataset. We can observe that the relative increase of the objective value gets smaller as α increases. This phenomenon can be well explained by considering the extreme case that $\alpha = \max f(r\mathbf{1})$. In this case, we need to choose $\boldsymbol{x} = r\mathbf{1}$ anyway in order to achieve the worst quality guarantee, and the order of



(a) Relative cost increase

(b) Runtime

Figure 5.3: Effect of ϵ

increasing coordinates of $\boldsymbol{x}$ does not matter. Also, we can see that the empirical runtime grows as a function of $\frac{1}{\epsilon}$, which matches our theoretical bound.

Chapter 6

Introduction to Matrix Completion

This chapter provides a literature overview on matrices with indeterminates and matrix completion.

6.1 Matrices with Indeterminates and Matrix Subspace

Let $\mathbb{F}$ be a field and $x_1, \ldots, x_d$ be indeterminates. A *matrix with indeterminates* over $\mathbb{F}$ is a matrix whose entry is a multilinear polynomial in $x_1, \ldots, x_d$ over $\mathbb{F}$. A matrix is said to be *generic* if the set of its nonzero entries are algebraically independent over $\mathbb{F}$.

In the most of previous work on matrices with indeterminates and this dissertation, entries of a matrix with indeterminates are *linear* polynomials, i.e., polynomials of degree at most one. Such a matrix with indeterminates can be written as $B_0 + x_1B_1 + \cdots + x_dB_d$, where $B_0, B_1, \ldots, B_d \in \mathbb{F}^{m\times n}$ are coefficient matrices. If $B_0 = O$, it can be regarded as a *matrix subspace* in $\mathbb{F}^{m\times n}$

$$\mathcal{M} = \mathrm{span}\{B_1, \ldots, B_d\} := \{\alpha_1B_1 + \cdots + \alpha_dB_d : \alpha_1, \ldots, \alpha_d \in \mathbb{F}\}.$$

Example 6.1.1 (Edmonds Matrix [56])**.** Let $G = (S, T; E)$ be a bipartite graph. The *Edmonds matrix* of G is an $S \times T$ matrix A with entries

$$A_{ij} = \begin{cases} x_{ij} & \text{if } ij \in E, \\ 0 & \text{otherwise,} \end{cases}$$

where x_{ij} $(ij \in E)$ are independent indeterminates. Edmonds [56] proved that the rank of the Edmonds matrix equal to the size of maximum matching in G.

Example 6.1.2 (Mixed Matrix [125])**.** The mixed matrix was introduced by Murota and Iri [126], for system analysis of electric circuits. Formally, a mixed matrix over $\mathbb{F}$ is a matrix $A = Q + T$, where Q is a constant matrix over $\mathbb{F}$ and T is a generic matrix. One can easily see that an Edmonds matrix is a special case of a mixed matrix. Mixed matrices admit elegant combinatorial structures in terms of matroids (for more details, see a monograph of Murota [125]).

Example 6.1.3 (Tutte Matrix [165])**.** The *Tutte matrix* was introduced by Tutte [165] to study the maximum matching problem. Let $G = (V, E)$ be a simple graph, i.e., a graph without self-loops and multiple edges. The Tutte matrix of G is a $V \times V$ skew-symmetric matrix T such that

$$T_{ij} = \begin{cases} x_{ij} & \text{if } ij \in E \text{ and } i > j, \\ -x_{ij} & \text{if } ij \in E \text{ and } j > i, \\ 0 & \text{otherwise,} \end{cases}$$

where x_{ij} ($ij \in E$, $i > j$) are independent indeterminates. Tutte [165] proved that the size of a maximum matching in G equals to the half of the rank of the Tutte matrix T. Note that this theorem of Tutte [165] was proved before Edmonds [56].

6.2 Max-Rank Matrix Completion

Let A be a matrix with indeterminates over $\mathbb{F}$. The task of *max-rank matrix completion* is to find a value assignment for the indeterminates that maximizes the rank of the resulting matrix.

6.2.1 Lovász's Randomized Algorithm

Lovász [115] presented a randomized algorithm for max-rank matrix completion. Originally his algorithm was described for the Tutte matrix but is applicable to arbitrary matrices with indeterminates. The idea of Lovász's algorithm is extremely simple; just pick random values from $\mathbb{F}$ and substitute them to the indeterminates. He showed that this randomized algorithm yields a solution with high probability if the size of $\mathbb{F}$ is sufficiently large. At the heart of his analysis lies the following *Schwartz-Zippel* lemma.

Lemma 6.2.1 (The Schwartz-Zippel Lemma [147], [178])**.** *Let $p \in \mathbb{F}[x_1, \ldots, x_d]$ be a nonzero polynomial of degree k and $U \subseteq \mathbb{F}$ be a finite set. If we choose $\alpha_1, \ldots, \alpha_d$ from U uniformly at random, then we have*

$$\Pr(p(\alpha_1, \ldots, \alpha_d) = 0) \leq \frac{k}{|U|}. \tag{6.1}$$

Let A be a given matrix with indeterminates of size $n \times n$. Applying the Schwartz-Zippel lemma for the determinant of A, we can see that the success probability of Lovász's randomized algorithm is at least $1/2$, if we take $U \subseteq \mathbb{F}$ with $|U| \geq 2 \deg \det A$. Especially if we use Lovász's randomized algorithm to the Tutte matrix, we obtain a very simple randomized algorithm for maximum matching in a general graph.

6.2.2 Deterministic Algorithms for Max-Rank Matrix Completion

The Lovász's algorithm requires that the field is sufficiently large and the algorithm can access to random bits. One can naturally ask if we relax this restriction. Indeed this is possible for some classes of matrices; i.e., some classes of matrices admit a deterministic polynomial-time algorithm for max-rank completion over an arbitrary field. We review these classes in detail below.

Mixed Matrix The first deterministic max-rank completion algorithm for a mixed matrix[1] was presented by Geelen [70]. His algorithm takes $O(n^9)$ time and works only over a field of size at least n, where n is the maximum of row and column size of a given matrix. Later an $O(n^{2.77})$ time algorithm for an arbitrary field was devised by Harvey, Karger, and Murota [80]. The detail of their algorithm will be described in Chapter 7.

[1] In the context of matrix completion, we assume that a given mixed matrix is one in which each entry depends on at most one indeterminates. This is a somewhat technical requirement but does not matter in many applications.

Matrix Completion by Rank-one Matrices In this problem, a given matrix is in the form of $B_0 + x_1 B_1 + \cdots + x_d B_d$, where $B_1, \ldots, B_d$ are rank-one matrices. If $B_0 = O$, this is equivalent to find a matrix of maximum rank from a matrix subspace $\mathrm{span}\{B_1, \ldots, B_d\}$. The case of $B_0 = 0$ was considered by Lovász [117], he showed that this case reduces to linear matroid intersection and therefore this case can be deterministically solved in polynomial time. For the general case, Ivanyos, Karpinski, and Saxena [89] devised a first deterministic polynomial-time algorithm with $O(n^{4.38}d)$ running time. Later Soma [152] improved the running time to $O((n+d)^{2.77})$ by showing that this case boils down to max-rank matrix completion for a mixed matrix.

Mixed Skew-Symmetric Matrices In this problem, a given matrix is a *mixed skew-symmetric matrix* [125], a matrix $A = Q + T$, where Q is a constant skew-symmetric matrix and T is a skew-symmetric matrix such that the variables T_{ij} ($T_{ij} \neq 0, i > j$) are algebraically independent over $\mathbb{F}$. Soma [152] presented a deterministic polynomial-time algorithm with $O(n^4)$ running time.

Skew-Symmetric Matrix Completion by Rank-Two Skew-Symmetric Matrices This problem is a skew-symmetric version of matrix completion by rank-one matrices. A given matrix is in the form of $B_0 + x_1 B_1 + \cdots + x_d B_d$, where B_0 is skew-symmetric and $B_1, \ldots, B_d$ are rank-two skew-symmetric matrices. Soma [152] showed that this problem can be reduced to max-rank matrix completion for a mixed skew-symmetric matrix, as in matrix completion by rank-one matrices.

6.2.3 Simultaneous Max-Rank Matrix Completion

Harvey, Karger, and Murota [80] introduced more general setting on max-rank matrix completion, namely, *simultaneous max-rank matrix completion*. In this generalized problem, we are given a collection $\mathcal{A}$ of matrices with indeterminates rather than a single matrix. Note that we allow that indeterminates can be shared by multiple matrices. The objective is to find a value assignment of the indeterminates that maximizes the rank of *every* resulting matrix obtained by substitution. In other words, the value assignment must be a solution of max-rank matrix completion for every matrix in the collection. Interestingly, for the classes of matrices described above admitting a deterministic polynomial-time algorithm, one can also solve the simultaneous version, if $|\mathbb{F}| > |\mathcal{A}|$ [80], [152].

6.2.4 Hardness of Max-Rank Matrix Completion

The hardness result of max-rank matrix completion was established by Harvey, Karger, and Yekhanin [81]. They showed that finding a max-rank completion over the binary field $\mathbb{F}_2$ is NP-hard if we allow indeterminates in a given matrix appears twice.

6.3 Low-Rank Matrix Completion

In low-rank matrix completion, we want to find values for indeterminates to minimize the rank of the resulting matrix. In what follows we assume that $\mathbb{F} = \mathbb{R}$ and that a given matrix is a mixed matrix, since most of the existing work deals with such a case. Without loss of generality, we can assume that each entry of a given mixed matrix is either a numeric value or an indeterminate. Let us denote by Ω the set indices of numeric entries in a given matrix A. Low-rank matrix completion can be formulated as follows.

$$\begin{aligned} \text{minimize} \quad & \operatorname{rank} X \\ \text{subject to} \quad & X_{ij} = A_{ij} \quad (ij \in \Omega) \end{aligned} \tag{6.2}$$

Example 6.3.1 (The Netflix Problem [3])**.** The *Netflix* problem is a problem for predicting users' preferences using only a few samples. In this problem, we are given a dataset containing each user's ratings on movies. Since the number of movies is huge, a user does not give rating on all the movies. The task is to predict unknown ratings with information of known ratings. Naturally we can construct a data matrix whose row and column indices are users and movies, respectively. Known and unknown ratings are numerical entries and indeterminates in the data matrix, respectively.

If users' ratings are uncorrelated at all, this problem is ill-posed. A reasonable assumption is that users' preferences depend on only a few hidden factors, e.g., theme, music, actors, directors, etc. Mathematically this observation can be modeled as the low-rank assumption; i.e., the true matrix $A \in \mathbb{R}^{m \times n}$ of users' ratings can be factored as $A = UV$, where $U \in \mathbb{R}^{m \times r}$, $V \in \mathbb{R}^{r \times n}$, and $r \ll m, n$ is the number of the hidden factors. Then this problem can be formulated as low-rank matrix completion.

6.3.1 Nuclear Norm Relaxation

Although the problem (6.2) is NP-hard in general, there are many algorithms for (approximately) solving it. The *nuclear norm relaxation* is the most popular approach, whose idea is replacing the rank function by its convex envelope, the nuclear norm. The convex relaxation of (6.2) is given by

$$\begin{aligned} \text{minimize} \quad & \|X\|_* \\ \text{subject to} \quad & X_{ij} = A_{ij} \quad (ij \in \Omega) \end{aligned} \tag{6.3}$$

The nuclear norm relaxation was inspired by ℓ_1 minimization in compressed sending; indeed we can view the nuclear norm heuristic as a matrix version of ℓ_1 minimization (see Chapter 9). The nuclear norm convex relaxation has demonstrated its theoretical and practical usefulness in matrix completion and other low-rank optimization tasks, and is nowadays accompanied by an endless array of optimization algorithms; see, e.g., [7], [30], [33], [38], [113], [114], [139], and [140] for a general overview.

Here, we review one of the most efficient algorithm due to Candés and Recht [33]. Their algorithm actually tries to solve the following modification of (6.3).

$$\begin{aligned} \text{minimize} \quad & \tau\|X\|_* + \frac{1}{2}\|X\|_F^2 \\ \text{subject to} \quad & X_{ij} = A_{ij} \quad (ij \in \Omega), \end{aligned} \tag{6.4}$$

where $\tau > 0$ is a parameter. Compared to (6.3), the objective function in (6.4) is strongly convex and thus (6.4) has the unique optimum. The key ingredient of their algorithm is the use of the *soft thresholding operator* (also known as the *shrinkage operator*) $\mathcal{S}_\tau$ [30]. Let $X = U\Sigma V^\top$ be a singular value decomposition of a matrix X with $\Sigma = \operatorname{diag}(\sigma_1, \dots, \sigma_{\operatorname{rank}(X)})$ and $\sigma_1 \geq \cdots \geq \sigma_{\operatorname{rank}(X)} > 0$. The value of $\mathcal{S}_\tau(X)$ is defined as

$$\mathcal{S}_\tau(X) = U\mathcal{S}_\tau(\Sigma)V^\top, \quad \mathcal{S}_\tau(\Sigma) = \operatorname{diag}(\sigma_1 - \tau, \dots, \sigma_{\operatorname{rank}(X)} - \tau)_+, \tag{6.5}$$

where $x_+ := \max(x, 0)$ and $\tau > 0$ is a thresholding parameter. Candés and Recht [33] proved that the thresholding operator for X yields the unique optimum of the following convex program.

$$\underset{Y}{\text{minimize}} \quad \tau\|Y\|_*^2 + \frac{1}{2}\|Y - X\|_F^2, \tag{6.6}$$

The algorithm of [33] starts with an arbitrary matrix X. Then apply the soft thresholding operator (with approximate choice of τ): $X \leftarrow \mathcal{S}_\tau(X)$. Since the new X may not be feasible, we modify $X_{ij} \leftarrow A_{ij}$ for each $ij \in \Omega$, and repeat this. In [33], they reported various examples that their algorithm effectively finds optimal solutions.

6.3.2 Random Matrix and Observation Model

How good is the convex relaxation (6.3) of (6.2)? Surprisingly, if Ω is a *random* set of fairly large size and a given matrix satisfies some technical condition, (6.2) and its convex relaxation (6.3) are indeed equivalent [33]. More precisely, they assume that an underlying (constant) matrix $M \in \mathbb{R}^{m\times n}$ is generated as follows:

$$M = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top, \tag{6.7}$$

where $\{\boldsymbol{u}_i\}_{i=1}^r$ and $\{\boldsymbol{v}_i\}_{i=1}^r$ are uniformly drawn from the family of r orthogonal vectors in $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively. Note that $\sigma_1, \ldots, \sigma_r$ are arbitrary constants. The observation set Ω is simply a uniformly random set over $[m] \times [n]$; i.e., some entries in M are independently replaced with indeterminates with some probability. Then they proved that if

$$|\Omega| \geq C n^{5/4} r \log n \tag{6.8}$$

then (6.3) has the unique solution that is M with high probability, where C is a global constant. Similar results for other generating models and algorithms were established in [38], [94].

Chapter 7

Faster Algorithm for Multicasting in Linear Deterministic Relay Network

In this chapter, we provide a deterministic polynomial time algorithm for multicasting in a linear deterministic relay network, a wireless communication framework proposed by Avestimehr, Diggavi and Tse [11]. The running time of our algorithm is faster than existing ones and matches the current best complexity of unicast computations for each sink. Our approach is based on the polylinking flow model of Goemans, Iwata and Zenklusen [72], and the mixed matrix completion technique of Harvey, Karger and Murota [80].

7.1 Introduction

Determining the capacity of a wireless information channel is a longstanding open problem. The difficulty comes from the two essential features of a wireless information channel: *broadcast* and *superposition*. In wireless communication, signals are sent to multiple users in the network, and superposition makes it hard to recover original signals. By these obstacles, even a simple wireless channel, such as network with a single source, a single relay, and a single sink, has not been fully characterized. This is in contrast to classic wired information channels, whose capacity can be characterized by the celebrated Ford-Fulkerson max-flow min-cut theorem.

A *linear deterministic relay network (LDRN)* was introduced by Avestimehr, Diggavi and Tse [11], to study the capacity of a wireless information channel. It captures the two main features of wireless communication, and they showed that a linear deterministic relay network approximates a wireless information channel with an additive constant factor. Furthermore, the capacity of a linear deterministic relay network can be characterized in terms of "source-destination cuts," which generalize the concept of cuts in a wired network. Although the original proof in [11] employed the *probabilistic method* and therefore the algorithmic aspects were not studied, deterministic algorithms to compute the capacity have been developed subsequently [6], [72].

Multicast algorithms in a linear deterministic relay network also have been developed. In the multicast problem, we are to find a linear coding scheme that transmits a message from a single source to desired multiple sinks in a network. Kim and Médard [98] showed that *random* linear coding (over a sufficiently large field) yields a multicast scheme with high probability. Ebrahimi and Fragouli [54], [55] presented the first *deterministic* multicast algorithm working on an arbitrary linear deterministic relay network. The fastest known deterministic algorithm was proposed by Yazdi and Savari [175]. The running time[1] is $O(dq((nr)^3 \log(nr) + n^2r^4))$,

[1] Although the running time stated in [175] is $O(dq((nr)^3 \log(nr) + n^2r^3 + nr^4))$, the complexity analysis there

where d is the number of sinks, q is the number of layers in the network, n is the maximum number of nodes in each layer, and r is the capacity of a node. Their method uses an efficient algorithm of Goemans, Iwata and Zenklusen [72] to compute a unicast flow for each sink, and then determines a linear coding scheme with the unicast flow information. All of these multicast algorithms assume that the field size is larger than the number of sinks.

7.1.1 Our Contribution

In this chapter, we develop a faster deterministic algorithm for constructing a multicast scheme in a linear deterministic relay network. Using the unicast algorithm of Goemans, Iwata and Zenklusen [72], our algorithm runs in $O(dq(nr)^3 \log(nr))$ time, while the running time of Yazdi and Savari [175] is $O(dq((nr)^3 \log(nr) + n^2r^4))$. Our algorithm is *optimal* in the sense that the simpler problem of unicast computations for d sinks already take $O(dq(nr)^3 \log(nr))$ time, as long as we use the unicast algorithm of Goemans, Iwata and Zenklusen [72]. Note that by the multicast theorem shown by Avestimehr, Diggavi and Tse [11], the unicast capacity for each sink can be obtained immediately once we solve the multicast problem. Also, it is another long-standing open problem to find a deterministic algorithm for computing multicast capacity without finding individual unicast capacity, even for a *wired* directed network.

We also compare the running time excluding the complexity of unicast flow computations. Our algorithm requires $O(dqn^3r^3)$ time, while the algorithm of [175] requires $O(dqn^2r^4)$ time. In this comparison, our algorithm is faster than that of Yazdi and Savari when $n = o(r)$. Typically in practice, r is the number of bits exchanged between two nodes and therefore it can be considerably greater than n, the maximum number of nodes in each layer.

While both Yazdi and Savari's algorithm and ours use the same unicast algorithm of Goemans, Iwata and Zenklusen [72], we achieve several technical improvements. The main differences of the present work are as follows:

- After finding a unicast flow for each sink, Yazdi and Savari's algorithm determines a linear coding scheme in a *node-by-node* manner, which increases the number of matrix operations that have to be carried out. To reduce this complexity, we introduce a *matrix completion technique* which enables us to determine a linear coding scheme of a *layer* at once.

- The method of Ebrahimi and Fragouli [54], [55] reduces multicasting in a linear deterministic relay network to a modified network coding problem and then applies the matrix completion technique to the modified problem. However, their approach does not consider the layered structure of linear deterministic relay networks, and therefore needs to handle a relatively large matrix. By exploiting the layered nature, we design an algorithm that only needs to handle smaller matrices.

A wireless network is not necessarily layered in general. Avestimehr, Diggavi, and Tse [11] showed that a general wireless network can be reduced to a layered network by considering the *time-expanded network*. Our method can take the advantage of the structure of their reduction. For example, the maximum number of nodes in each layer in the time-expanded network is equal to the number of nodes in the original wireless network. Since the time complexity of our algorithm depends on the maximum number of nodes in each layer, the increase of the time complexity is quite moderate. Note that the *total number* of nodes in the time-expanded network can be greatly increased through the reduction, and therefore the method of [54], [55] can be inefficient. Furthermore, even applied for the time-expanded network, our method is

counts a matrix multiplication of an $r \times n$ matrix and an $n \times r$ matrix by $O(nr)$, which is not generically true; We adopt the standard $O(nr^2)$ complexity.

still faster than Yazdi and Savari's method if the node capacity is larger than the number of nodes in the original wireless network.

7.1.2 Related Work

Combinatorial properties of linear deterministic relay networks have been studied in the literature. Goemans, Iwata and Zenklusen [72] studied more general flow model called a *polylinking network*, and showed that a unicast flow in a polylinking network can be found by solving the *submodular flow problem* [58]. Subsequently, Fujishige [67] slightly extended the polylinking network model and showed that the extended flow model is equivalent to the *neoflow* problems [66], which include the submodular flow problem and other variants. Note that similar results were obtained independently by Yazdi and Savari [174].

Connections between linear deterministic relay networks and *network coding* [4] were pointed out by several authors. In the original paper of Avestimehr, Diggavi and Tse [11], it is shown that the multicast capacity of a linear deterministic relay network equals the minimum unicast capacity for each sink. This result can be compared to the famous result for the multicast capacity achievable with network coding [4]. Ebrahimi and Fragouli [55] devised a multicast algorithm for a linear deterministic relay network based on reduction to the modified network coding problem, in which a network has nodes performing predetermined linear operation. Such extended network coding problems were studied by Király and Kovács [99].

7.2 Preliminaries

This section provides a brief summary in mixed matrix theory and mixed matrix completion. For further details of the mixed matrix theory, the reader is referred to the monograph of Murota [125]. For the sake of simplicity, we introduce some notation. For a matrix A, we denote by $\mathrm{Row}(A)$ the set of row indices of A. Similarly, we define $\mathrm{Col}(A)$ as the set of column indices of A.

7.2.1 Mixed Matrix

Let $\mathbb{F}$ be a field. A matrix A is a mixed matrix if $A = Q + T$, where Q is a constant matrix over $\mathbb{F}$ and T is a generic matrix (see Example 6.1.2). A mixed matrix $A = Q + T$ is called a *layered mixed matrix*, or, an *LM-matrix* for short, if the set of nonzero rows in Q and that in T are disjoint. In other words, an LM-matrix is a mixed matrix in the form of $A = \begin{bmatrix} Q \\ T \end{bmatrix}$. The following is basic in the mixed matrix theory.

Lemma 7.2.1 (Murota [125]). *For a square LM-matrix $A = \begin{bmatrix} Q \\ T \end{bmatrix}$, let $C := \mathrm{Col}(A)$, $R_Q := \mathrm{Row}(Q)$ and $R_T := \mathrm{Row}(T)$. Then A is nonsingular if and only if there exists a column subset $J \subseteq C$ such that both $Q[R_Q, C \setminus J]$ and $T[R_T, J]$ are nonsingular.*

In fact, the set J described in the lemma can be found by solving the *independent matching problem* [171], a generalization of bipartite matching. Formally, the independent matching problem is defined as follows. Let $G = (V^+, V^-; E)$ be a bipartite graph with vertex set $V^+ \dot\cup V^-$ and edge set E. Let $\mathbf{M}^+$ and $\mathbf{M}^-$ be matroids on V^+ and V^-, respectively. A matching M in G is said to be *independent* if the sets of vertices in V^+ and V^- incident to M are independent in $\mathbf{M}^+$ and in $\mathbf{M}^-$, respectively. The independent matching problem is to find an independent matching of maximum size.

In the mixed matrix theory, the independent matching problem is often used in the following form. Define a bipartite graph $G = (V^+, V^-; E)$ with $V^+ := C_Q \dot\cup R_T$ and $V^- := C$, where

C_Q is a copy of $C = \mathrm{Col}(A)$. Let E be the set $\{ij : i \in R_T, j \in C \text{ and } T_{ij} \neq 0\} \cup \{j_Q j : j_Q \text{ is the copy of } j\}$. Define $\mathbf{M}^+$ to be the direct sum of the vector matroid of Q and the free matroid on R_T (see Example 2.2.1). Let $\mathbf{M}^-$ be the free matroid on V^-. Then, a matching M is independent if and only if $\partial^+ M \cap C_Q$ is an independent set in the vector matroid of Q, where $\partial^+ M$ is the set of vertices in V^+ incident to M. Then the set J described in Lemma 7.2.1 coincides with the set of vertices matched to R_T by a maximum independent matching.

Conversely, if we have a column subset $J \subseteq C$ such that both $Q[R_Q, C \setminus J]$ and $T[R_T, J]$ are nonsingular, the corresponding independent matching can be found as follows. Since $T[R_T, J]$ is nonsingular, $G[R_T \dot\cup J]$ has a bipartite matching N such that $J \subseteq \partial N$, where $G[R_T \dot\cup J]$ is the subgraph of G induced by $R_T \dot\cup J$. Then $M := N \dot\cup \{j_Q j : j \in C \setminus J\}$ is a maximum matching in G and the set of vertices matched to R_T by M coincides with J.

Example 7.2.2. The bipartite graph G corresponding to the LM-matrix

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ x & y & z & 0 \\ w & 0 & 0 & t \end{bmatrix} \tag{7.1}$$

is shown in Figure 7.1. An independent matching is shown in thick edges, and the corresponding subset $J \subseteq C$ is shown in a shaded box.

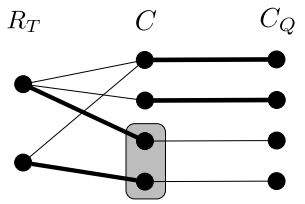


Figure 7.1: The corresponding bipartite graph of the LM-matrix A in Example 7.2.2.

7.2.2 Mixed Matrix Completion

Let us make a brief overview on simltaneous mixed matrix completion by Harvey, Karger, and Murota [80]. In simultaneous mixed matrix completion, we are given a *collection* of mixed matrices and some indeterminates may appear in more than one of these matrices. The objective is to find a value assignment of the indeterminates that maximizes the rank of *every* resulting matrix obtained by substitution.

For each mixed matrix $A = Q + T$ in a collection, where we assume that A is an $n \times n$ matrix for simplicity, we define the corresponding LM-matrix $\tilde{A}$ by

$$\tilde{A} = \begin{bmatrix} I & Q \\ Z & ZT \end{bmatrix}, \tag{7.2}$$

where Z is a nonsingular diagonal matrix whose nonzero entries are independent indeterminates. Note that $\operatorname{rank} \tilde{A} = n + \operatorname{rank} A$. Let M be a maximum independent matching in the bipartite graph G corresponding to the LM-matrix $\tilde{A}$. Roughly speaking, the independent matching M captures a combinatorial structure of the LM-matrix $\tilde{A}$. Harvey, Karger and Murota [80] showed that we can determine a value of each indeterminate with clever use of the combinatorial structure. The complexity of their algorithm is $O(|\mathcal{A}| \cdot (\mathrm{IM}(n) + kn^2))$ time, where $\mathrm{IM}(n)$ is the complexity of solving the independent matching problem for a single mixed matrix and k is the

number of indeterminates in $\mathcal{A}$. For example, in a standard augmenting path algorithm for the independent matching problem, we have $\mathrm{IM}(n) = O(n^3 \log n)$ [51].

Furthermore, this running time can be improved if the collection admits a certain structure. A collection of mixed matrices is said to be *column-compatible* if the following condition holds: for arbitrary two indeterminates, if some matrix contains them in the same column, then no matrix in the collection contains them in distinct columns.

Theorem 7.2.3 (Harvey, Karger and Murota [80]). *Let $\mathbb{F}$ be a field and let $\mathcal{A}$ be a column-compatible collection of $n \times n$ mixed matrices. If $|\mathbb{F}| > |\mathcal{A}|$, then there exists a solution of the simultaneous matrix completion for $\mathcal{A}$ and it can be found in $O(|\mathcal{A}| \cdot (\mathrm{IM}(n) + kn + n^3))$ time, where k is the number of indeterminates in $\mathcal{A}$. Using a standard independent matching algorithm, the algorithm can be implemented to run in $O(|\mathcal{A}| \cdot n^3 \log n)$ time.*

7.2.3 Cauchy-Binet Formula

The Cauchy-Binet formula describes the terms in the determinant of the product of two rectangular matrices.

Lemma 7.2.4 (Cauchy-Binet Formula). *For an $n \times r$ matrix A and an $r \times n$ matrix B with $n \leq r$, we have*

$$\det AB = \sum_{J:|J|=n} \det A[R, J] \cdot \det B[J, C], \tag{7.3}$$

where $R := \mathrm{Row}(A)$, $C := \mathrm{Col}(B)$ and the sum is taken over all the subsets J of size n in $\mathrm{Col}(A) = \mathrm{Row}(B)$.

7.3 Flow Model

This section provides the flow model for multicast in a linear deterministic relay network. The model is based on the linking network model of Goemans, Iwata and Zenklusen [72].

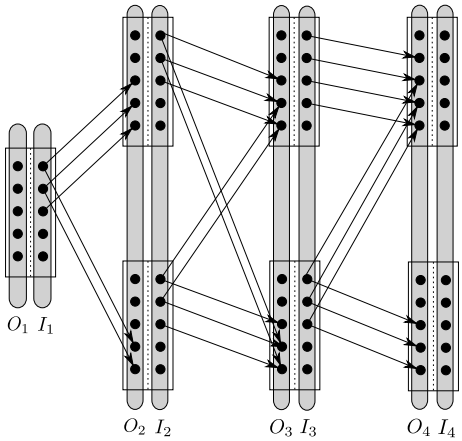


Figure 7.2: An example of a linear deterministic relay network

A linear deterministic relay network is a layered network with node set $V = V_1 \dot{\cup} \ldots \dot{\cup} V_q$, where V_i is the set of nodes in the ith layer for $i = 1, \ldots, q$. Each node consists of r *input* vertices and r *output* vertices, for some global constant $r \in \mathbb{Z}_+$. For each i, let I_i and O_i be the set of all the inputs and the set of all the outputs in the layer V_i, respectively. We may assume that the first layer V_1 consists of a single node s called the *source node*.

Signals are modeled as elements of a finite field $\mathbb{F}$ and are sent as follows. For each i, let $\boldsymbol{z}_i$ denote the vector consisting of signals at the inputs in the layer V_i and let $\boldsymbol{y}_{i+1}$ denote the vector of signals received in the outputs of the next layer V_{i+1}. Then $\boldsymbol{y}_{i+1}$ is equal to $M_i\boldsymbol{z}_i$, where M_i is a given $O_{i+1} \times I_i$ matrix over $\mathbb{F}$ representing a "connection" between V_i and V_{i+1}. In the original paper [11] of a linear deterministic relay network, M_i is represented as a block diagonal matrix of *shift* matrices. That is, for $u \in V_i$ and $v \in V_{i+1}$, $M_i[u, v] = S^{r-n_{uv}}$ for some integer n_{uv}, where S is the following $r \times r$ matrix:

$$S = \begin{bmatrix} 0 & & & & \\ 1 & 0 & & & \\ & 1 & 0 & & \\ & & \ddots & \ddots & \\ & & & 1 & 0 \end{bmatrix}. \tag{7.4}$$

However, we consider more general settings in this chapter; i.e., we assume that M_i can be an arbitrary matrix.

Let t be a node in the layer V_k. An *s-t flow* F in the linear deterministic network is a subset of $\cup_{i=1}^{q}(O_i \cup I_i)$ such that:

1. For each node, the number of outputs contained in F is equal to the number of inputs contained in F.
2. For each $i = 1, \ldots, k-1$, $M_i[F \cap O_{i+1}, F \cap I_i]$ is nonsingular.
3. The outputs of the sink t contains $F \cap O_k$.

The *rate* of an *s-t* flow F is the value $|F \cap O_1|$.

The *unicast* problem in a linear deterministic relay network is a problem to find an *s-t* flow for a single *s-t* pair. Goemans, Iwata and Zenklusen [72] showed that the unicast problem can be solved with *matroid partition*.

Theorem 7.3.1 (Goemans, Iwata and Zenklusen [72])**.** *An s-t flow in a linear deterministic relay network can be found in $O(q(nr)^3 \log(nr))$ time, where q is the number of layers, n is the maximum number of nodes in each layer, and r is the capacity of nodes.*

In the paper of Goemans, Iwata and Zenklusen [72], a node can only send its receiving signals. We cannot improve throughput by introducing linear coding operation in nodes in the unicast problem. However, in the multicast problem, linear coding in node improves throughput, in fact, we can achieve the maximum multicast throughput [11]. Thus it is natural to assume that a node can send a linear combination of its receiving signals. Formally, for each i, the input vector $\boldsymbol{z}_i \in \mathbb{F}^{I_i}$ is determined as $\boldsymbol{z}_i = X_i\boldsymbol{y}_i$, where $\boldsymbol{y}_i$ is the output vector of layer V_i and X_i is a block diagonal matrix each of whose block size is r. We can easily see that the output vector $\boldsymbol{y}_{i+1}$ of the next layer V_{i+1} is determined by $\boldsymbol{y}_{i+1} = M_iX_i\boldsymbol{y}_i$.

The *multicast* problem in a linear deterministic relay network is defined as follows. Givens are a linear deterministic relay network $\mathcal{N}$ and a set T of sink nodes in $\mathcal{N}$. We consider the source node s as an information source generating a *message* $\boldsymbol{w} \in \mathbb{F}^r$. The objective is to design coding matrices $X_1, \ldots, X_{q-1}$ so that each sink t in T can decode $\boldsymbol{w}$ from its receiving signals for an *arbitrary* $\boldsymbol{w}$.

7.4 Algorithm

In this section, we describe a new algorithm for finding a multicast scheme in a linear deterministic relay network. For simplicity of description, we assume that there exists an *s-t* flow with

rate r for each sink t. Let $\boldsymbol{w} \in \mathbb{F}^r$ denote the message vector, T denote the set of sinks and let d denote the number of sinks.

Our algorithm splits into two parts. In the first part, we compute an s-t flow F_t in the linear deterministic relay network for each sink t in T. This part can be done efficiently using the unicast algorithm of Goemans, Iwata and Zenklusen [72]. In the second part, we determine linear coding coefficients from the first layer to the last layer. More precisely, we determine the entries of the linear coding coefficients matrix X_i so that the following condition is satisfied: the original message vector $\boldsymbol{w}$ can be recovered from subvector $\boldsymbol{y}_{i+1}[F_t \cap O_{i+1}]$ for each sink t. If we can find $X_1, \ldots, X_{q-1}$ satisfying the condition, then they form a multicast coding scheme. To see this, suppose that a sink t is in the kth layer V_k for some k. Then t can recover the original message $\boldsymbol{w}$ because the set of output vertices of s contains $F_t \cap O_k$, if the condition holds. The following lemma summarizes the above arguments.

Lemma 7.4.1. *If $|\mathbb{F}| > d$, there exist constant matrices $X_1, \ldots, X_{q-1}$ such that $\boldsymbol{w}$ can be recovered from $\boldsymbol{y}_i[F_t \cap O_i]$ for $i = 1, \ldots, q$ and for each $t \in T$.*

We will prove the lemma by induction on i. For $i = 1$, $\boldsymbol{w}$ can be trivially recovered from $\boldsymbol{y}_1[F_t \cap O_1] = \boldsymbol{y}_1$ because we have $\boldsymbol{y}_1 = \boldsymbol{w}$ and $F_t \cap O_1 = O_1$. Suppose that $i > 1$; i.e., $X_1, \ldots, X_{i-1}$ are fixed. We will find a constant matrix X_i such that $\boldsymbol{w}$ can be recovered from $\boldsymbol{y}_i[F_t \cap O_i]$. At the moment, we do not know X_i, so we regard all the potentially nonzero entries of X_i as independent indeterminates. To distinguish the constant coding matrix X_i, we denote by $\boldsymbol{X}_i$ the matrix obtained by regarding all the potentially nonzero entries as independent indeterminates. Equivalently, $\boldsymbol{X}_i$ is the generic block diagonal matrix such that each block corresponds to a node in the ith layer and has no nonzero entry. In what follows, we will find a suitable value assignment for the indeterminates in $\boldsymbol{X}_i$. Let P_i be the matrix satisfying that $\boldsymbol{y}_i = P_i \boldsymbol{w}$. By the induction hypothesis, $P_i[F_t \cap O_i, O_1]$ is nonsingular. Since $\boldsymbol{y}_{i+1} = M_i \boldsymbol{X}_i \boldsymbol{y}_i = M_i \boldsymbol{X}_i P_i \boldsymbol{w}$, we can see that the original message $\boldsymbol{w}$ can be recovered from subvector $\boldsymbol{y}_{i+1}[F_t \cap O_{i+1}]$ if and only if $M_i[F_t \cap O_{i+1}, I_i]\boldsymbol{X}_i P_i$ is nonsingular.

Let A_t be the mixed matrix defined as follows:

$$A_t := \begin{bmatrix} I & O & P_i \\ \boldsymbol{X}_i & I & O \\ O & M_i[F_t \cap O_{i+1}, I_i] & O \end{bmatrix}. \tag{7.5}$$

The matrix A_t is nonsingular if and only if $M_i[F_t \cap O_{i+1}, I_i]\boldsymbol{X}_i P_i$ is nonsingular, which can be verified, for example, by the Gaussian elimination. Therefore, if there exists a value assignment to the indeterminates of $\boldsymbol{X}_i$ such that A_t is nonsingular (after substitution) for each sink t, then the lemma is proved. Finding such values for the indeterminates in $\boldsymbol{X}_i$ boils down to simultaneous mixed matrix completion. In order to prove that there exists a solution for the simultaneous matrix completion, we have to check that A_t is a nonsingular mixed matrix. From the above observation, it suffices to show that $M_i[F_t \cap O_{i+1}, I_i]\boldsymbol{X}_i P_i$ is nonsingular.

Lemma 7.4.2. *The matrix $M_i[F_t \cap O_{i+1}, I_i]\boldsymbol{X}_i P_i$ is nonsingular; i.e., $\det M_i[F_t \cap O_{i+1}, I_i]\boldsymbol{X}_i P_i$ is a nonzero polynomial.*

Proof. By the Cauchy-Binet formula (Lemma 7.2.4), we have

$$\begin{aligned} &\det M_i[F_t \cap O_{i+1}, I_i]\boldsymbol{X}_i P_i \\ &= \sum \det M_i[F_t \cap O_{i+1}, I] \det \boldsymbol{X}_i[I, J] \det P_i[J, O_1], \end{aligned} \tag{7.6}$$

where the sum is taken over all the subsets $I \subseteq I_i$ and $J \subseteq O_i$ with $|I| = |J| = r$. No cancellation occurs among nonzero terms in the right-hand side because nonzero entries of $\boldsymbol{X}_i$

Algorithm 7.1 An algorithm for multicasting in a linear deterministic relay network

1: Compute an s-t flow F_t for each sink $t \in T$.
2: Set $U := T$ and $P_1 := I$.
3: **for** $i = 1$ to q :
4: Let $\mathcal{A}_i := \{A_t : t \in U\}$.
5: Compute a solution of simultaneous mixed matrix completion for $\mathcal{A}_i$.
6: Let $P_{i+1} := M_i X_i P_i$, where X_i is the substituted matrix according to the solution of simultaneous matrix completion.
7: Remove each sink t from U if t is in the ith layer.

are independent indeterminates and distinct nonzero terms contain at least one distinct nonzero entry from $\boldsymbol{X}_i$. On the other hand, $M_i[F_t \cap O_{i+1}, F_t \cap I_i]$ and $\boldsymbol{X}_i[F_t \cap I_i, F_t \cap O_i]$ are nonsingular because F_t is a flow. Furthermore, $P_i[F_t \cap O_i, O_1]$ is nonsingular by the inductive hypothesis. Thus if we take $I = F_t \cap I_i$ and $J = F_t \cap O_i$, the corresponding term is nonzero, and this implies that $\det M_i[F_t \cap O_{i+1}, I_i]\boldsymbol{X}_i P_i$ is a nonzero polynomial. □

We are now ready to prove Lemma 7.4.1. Let $\mathcal{A}_i$ be the collection of mixed matrix A_t for each sink t such that t is in V_j with $j > i$. Since $|\mathbb{F}| > d$ and each mixed matrix A_t is nonsingular, there exists a value assignment for $\boldsymbol{X}_i$ such that every resulting matrix is nonsingular. Therefore, $\boldsymbol{w}$ can be recovered from $\boldsymbol{y}_{i+1}[F_t \cap O_{i+1}]$ for each t, which proves the lemma.

The above arguments provide an algorithm for finding a multicast encoding scheme, whose pseudocode description is presented in Algorithm 7.1. Let us analyze the running time of the algorithm. A unicast flow can be found in $O(q(nr)^3 \log(nr))$ time for each sink, by the algorithm of Theorem 7.3.1. Using the simultaneous mixed matrix completion algorithm of Theorem 7.2.3, linear encoding matrix X_i can be found in $O(d(nr)^3 \log(nr))$ time, for each layer i. Note that the collection of A_t's is column compatible since each indeterminate appears in exactly the same column among A_t's.

Theorem 7.4.3. *If $|\mathbb{F}| > d$, a multicast linear encoding scheme over $\mathbb{F}$ for a linear deterministic relay network can be found in $O(dq(nr)^3 \log(nr))$ time, where d is the number of sinks, q is the number of layers, n is the maximum number of nodes in each layer, and r is the capacity of a node.*

7.5 Complexity Excluding Unicast Computation

In this section, we estimate the complexity of our multicast algorithm excluding the complexity of unicast computations. Of course, as we have argued in the previous section, we can find a multicast encoding scheme by solving the simultaneous matrix completion repeatedly. However, this straightforward algorithm requires $O(dq(nr)^3 \log(nr))$ time. In fact, using the unicast flow information, we do not need to solve the independent matching problems in the simultaneous matrix completion. More precisely, an independent matching associated with the matrix A_t can be found in $O(1)$ time. Therefore, if we know a unicast flow for each sink, we can skip the computation of independent matching in the simultaneous matrix completion algorithm.

Let us transform the mixed matrix A_t into the following LM-matrix:

$$\tilde{A}_t := \begin{bmatrix} I & O & P_i \\ O & M_i[F_t \cap O_{i+1}, I_i] & O \\ Z\boldsymbol{X}_i & Z & O \end{bmatrix}, \tag{7.7}$$

where Z is a nonsingular diagonal matrix whose nonzero entries are independent indeterminates. Then A_t is nonsingular if and only if $\tilde{A}_t$ is nonsingular. We denote $\tilde{A}_t = \begin{bmatrix} Q \\ S \end{bmatrix}$ as usual. Let us denote $R_Q := \mathrm{Row}(Q)$, $R_S := \mathrm{Row}(S)$, and $C := \mathrm{Col}(\tilde{A}_t)$. The proof of the following lemma is almost same as that of Lemma 7.4.2.

Lemma 7.5.1. *For an s-t flow F_t, put $J := (F_t \cap O_i) \dot{\cup} (I_i \setminus F_t)$. Then $Q[R_Q, C \setminus J]$ and $S[R_S, J]$ are both nonsingular.*

Let G be the bipartite graph $G = (V^+, V^-; E)$ corresponding to the LM-matrix $\tilde{A}_t$ as defined in Section 7.2.1. The subset J described in Lemma 7.2.1 corresponds to the set of vertices in V^+ incident to some maximum independent matching, say M. Furthermore, we can find the maximum independent matching M as follows. Observe that it is sufficient to find a maximum matching in $G[R_S \dot{\cup} J]$, where $G[R_S \dot{\cup} J]$ is the subgraph of G induced by $R_S \dot{\cup} J$. Since $Z\boldsymbol{X}_i$ is block diagonal and each block has no zero entries, $G[R_S \dot{\cup} J]$ is the direct sum of complete bipartite graphs. Thus we can find a maximum matching in $G[R_S \dot{\cup} J]$ immediately.

Let us analyze the running time of our algorithm. For each i, the simultaneous matrix completion can be done in $O(dn^3r^3)$ time by the algorithm of Theorem 7.2.3 because the collection of A_t's is column-compatible and we can consider that $\mathrm{IM}(n) = O(1)$. Computing P_{i+1} can be done in $O(n^2r^3)$ time. In total, a linear coding scheme in the ith layer can be found in $O(dn^3r^3)$ time. Summing up $i = 1$ to $q-1$, we can find an entire linear coding scheme in $O(qdn^3r^3)$ time. Summarizing the above arguments, we have the following refinement of Theorem 7.4.3.

Theorem 7.5.2. *If $|\mathbb{F}| > d$, a multicast encoding scheme in a linear deterministic relay network can be found in $O(d \cdot \mathrm{UF}(n,q,r) + qdn^3r^3)$ time if, where $\mathrm{UF}(n,q,r)$ is the complexity of unicast computation for a single sink, d is the number of sinks, q is the number of layers, n is the maximum number of nodes in each layer, and r is the capacity of nodes.*

7.6 Concluding Remarks

We studied multicasting in a linear deterministic relay network. Using the simultaneous matrix completion, we devised a new algorithm whose complexity matches to that of unicast computations for each sinks. We also estimated the complexity of our algorithm excluding the complexity of unicast computations.

The mixed matrix A_t used in our algorithm has certain structures; in particular, A_t is a *sparse* matrix. In fact, the number of *nonzero* entries of A_t is $O(nr^2)$, while A_t has $O(n^2r^2)$ entries in total. However, our algorithm does not use the sparsity because Harvey, Karger and Murota's simultaneous mixed matrix completion algorithm is incompatible with sparsity. A possible direction for future studies is to design an algorithm compatible with the sparsity of the mixed matrix A_t. For example, can one design a faster algorithm which runs in $O(d \cdot \mathrm{UF}(n,q,r) + qdn^2r^3)$ time?

Chapter 8

The Low-Rank Basis Problem for a Matrix Subspace

In this chapter, we consider the following problem, which we call the *low-rank basis problem*. Let $\mathcal{M}$ be a matrix subspace in $\mathbb{R}^{m\times n}$ of dimension d. The goal is to

$$\begin{array}{ll} \text{minimize} & \operatorname{rank}(X_1) + \cdots + \operatorname{rank}(X_d) \\ \text{subject to} & \operatorname{span}\{X_1, \ldots, X_d\} = \mathcal{M}. \end{array} \tag{8.1}$$

As we will see below, this problem turns out to have various applications in machine learning, signal processing, and matrix completion. We devise a practical heuristic algorithm for this problem and verify its effectiveness with numerical experiments on synthetic and real data.

8.1 Introduction

Finding a succinct representation of a given object has been one of the central tasks in the computer and data sciences. For vectors, sparsity (i.e., ℓ_0-norm) is a common measure of succinctness. For example, exploiting prior knowledge on sparsity of a model is now considered crucial in machine learning and statistics [29]. Although the naive penalization of the ℓ_0-norm easily makes the problem intractable, it turns out that exploiting the ℓ_1-norm as a regularizer yields a tractable convex problem and performs very well in many settings [32], [37]. This phenomenon is strongly related to compressed sensing, which shows under reasonable assumptions that the ℓ_1-recovery almost always recovers a sparse solution for an undetermined linear system. Since matrices are more complex objects, one may consider different criteria on succinctness for matrices, namely the rank. Interestingly, a variety of concepts from sparse vector recovery carry over to low-rank matrix recovery, for which the nuclear norm plays a role analogous to the ℓ_1-norm for sparse vectors [59], [60].

Recently, the *sparse vector problem* has been studied by several authors [18], [79], [138], [155]. In the sparse vector problem, we are given a linear subspace S in $\mathbb{R}^n$, and the task is to find the sparsest *nonzero* vector in S. The celebrated results for ℓ_1-regularization are not directly applicable, and the sparse vector problem is less understood. The difficulty arises from the nonzero constraints; a natural ℓ_1-relaxation only yields the trivial zero vector. Thus the sparse vector problem is nonconvex in its own nature. A common approach is based on the *hypercontractivity*, that is, optimizing the ratio of two different norms. An algorithm that optimizes the ℓ_1/ℓ_∞-ratio is studied in [79], [155]. Optimization of the ℓ_4/ℓ_2-ratio was recently considered by Barak, Kelner, and Steurer [18]. A closely related problem is the *sparse basis problem*, in which we want to find a basis of S that minimizes the sum of the ℓ_0-norms. In

addition to imposing the sparsity of vectors as in the sparse vector problem, a new difficulty arises of controlling the linear independence of basis vectors in the sparse basis problem.

The low-rank basis problem (8.1) generalizes the sparse basis problem. To see this, it suffices to identify $\mathbb{R}^n$ with the subspace of diagonal matrices in $\mathbb{R}^{n\times n}$. As a consequence, any result on the low-rank basis problem (8.1) (including relaxations and algorithms) will apply to the sparse basis problem with appropriate changes in notation (matrix nuclear norm for diagonal matrices becomes ℓ_1-norm etc.). Conversely, some known results on the sparse basis problem may generalize to the low-rank basis problem (8.1). An obvious but important example of this logic concerns the complexity of the problem: It has been shown in Coleman and Pothen [49] that even if one is given the minimum possible value for $\|\boldsymbol{x}_1\|_0 + \cdots + \|\boldsymbol{x}_d\|_0$ in the sparse basis problem, it is NP-hard to find a sparse basis. Thus the low-rank basis problem (8.1) is also NP-hard in general.

A closely related (somewhat simpler) problem is the following *low-rank matrix problem*:

$$\begin{array}{ll} \text{minimize} & \operatorname{rank}(X) \\ \text{subject to} & X \in \mathcal{M}, X \neq O. \end{array} \tag{8.2}$$

This problem is a matrix counterpart of the sparse vector problem. Again, (8.2) is NP-hard [49], even if $\mathcal{M}$ is spanned by diagonal matrices. Note that our problem (8.2) does not fall into the framework of Cai, Candés, and Shen [30], in which algorithms have been developed for finding a low-rank matrix X in an affine linear space described as $\mathcal{A}(X) = \boldsymbol{b}$ (matrix completion problems are of that type). In our case we have $\boldsymbol{b} = \boldsymbol{0}$, but we are of course not looking for the zero matrix, which is a trivial solution for (8.2). This requires an additional norm constraint, and modifications to the algorithms in [30].

8.1.1 Our Contribution

We propose an alternating direction algorithm for the low-rank basis problem that does (i) rank estimation, and (ii) obtains a low-rank basis. We also provide convergence analysis for our algorithm. Our algorithm is based on a greedy process, whose use we fully justify. In each greedy step we solve the low-rank matrix problem (8.2) in a certain subspace, and hence our algorithm can also solve the low-rank matrix problem.

Our methods are iterative, and switch between the search of a good low-rank matrix and the projection on the admissible set. The second step typically increases the rank again. The solution would be a fixed point of such a procedure. We use two phases with different strategies for the first step, i.e., finding a low-rank matrix.

In the first phase we find new low-rank guesses by applying the singular value shrinkage operator (called *soft thresholding*) considered in Cai, Candés, and Shen [30]. In combination with the subspace projection, this results in the matrix analog of the alternating direction algorithm proposed very recently by Qu, Sun, and Wright [138] for finding sparse vectors in a subspace. An additional difficulty, however, arises from the fact that we are required to find more than one linearly independent low-rank matrix in the subspace. Also note that our algorithm adaptively changes the thresholding parameter during its execution, which seems necessary for our matrix problem, although [138] fixes the thresholding parameter before starting their algorithm. In our experiments it turns out that the use of the shrinkage operator clearly outperforms alternative operations, e.g., truncating singular values below some threshold, in that it finds matrices with correct rank quite often, but that the distance to the subspace $\mathcal{M}$ is too large. This is reasonable as the only fixed points of soft thresholding operator are either zero, or, when combined with normalization, matrices with identical nonzero singular values, e.g., rank-one matrices.

As we will treat the subspace constraint as non-negotiable, we will need further improvement. We replace the shrinkage operator in the second phase by best approximation of the estimated

rank (which we call *hard thresholding*). Combined with the projection onto the admissible set $\mathcal{M}$, this then delivers low-rank matrices in the subspace $\mathcal{M}$ astonishingly reliably (as we shall see, this second phase is typically not needed when a rank-one basis exists).

Our convergence analysis in Section 8.4 provides further insights into the behavior of the process, in particular the second phase.

8.1.2 Applications

We were originally motivated to consider the low-rank basis problem by applications in discrete mathematics [78], [88], [117].

The Rank-One Basis Problem

An interesting and important subcase of the low-rank basis problem is the *rank-one basis problem*; in this problem, we are further promised that a given subspace $\mathcal{M}$ is spanned by rank-one matrices. The rank-one basis problem has been studied by several authors [78], [88] in the area of theoretical computer science and discrete mathematics.

Their motivation comes from the following problem, which they call the *Edmonds' problem*; given a linear subspace $\mathcal{M}$ of $m \times n$ matrices, determine whether $\mathcal{A}$ contains a nonsingular matrix. This problem is a decision problem version of max-rank matrix completion for matrices without constant parts. A slightly generalized problem is to find an element of maximum rank in $\mathcal{M}$. As we have seen in Chapter 6, if the underlying field is sufficiently large, then Lovász's randomized algorithm solves the Edmonds' problem. Also, deterministic algorithms are known for special classes of matrix spaces, for example, matrix spaces spanned by rank-one matrices [117] or rank-two skew-symmetric matrices [152]. In what follows, we call a basis of $\mathcal{M}$ consisting of rank-one matrices a *rank-one basis*.

Gurvits [78] considered a more general setting, in which we know in advance that $\mathcal{M}$ has a rank-one basis, but are not given the basis elements explicitly. Instead, we are given matrices $M_1, \ldots, M_d$ as a basis of $\mathcal{M}$ such that each M_k can be represented as a linear combination of unknown rank-one matrices $X_\ell = \mathbf{a}_\ell \mathbf{b}_\ell^\top$ for $\ell = 1, \ldots, d$. If we can solve the rank-one basis problem, then this settings can be reduced into the setting of [117]. However, Gurvits [78] conjectured that rank-one basis problem is NP-hard in general, and he presented a deterministic algorithm for such matrix spaces over $\mathbb{R}$ without such reduction. His algorithm was extended for any finite field $\mathbb{F}$ with $|\mathbb{F}| > n$ in the very recent paper of Ivanyos et al [88].

Nevertheless, the task might be practically feasible within some numerical tolerance via nonlinear optimization. For example, the rank-one basis problem has an interesting connection to tensor decomposition: finding a rank-one basis for a d-dimensional matrix subspace amounts to finding a CP decomposition (e.g., [101, Sec. 3]) of representation rank d for the third-order tensor with slices M_k. For the latter task very efficient nonconvex optimization algorithm like alternating least squares exist, which, however, typically come without any convergence certificates. An alternative, less cheap, but exact method uses simultaneous diagonalization, which are applicable when $d \leq \min(m, n)$. Applying these methods will often be successful when a rank-one basis exists, but fails if not. This tensor approach seems to have been overseen in the discrete optimization community so far, and we explain it in Section 8.6.

Compression of SVD matrices

Low-rank matrices arise frequently in applications and a low-rank (approximate) decomposition such as the SVD is often used to reduce the storage to represent the matrix $A \in \mathbb{R}^{m \times n}$: $A \approx U \Sigma V^\top$. Here $\Sigma \in \mathbb{R}^{r \times r}$ where r is the rank. The memory requirement for storing the whole

A is clearly mn, whereas U, Σ, V altogether require $(m+n)r$ memory (we can dismiss Σ by merging it into V). Hence, the storage reduction factor is

$$\frac{(m+n)r}{mn}, \tag{8.3}$$

so if the rank r is much smaller than $\min(m,n)$ then we achieve significant memory savings.

This is all well known, but here we go one step further and try to reduce the memory cost for representing the matrices U, V. Note that the same idea of using a low-rank representation is useless here, as these matrices have orthonormal columns and hence the singular values are all ones.

The idea is the following: if we *matricize* the columns of U (or V), those matrices might have a low-rank structure. More commonly, there might exist a nonsingular matrix $W \in \mathbb{R}^{r\times r}$ such that the columns of UW have low-rank structure when matricized. We shall see that many orthogonal matrices that arise in practice have this property. The question is, then, how do we find such W and the resulting compressed representation of U? This problem boils down to the low-rank basis problem, in which $\mathcal{M}$ is the linear subspace spanned by the matricized columns of U.

To simplify the discussion here we assume $m = s^2$ for an integer s (otherwise, e.g. when m is prime, a remedy is to pad zeros to the bottom). Once we find an appropriate W for $U \in \mathbb{R}^{s^2\times r}$, we represent the matricization of each column as a low-rank (rank $\hat{r}$) matrix $U_{\hat{r}}\hat{\Sigma}V_{\hat{r}}$, which is represented using $2s\hat{r}$ memory, totalling to $2sr\hat{r} + r^2$ where r^2 is for W. Since the original U requires $s^2 r$ memory with $r \ll s^2$, this can significantly reduce the storage if $\hat{r} \ll r$.

When this is employed for both U and V the overall storage reduction for representing A becomes

$$\frac{4sr\hat{r} + r^2}{mn}. \tag{8.4}$$

For example, when $m = n$, $r = \delta m$ and $\hat{r} = \hat{\delta}s$ for $\delta, \delta \ll 1$ this factor is

$$4\delta\hat{\delta} + \delta^2, \tag{8.5}$$

achieving a "squared" reduction factor compared with (8.3), which is about 2δ.

Of course, we can further reduce the memory by recursively matricizing the columns of $U_{\hat{r}}$, as long as it results in data compression.

Computing and Compressing an Exact Eigenvector of a Multiple Eigenvalue

Eigenvectors of a multiple eigenvalue are not unique, and those corresponding to near-multiple eigenvalues generally cannot be computed to high accuracy. We shall show that it is nonetheless sometimes possible to compute exact eigenvectors of near-multiple eigenvalues, if additional property is present that the eigenvectors are low-rank when matricized. This comes with the additional benefit of storage reduction, as discussed above. We describe more details and experiments in Section 8.5.4.

8.2 The Abstract Greedy Algorithm for the Low-Rank Basis Problem

As already mentioned, the low-rank basis problem (8.1) for a matrix subspace $\mathcal{M}$ is a generalization of the sparse basis problem for subspaces of $\mathbb{R}^n$. In [49] it was shown that a solution to the sparse basis problem can be in principle found using a greedy strategy. The same is true

Algorithm 8.1 Greedy meta-algorithm for computing a low-rank basis

Input: Subspace $\mathcal{M} \subseteq \mathbb{R}^{m\times n}$ of dimension d.
Output: Basis $\mathcal{B} = \{X_1^*, \dots, X_d^*\}$ of $\mathcal{M}$.
1: Initialize $\mathcal{B} = \emptyset$.
2: **for** $\ell = 1, \dots, d$:
3: Find $X_\ell^* \in \mathcal{M}$ of lowest possible rank such that $\mathcal{B} \cup \{X_\ell^*\}$ is linearly independent.
4: $\mathcal{B} \leftarrow \mathcal{B} \cup \{X_\ell^*\}$

for (8.1), as we will show next. The corresponding greedy algorithm is given as Algorithm 8.1. Indeed, this algorithm can be understood as a greedy algorithm for an *infinite matroid* [135] of finite rank. We can prove that Algorithm 8.1 finds a minimizer of (8.1), by adapting a standard proof for greedy algorithms on *finite* matroids. Note that this fact does not imply that (8.1) is tractable, since finding X_ℓ^* in the algorithm is a nontrivial task.

Lemma 8.2.1. *Let $X_1^*, \dots, X_d^*$ denote matrices constructed by the greedy Algorithm 8.1. Then for any $1 \le \ell \le d$ and linearly independent set $\{X_1, \dots, X_\ell\} \subseteq \mathcal{M}$ with $\operatorname{rank}(X_1) \le \cdots \le \operatorname{rank}(X_\ell)$, it holds*

$$\operatorname{rank}(X_i^*) \le \operatorname{rank}(X_i) \quad \text{for } i = 1, \dots, \ell.$$

Proof. The proof is by induction. By the greedy property, X_1^* is a (nonzero) matrix of minimal possible rank in $\mathcal{M}$, i.e., $\operatorname{rank}(X_1^*) \le \operatorname{rank}(X_1)$. For $\ell > 1$, if $\operatorname{rank}(X_\ell) < \operatorname{rank}(X_\ell^*)$, then $\operatorname{rank}(X_i) < \operatorname{rank}(X_\ell^*)$ for all $i = 1, \dots, \ell$. But since one X_i must be linearly independent from $X_1^*, \dots, X_{\ell-1}^*$, this would contradict the choice of X_ℓ^* in the greedy algorithm. □

We say a linearly independent set $\mathcal{B}_\ell = \{\hat{X}_1, \dots, \hat{X}_\ell\} \subseteq \mathcal{M}$ is of *minimal rank*, if

$$\sum_{i=1}^{\ell} \operatorname{rank}(\hat{X}_i) = \min \left\{ \sum_{i=1}^{\ell} \operatorname{rank}(X_i) : \{X_1, \dots, X_\ell\} \subseteq \mathcal{M} \text{ is linearly independent} \right\}.$$

The following theorem is immediate from the previous lemma.

Theorem 8.2.2. *Let $X_1^*, \dots, X_d^*$ denote matrices constructed by the greedy Algorithm 8.1, and let $\mathcal{B}_\ell = \{X_1, \dots, X_\ell\} \subseteq \mathcal{M}$ be a linearly independent set with $\operatorname{rank}(X_1) \le \cdots \le \operatorname{rank}(X_\ell)$. Then $\mathcal{B}_\ell$ is of minimal rank if (and hence only if)*

$$\operatorname{rank}(X_i) = \operatorname{rank}(X_i^*) \quad \text{for } i = 1, \dots, \ell.$$

In particular, $\{X_1^*, \dots, X_\ell^*\}$ is of minimal rank.

A remarkable corollary is that the ranks of the elements in a basis of lowest rank are essentially unique.

Corollary 8.2.3. *The output $\mathcal{B} = \{X_1^*, \dots, X_d^*\}$ of Algorithm 8.1 solves the low-rank basis problem (8.1), that is, provides a basis for $\mathcal{M}$ of lowest possible rank. Any other basis of lowest rank takes the same ranks $\operatorname{rank}(X_\ell^*)$ up to permutation.*

8.3 Finding Low-Rank Bases via Thresholding and Projection

In this main section of this article, we propose an algorithm that tries to execute the abstract greedy Algorithm 8.1 using iterative methods on relaxed formulations.

The greedy algorithm suggests finding the matrices $X_1, \ldots, X_d$ one after another, during which we monitor the linear dependence when computing X_ℓ with respect to the previously computed $X_1, \ldots, X_{\ell-1}$, and apply some restart procedure when necessary. Alternatively, one can try to find low-rank matrices $X_1, \ldots, X_d \in \mathcal{M}$ in parallel, monitor their linear independence, and reinitialize the ones with largest current ranks in case the basis becomes close to linearly dependent. In both cases, the linear independence constraint, which substantially increases the hardness of the problem, is in principle ignored as long as possible, and shifted into a restart procedure. Therefore, we mainly focus on iterative methods to solve the problem (8.2) of finding a single low-rank matrix X in $\mathcal{M}$. The complete procedure for the low-rank basis problem will be given afterwards in Section 8.3.3.

The first algorithm we consider for solving the low-rank basis problem (8.2) alternates between soft singular value thresholding (shrinkage) and projection onto the subspace $\mathcal{M}$, and will be presented in the next subsection. We note that an analogous method for the corresponding sparse vector problem of minimizing $\|\boldsymbol{x}\|_0$ over $\boldsymbol{x} \in S$, $\|\boldsymbol{x}\|_2 = 1$ has been independently derived in the very recent publication [138]. While our method shares some basic ideas with their method, but we will introduce several new ideas such as rank estimation and hard thresholding to deal with matrices rather than vectors.

8.3.1 Phase I: Estimating a Single Low-Rank Matrix via Soft Thresholding

The starting point is a further relaxation of (8.2): the rank function, that is, the number of nonzero singular values, is replaced by the matrix nuclear norm $\|X\|_*$, which equals the sum of singular values. This leads to the problem

$$\begin{array}{ll} \text{minimize} & \|X\|_* \\ \text{subject to} & X \in \mathcal{M}, \text{ and } \|X\|_F = 1. \end{array} \tag{8.6}$$

The relaxation from rank to nuclear norm can be motivated by the fact that in case a rank-one solution exists, it will certainly be recovered by solving (8.6). For higher ranks, it is less clear under which circumstances the nuclear norm provides an exact surrogate for the rank function under the given spherical constraint. For an example of a space $\mathcal{M}$ spanned by matrices of rank at most r for which the minimum in (8.6) is attained at a matrix of rank larger than r, consider $M_1 = \operatorname{diag}(1, -\sqrt{\epsilon}, -\sqrt{\epsilon}, 0, 0, 0, 0)$, $M_2 = \operatorname{diag}(0, 1, 0, \sqrt{\epsilon}, \sqrt{\epsilon}, 0, 0)$, and $M_3 = \operatorname{diag}(0, 0, 1, 0, 0, \sqrt{\epsilon}, \sqrt{\epsilon})$. Every linear combination involving at least two matrices has at least rank four. So the M_i are the only matrices in their span of rank at most three. After normalization w.r.t. the Frobenius norm, their nuclear norm equals $\|M_i\|_*/\|M_i\|_F = (1 + 2\sqrt{\epsilon})/\sqrt{1 + 2\epsilon} = 1 + 2\sqrt{\epsilon} + O(\epsilon)$. But for ϵ small enough, the rank five matrix $X = M_1 - \sqrt{\epsilon}M_2 - \sqrt{\epsilon}M_3 = (1, 0, 0, \epsilon, \epsilon, \epsilon, \epsilon)$ has a smaller nuclear norm $\|X\|_*/\|X\|_F = (1+4\epsilon)/\sqrt{1 + 4\epsilon^2} = 1 + 4\epsilon + O(\epsilon^2)$ after normalization.

Soft Thresholding and Block Coordinate Descent

Nevertheless, such counterexamples are rather contrived, and we consider (8.6) a good surrogate for (8.2) in the generic case. The problem is still very challenging due to the non-convex constraint. In [138] a block coordinate descent (BCD) method has been proposed to minimize the ℓ_1-norm of a vector on an Euclidean sphere in a subspace of $\mathbb{R}^n$. As we explained above, this problem is a special case of (8.6), and the algorithm can be generalized as follows.

Given a current guess $X \in \mathcal{M}$ we are looking for a matrix Y of lower-rank in a neighborhood if possible. For this task, we use the *singular value shrinkage operator* $\mathcal{S}_\tau$ (see Chapter 6, Sec 6.3). The rank of $Y = \mathcal{S}_\tau(X)$ now equals the number of singular values σ_i larger than τ. Note that even if the rank is not reduced the ratios of the singular values increase, since

$(\sigma_i - \tau)/(\sigma_j - \tau) > \sigma_i/\sigma_j$ for all (i, j) such that $\tau < \sigma_j < \sigma_i$. Hence a successive application of the shift operator will eventually remove all but the dominant singular value(s), even if the iterates are normalized in between (without in-between normalization it of course removes all singular values). This effect is not guaranteed when simply deleting singular values below the threshold τ without shifting the others, as it would preserve the ratios of the remaining singular values, and might result in no change at all if τ is too small. But even if this *hard threshold* was chosen such that at least one singular value is always removed, we found through experiments that this does not work as well in combination with projections onto a subspace $\mathcal{M}$ as the soft thresholding.

The new matrix $Y = \mathcal{S}_\tau(X)$ will typically not lie in $\mathcal{M}$, nor will it be normalized w.r.t. the Frobenius norm. Thus, introducing the orthogonal projector (w.r.t. the Frobenius inner product) $\mathcal{P}_\mathcal{M}$ from $\mathbb{R}^{m\times n}$ onto $\mathcal{M}$, which is available given *some* basis $M_1, \dots, M_d$ of $\mathcal{M}$, we consider the fixed point iteration:

$$Y = \mathcal{S}_\tau(X), \quad X = \frac{\mathcal{P}_\mathcal{M}(Y)}{\|\mathcal{P}_\mathcal{M}(Y)\|_F}. \tag{8.7}$$

The projection $\mathcal{P}_\mathcal{M}$ is precomputed at the beginning and defined as $\mathbf{M}\mathbf{M}^\top$ (only $\mathbf{M}$ is stored), where $\mathbf{M}$ is the orthogonal factor of the thin QR decomposition [73, Sec. 5] of the matrix $[\mathrm{vec}(M_1), \dots, \mathrm{vec}(M_d)] \in \mathbb{R}^{mn\times d}$, where the (not necessarily low-rank) matrices M_i span $\mathcal{M}$. It is used in the following way:

$$\mathcal{P}_\mathcal{M}(Y) = \mathrm{mat}(\mathbf{M}\mathbf{M}^\top \mathrm{vec}(Y)), \tag{8.8}$$

To obtain the interpretation as BCD, we recall the fact that the shrinkage operation provides the unique solution to the strictly convex problem

$$\underset{Y}{\text{minimize}} \quad \tau\|Y\|_*^2 + \frac{1}{2}\|Y - X\|_F^2, \tag{8.9}$$

see [30]. Intuitively, (8.9) attempts to solve (8.6) in a neighborhood of the current guess X, while ignoring the constraints. The parameter τ controls the balance between small nuclear norm and locality: the larger it is the lower $\mathrm{rank}(Y)$ becomes, but Y will be farther from X. Taking τ small has the opposite effect. Indeed, the distance between X and Y is equal to

$$\|X - Y\|_F^2 = \sum_{\sigma_i > \tau} \tau^2 + \sum_{\sigma_i \le \tau} \sigma_i^2.$$

As a result, we see that the formulas (8.7) represent the update rules when applying BCD to the problem

$$\begin{aligned} \underset{X,Y}{\text{minimize}} \quad & \tau\|Y\|_* + \frac{1}{2}\|Y - X\|_F^2 \\ \text{subject to} \quad & X \in \mathcal{M}, \text{ and } \|X\|_F = 1, \end{aligned}$$

which can be seen as a penalty approach to approximately solving (8.6).

Algorithm with Adaptive Shift τ

The considerations are summarized in the following Algorithm 8.2. We repeat that it is more or less analogous to the algorithm in [138]. However, a new feature is that the parameter τ is chosen adaptively in every iteration.

The choice of the singular value shift τ in line 3 is made to achieve faster progress, and motivated by the fact that a matrix X of Frobenius norm 1 has at least one singular value

Algorithm 8.2 Phase I – Rank estimation

Input: Orthogonal projection $\mathcal{P}_{\mathcal{M}}$ on $\mathcal{M}$; scalars $\delta, maxit, changeit > 0$; initial guess $X \in \mathcal{M}$, initialize $r = n$.

Output: X, Y, and r, where $X = \mathcal{P}_{\mathcal{M}}(Y) \in \mathcal{M}$, and Y is a matrix of low rank r which is close to or in $\mathcal{M}$.

1: **for** $it = 1, \ldots, maxit$:
2: $X = U\Sigma V^\top$ # singular value decomposition
3: $\tau = \delta/\sqrt{\operatorname{rank}(X)}$ # set singular value shift
4: $Y \leftarrow \mathcal{S}_\tau(X)$ # shrinkage
5: $r \leftarrow \min(r, \operatorname{rank}(Y))$ # new rank estimate
6: $X \leftarrow \mathcal{P}_{\mathcal{M}}(Y)$ # projection onto subspace
7: $X \leftarrow X/\|X\|_F$ # normalization
8: Terminate if r has not changed for *changeit* iterations.

below and one above $1/\sqrt{\operatorname{rank}(X)}$, unless all singular values are the same. Therefore, the choice of $\tau = 1/\sqrt{\operatorname{rank}(X)}$ would always remove at least one singular value, but can also remove all but the largest singular value in case the latter is very dominant. However, since the sought low-rank matrix may happen to have almost identical singular values, it is important to choose a less aggressive shift by multiplying with $0 < \delta < 1$. The value $\delta = 0.1$ worked well in our experiments. We should note that the value $\operatorname{rank}(X)$ is available at this point in the algorithm since an SVD of X has been computed in the previous step anyway. In fact, in practice we used the following slightly modified shift strategy: one first removes all singular values of X below some threshold regarded as noise to obtain $\tilde{X}$ of rank s, and then chooses $\tau = \delta\|\tilde{X}\|_F/\sqrt{s}$, to achieve faster progress.

Of course, one can think of similar adaptive strategies such as $\tau \sim \|X\|_*/\operatorname{rank}(X)$ or $\tau \sim \sigma_{\operatorname{rank}(X)}$. The choice $\tau = \sigma_2$ is equivalent to best rank-one approximation.

Choice of the initial guess Let us remark on the choice of the initial guess. As we shall see later and can be easily guessed, with randomly generated initial $X \in \mathcal{M}$, the output r is not always the rank of the lowest-rank matrix in $\mathcal{M}$. A simple way to improve the rank estimate is to repeat Phase I with several initial matrices, and adopt the one that results in the smallest rank. Another "good" initial guess seems to be the analogue of that suggested in [138], which in our context is to take $\boldsymbol{v}^\top$ to be a row vector of a matrix of the current subspace and take the initial guess to be $X = \mathcal{P}_{\mathcal{M}}\boldsymbol{v}$. A natural strategy is to take $\boldsymbol{v}^\top$ to be several rows with the largest norms. This worked fine but not evidently better than random initial guesses, and we therefore leave the issue of a good initial guess an open problem.

The Use as a Rank Estimator

In our experiments we observed that Algorithm 8.2 alone is often not capable of finding a low-rank matrix in the subspace. Typically the two subsequences for Y and X in (8.7) produce two different numerical limits: Y tends to a low-rank matrix which is close to, but not in the subspace $\mathcal{M}$; by contrast, the X are always in the subspace, but are typically not observed to converge to a low-rank matrix. In fact, we can only have $X = Y$ in (8.7) for rank-one matrices. Therefore, in the general case, further improvement will be necessary (phase II below). However, as it turns out, the rank found by the sequence of Y provides a surprisingly good estimate also for the sought minimal rank in the subspace. Moreover, the obtained X also provides the starting guess $X = \mathcal{P}_{\mathcal{M}}$ for further improvement in the second phase, described next. An analogous statement was proven in [138] for the sparse vector problem (which can be regarded as a special

case of ours), but the analysis there assumes the existence of a sparse vector in a subspace of otherwise random vectors; here we do not have such (or related) assumptions. In Section 8.4.1 we give some qualitative explanation for why we expect this process to obtain the correct rank, but we leave a detailed and rigorous analysis of Algorithm 8.2 an open problem and call this preliminary procedure "Rank estimation".

8.3.2 Phase II: Extracting Matrix of Estimated Rank via Alternating Projections and Hard Thresholding

We now turn to the second phase, in which we find the matrix $X \in \mathcal{M}$ such that $\operatorname{rank}(X) = r$, the output rank of Phase I.

Alternating Projections

In Phase II of our algorithm we assume that we know a rank r such that $\mathcal{M}$ contains a matrix of rank at most r. To find that matrix, we use the method of alternating projection between the Euclidean (Frobenius) unit sphere in the subspace $\mathcal{M}$ and the closed cone of matrices of rank at most r. The metric projection (in Frobenius norm) on this cone is given by the *singular value truncation operator* $\mathcal{T}_r$ defined as

$$\mathcal{T}_r(X) = U\mathcal{T}_r(\Sigma)V^\top, \quad \mathcal{T}_r(\Sigma) = \operatorname{diag}(\sigma_1, \dots, \sigma_r, 0, \dots, 0),$$

where $X = U\Sigma V^\top$ is an SVD of X with $\Sigma = \operatorname{diag}(\sigma_1, \dots, \sigma_{\min(m,n)})$. The method of alternating projections hence reads

$$Y = \mathcal{T}_r(X), \quad X = \frac{\mathcal{P}_{\mathcal{M}}(Y)}{\|\mathcal{P}_{\mathcal{M}}(Y)\|_F}. \tag{8.10}$$

Conceptually, this iteration is the same as (8.7) with the soft thresholding operator $\mathcal{S}_\tau$ replaced by the hard thresholding operator $\mathcal{T}_r$. Alternatively, (8.10) can be interpreted as employing BCD for the problem

$$\begin{aligned} \underset{X,Y}{\text{minimize}} \quad & \|Y - X\|_F \\ \text{subject to} \quad & X \in \mathcal{M},\ \|X\|_F = 1,\ \text{and}\ \operatorname{rank}(Y) \le r. \end{aligned}$$

As a result, we obtain Algorithm 8.3.

Algorithm 8.3 Phase II – Alternating projections

Output: Orthogonal projection $\mathcal{P}_{\mathcal{M}}$ on $\mathcal{M}$; rank $r \ge 1$; scalars $maxit, tol > 0$; initial guess $X \in \mathcal{M}$.

Input: X, Y, where $X = \mathcal{P}_{\mathcal{M}}(Y) \in \mathcal{M}$, and Y is a matrix of rank r, and hopefully $\|X - Y\|_F \le tol$.

1: **while** $\|X - Y\|_F > tol$ and $it \le maxit$:
2: $\quad X = U\Sigma V^\top$ # singular value decomposition
3: $\quad Y \leftarrow \mathcal{T}_r(X)$ # best rank-r approximation
4: $\quad X \leftarrow \mathcal{P}_{\mathcal{M}}(Y)$ # projection onto subspace
5: $\quad X \leftarrow X/\|X\|_F$ # normalization

The authors of the aforementioned reference [138], who proposed Algorithm 8.2 for the sparse vector problem, also suggest a second phase (called "rounding"), which is, however,

vastly different from our Algorithm 8.3. It is based on linear programming and its natural matrix analogue would be to solve

$$\begin{aligned} &\text{minimize} && \|X\|_* \\ &\text{subject to} && X \in \mathcal{M}, \text{ and } \operatorname{tr}(\tilde{X}^\top X) = 1. \end{aligned} \tag{8.11}$$

Here $\tilde{X}$ is the final matrix X from Algorithm 8.2. In [138] it is shown for the vector case that if $\tilde{X}$ is sufficiently close to a global solution of (8.6) then we can recover it exactly by solving (8.11).

Comparison with a Convex Optimization Solver

Note that unlike our original problem (8.1) or its nuclear norm relaxation (8.6), (8.11) is a convex optimization problem, since the constraints are now the linear constraint $\operatorname{tr}(\tilde{X}^\top X) = 1$ along with the restriction in the subspace $X \in \mathcal{M}$. Nuclear norm minimization under linear constraints has been intensively considered in the literature, see [30], [140] and references therein for seminal work. A natural approach is to attempt to solve (8.11) by some convex optimization solver.

In view of this, we conduct experiments to compare our algorithm with one where Phase II is replaced by the cvx optimization code [75]. For the test we used the default tolerance parameters in cvx. We vary n while taking $m = n$ and generated the matrix subspace $\mathcal{M}$ so that the exact ranks are all equal to five. Here and throughout, the numerical experiments were carried out in MATLAB version R2014a on a desktop machine with an Intel Core i7 processor and 16GB RAM.

The runtime and accuracy are shown in Table 8.1. Here the accuracy is measured as follows: letting $\hat{X}_i$ for $i = 1, \ldots, d$ be the computed rank-1 matrices, we form the $mn \times d$ matrix $\hat{\mathcal{X}} = [\operatorname{vec}(\hat{X}_1), \ldots, \operatorname{vec}(\hat{X}_d)]$, and compute the error as the subspace angle [73, Sec. 6.4.3] between $\hat{\mathcal{X}}$ and the exact subspace $[\operatorname{vec}(M_1), \ldots, \operatorname{vec}(M_d)]$. Observe that while both algorithms provide (approximate) desired solutions, Algorithm 8.5 is more accurate, and much faster with the difference in speed increasing rapidly with the matrix size.

Table 8.1: Comparison between Alg. 8.5 and Phase II replaced with cvx.

(a) Runtime(s)

n	20	30	40	50
Alg. 8.5	1.4	2.21	3.38	4.97
`cvx`	28.2	186	866	2960

(b) Error

n	20	30	40	50
Alg. 8.5	2.9e-15	8.0e-15	9.5e-15	2.1e-14
`cvx`	6.4e-09	5.2e-10	5.7e-10	2.8e-10

Another approach to solving (8.11) is Uzawa's algorithm as described in [30]. However, our experiments suggest that Uzawa's algorithm gives poor accuracy (in the order of magnitude 10^{-4}), especially when the rank is not one.

In view of these observations, in what follows we do not consider a general-purpose solver for convex optimization and focus on using Algorithm 8.3 for Phase II.

8.3.3 A Greedy Algorithm for the Low-Rank Basis Problem

Restart for Linear Independence

Algorithms 8.2 and 8.3 combined often finds a low-rank matrix in $\mathcal{M}$. To mimic the abstract greedy Algorithm 8.1, this can now be done consecutively for $\ell = 1, \dots, d$. However, to ensure linear independence among the computed matrices X_i, a restart procedure may be necessary. After having calculated $X_1, \dots, X_{\ell-1}$ and ensured that they are linearly independent, the orthogonal projector $\mathcal{P}_{\ell-1}$ onto $\operatorname{span}\{X_1, \dots, X_{\ell-1}\}$ is calculated. While searching for X_ℓ the norm of $X_\ell - \mathcal{P}_{\ell-1}(X_\ell)$ is monitored. If it becomes too small, it indicates (since X_ℓ is normalized) that X_ℓ is close to linearly dependent on the previously calculated matrices $X_1, \dots, X_{\ell-1}$. The process for X_ℓ is then randomly restarted in the orthogonal complement of $\operatorname{span}\{X_1, \dots, X_{\ell-1}\}$ *within* $\mathcal{M}$, which is the range of $\mathcal{Q}_{\ell-1} = \mathcal{P}_{\mathcal{M}} - \mathcal{P}_{\ell-1}$.

Algorithm 8.4 Restart for linear independence

Input: Orthogonal projection $\mathcal{Q}_{\ell-1}$, matrix X_ℓ and tolerance $restarttol > 0$.
Output: Eventually replaced X_ℓ.

1: **if** $\|\mathcal{Q}_{\ell-1}(X_\ell)\|_F < restarttol$:
2: Replace X_ℓ by a random element in the range of $\mathcal{Q}_{\ell-1}$.
3: $X_\ell \leftarrow X_\ell / \|X_\ell\|_F$

In our implementation, we do not apply a random matrix to $\mathcal{Q}_{\ell-1}$ to obtain a random element in the range. Instead $\mathcal{Q}_{\ell-1}$ is stored in form of an orthonormal basis for which random coefficients are computed.

Final Algorithm and Discussion

The algorithm we propose for the low-rank basis problem is outlined as Algorithm 8.5. Some remarks are in order.

Algorithm 8.5 Greedy algorithm for computing a low-rank basis

Input: Basis $M_1, \dots M_d \in \mathbb{R}^{m \times n}$ for $\mathcal{M}$, and integer $restartit > 0$.
Output: Low-rank basis $X_1, \dots, X_d$ of $\mathcal{M}$.

1: Assemble the projection $\mathcal{P}_{\mathcal{M}}$.
2: Set $\mathcal{Q}_0 = \mathcal{P}_{\mathcal{M}}$.
3: **for** $\ell = 1, \dots, d$:
4: Initialize normalized X_ℓ randomly in the range of $\mathcal{Q}_{\ell-1}$.
5: Run Algorithm 8.2 (Phase I) on X_ℓ, but every $restartit$ iterations run Algorithm 8.4.
6: Run Algorithm 8.3 (Phase II) on X_ℓ, but every $restartit$ iterations run Algorithm 8.4.
7: Assemble the projection $\mathcal{P}_\ell$ on $\operatorname{span}\{X_1, \dots, X_\ell\}$ (ensure linear independence),
8: Set $\mathcal{Q}_\ell = \mathcal{P}_{\mathcal{M}} - \mathcal{P}_\ell$.

1. The algorithm is not stated very precisely as the choice of many input parameters are not specified. We will specify at the beginning of Section 8.5 the values we used for numerical experiments.

2. Analogously to (8.8), the projections $\mathcal{P}_\ell$ are in practice obtained from a QR decomposition of $[\operatorname{vec}(X_1), \dots, \operatorname{vec}(X_\ell)]$.

3. Of course, after Algorithm 8.2 one can or should check whether it is necessary to run Algorithm 8.3 at all. Recall that Algorithm 8.2 provides a rank-estimate r and a new

matrix $X_\ell \in \mathcal{M}$. There is no need to run Algorithm 8.3 in case $\operatorname{rank}(X_\ell) = r$ at that point. However, we observed that this seems to happen only when $r = 1$ (see next subsection and Section 8.5.1), so in practice it is enough to check whether $r = 1$.

4. There is a principal difference between the greedy Algorithm 8.5, and the theoretical Algorithm 8.1 regarding the monotonicity of the found ranks. For instance, there is no guarantee that the first matrix X_1 found by Algorithm 8.5 will be of lowest possible rank in $\mathcal{M}$, and it will often not happen. It seems to depend on the starting value (recall the discussion on the initial guess in Section 8.3.1), and an explanation may be that the soft thresholding procedure gets attracted by "some nearest" low-rank matrix, although a rigorous argument remains an open problem. In any case, finding a matrix of lowest rank is NP-hard as we have already explained in the introduction. In conclusion, one should hence not be surprised if the algorithm produces linearly independent matrices $X_1, \ldots, X_d$ for which the sequence $\operatorname{rank}(X_\ell)$ is not monotonically increasing. Nevertheless, at least in synthetic test, where the exact lowest-rank basis is exactly known and not too badly conditioned, the correct ranks are often recovered, albeit in a wrong order; see Figures 8.1, 8.3 and Section 8.5.1.

5. It is interesting to note that in case the restart procedure is not activated in any of the loops (that is, the if-clause in Algorithm 8.4 always fails), one would have been able to find the matrices $X_1, \ldots, X_d$ independently of each other, e.g., with a random orthogonal basis as a starting guess. In practice, we rarely observed that restart can be omitted, although it might still be a valuable idea to run the processes independently of each other, and monitor the linear independence of $X_1, \ldots, X_d$ as a whole. If some of the X_ℓ start converging towards the same matrix, or become close to linearly dependent, a modified restart procedure will be required. A practical way for such a simultaneous restart is to use a QR decomposition with pivoting. It will provide a triangular matrix with decreasing diagonal entries, with too small entries indicating basis elements that should be restarted. Elements corresponding to sufficiently large diagonal elements in the QR decomposition can be kept. We initially tried this kind of method, an advantage being that the overall cost for restarts is typically lower. We found that it typically works well when all basis elements have the same rank. However, the method performed very poorly in situations where the ranks of the low-rank basis significantly differ from each other. Nonetheless, this "parallel" strategy might be worth a second look in future work.

8.3.4 Complexity and Typical Convergence Plot

The main step in both Algorithms 8.2 and 8.3 is the computation of an SVD of an $m \times n$ matrix, which costs about $14mn^2 + 8n^3$ flops [73, Sec. 8.6]. This is done at most $2maxit$ times (assuming the same in both algorithms) for d matrices. Since we check the need for restarting only infrequently, the cost there is marginal. The overall worst-case complexity of our greedy Algorithm 8.5 hence depends on the choice of $maxit$. In most of our experiments, the inner loops in Algorithms 8.2 and 8.3 terminated according to the stopping criteria, long before $maxit$ iterations was reached.

Figure 8.1 illustrates the typical convergence behavior of Algorithm 8.5 for a problem with $m = 20$, $n = 10$, and $d = 5$, and the exact ranks for a basis are $(1, 2, 3, 4, 5)$. The colors correspond to $\ell = 1, \ldots, 5$. For each color, the evolution of the $n = 10$ singular values of X_ℓ are plotted during the iteration (always after projection on $\mathcal{M}$). The shaded areas show Phase I, the unshaded areas Phase II. In both phases we used $maxit = 1000$ and $restartit = 50$, moreover, Phase I was terminated when the rank did not change for $changeit = 50$ iterations

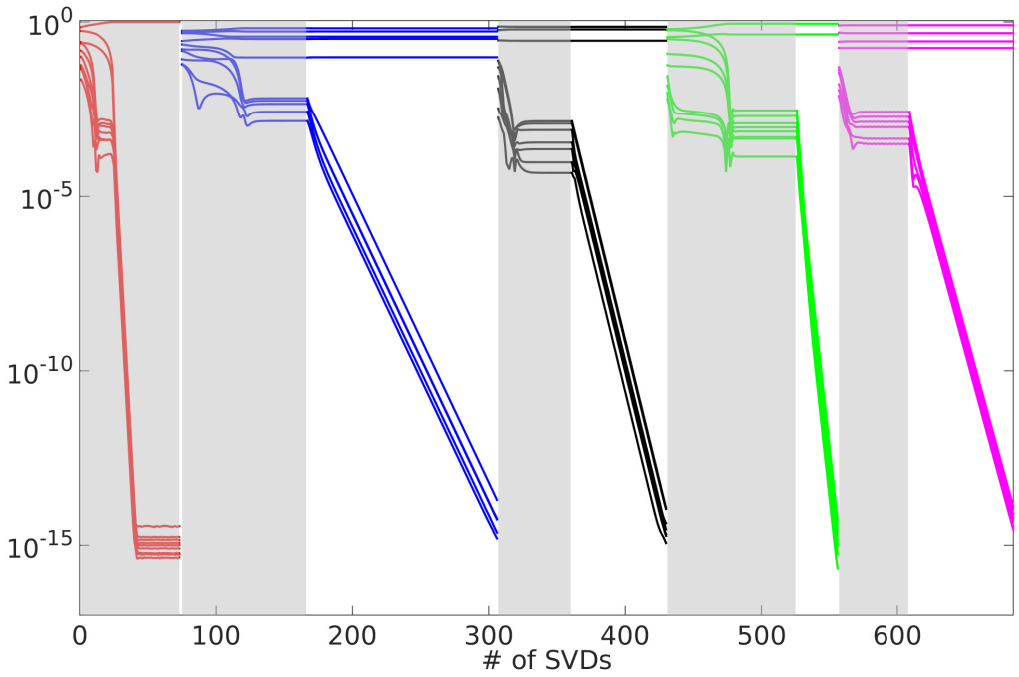


Figure 8.1: Typical convergence of Algorithm 8.5 for a randomly generated problem with $m = 20, n = 10, d = 5$. Each color represents one outmost loop $\ell = 1, \ldots, 5$. The shaded regions indicate Phase I, and the rest is Phase II. The exact ranks are $(1, 2, 3, 4, 5)$, but recovered in the order $(1, 5, 3, 2, 4)$.

(which appears to be still conservative), while Phase II was terminated when X_ℓ remained unchanged up to a tolerance of 10^{-14} in Frobenius norm. The number of SVDs is in principle equal to the number of single iterations, and governs the complexity, so it was used for the x-axis. Illustrating the fourth remark above, the ranks are not recovered in increasing order, but in the order $(1, 5, 3, 2, 4)$ (corresponding to the number of curves per color not converging to zero). Repeated experiments suggest that all orderings of rank recovery is possible.

Regarding the convergence of a single matrix, we make two observations in Figure 8.1. First, one can nicely see that in Phase I the singular values beyond σ_{r_ℓ} usually decrease until they stagnate at a certain plateau. The length of this plateau corresponds to the 50 iterations we have waited until accepting an unchanged rank (before projection) as a correct guess. Except for the first (red) curve, which shows convergence towards a rank-one basis element, the error level of this plateau is too high to count as a low-rank matrix. Put differently, the (normalized) low-rank matrix Y_ℓ found by the shrinkage, on which the rank guess is based, is unacceptably far away from the subspace, illustrating the need for Phase II. Phase II seems unnecessary only when a rank-one matrix is targeted.

Second, the convergence of $\sigma_{r+1}, \ldots, \sigma_n$ towards zero in the second phase is typically linear, and tends to be faster if the limit is lower in rank. We give an explanation for this observation in Section 8.4.

The fact that the matrices are obtained in somewhat random order can be problematic in some cases, such as the low-rank matrix problem where only one matrix of lowest rank is sought. One remedy is to try multiple initial guesses for Phase I, and adopt the one that results in the lowest rank estimate r. Figure 8.2 is a typical illustration when three initial guesses are attempted. The shaded regions represent Phase I repeated three times for each greedy step,

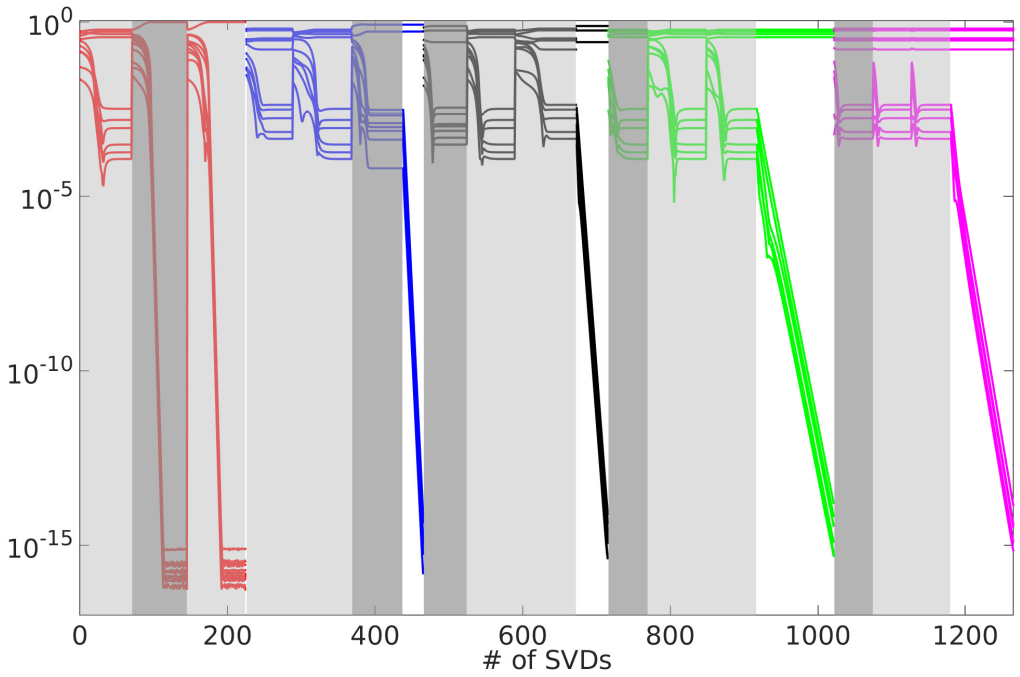


Figure 8.2: Same settings as Figure 8.1, but with three random initial guesses. Darkly shaded region represent the initial guess that was adopted. For instance, for the first matrix (red curve) the rank was estimated to be four in the first run of Phase I, and one in the second and third. The second was adopted, Phase II was not necessary. The final rank order is the correct $(1, 2, 3, 4, 5)$.

and the ones that were adopted are shaded darkly. Observe that now the matrices are obtained in the correct rank order $(1, 2, 3, 4, 5)$.

Finally, Figure 8.3 contains an outcome of the initial experiment (without multiple initial guesses) with square matrices of size $m = n = 20$. The ranks are recovered in another ordering, namely $(5, 1, 2, 3, 4)$. One can see that the ratio of the number of iterations in Phases I and II is quite different, and the overall number of required SVDs is smaller. The convergence analysis in Section 8.4, which suggests a typical convergence factor $\sqrt{\frac{r}{n}}$, gives a partial explanation. The main reason we give this third plot is the following interesting fact: the maximum rank a matrix in the generated subspace can have is $1 + 2 + 3 + 4 + 5 = 15$. Consequently, there seems to be a principal difference here from the previous example in that the subspace does not contain a full-rank matrix. This is perhaps part of why this problem seems somewhat easier in terms of the total number of iterations. Indeed, a close inspection of Figure 8.3 reveals that for every color we have five singular values below machine precision (recall that the plot shows the singular values after projection on the subspace).

8.4 Convergence Analysis

Given the hardness of the low-rank basis problem, the formulation of conditions for success or failure of Algorithm 8.5 must be a challenging task. Not to mention the discontinuous nature of the restart procedure, a satisfying rigorous and global convergence result remains a potentially interesting problem for future work.

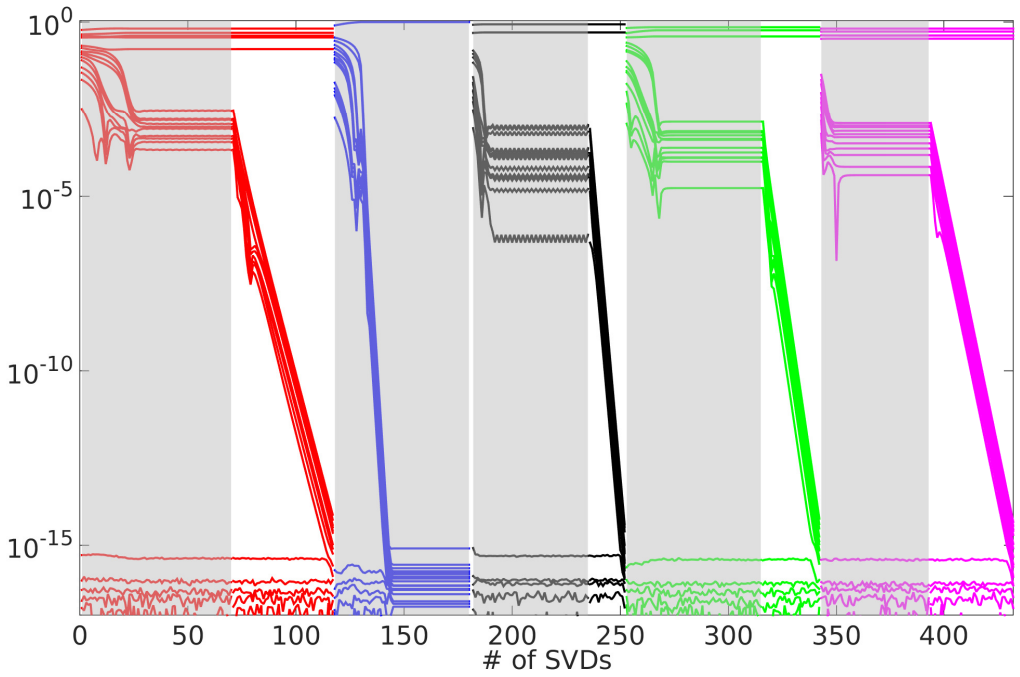


Figure 8.3: $m = n = 20$. All matrices in the subspace are rank-deficient, and we observe that the number or SVDs is fewer.

Here we confine ourselves to a local convergence analysis of Phase II for a single matrix given a correct rank estimate r, which is an alternating projection method. We will consider the simplest case where at the point of interest the tangent space of the manifold of rank-r matrices intersects trivially with the tangent space of the sphere in $\mathcal{M}$, which is arguably the simplest possible assumption when it comes to local convergence of the alternating projection method between smooth manifolds.

Regarding Phase I, we can at least note that if $X \in \mathcal{M}$ is a normalized rank-one matrix, then in a neighborhood of X a single step of Algorithms 8.2 and 8.3, respectively, will give the same result (this is also true for some similar choices of τ discussed above). Under the same assumptions as for Phase II this hence shows the local convergence of Phase I toward isolated normalized rank-one matrices in $\mathcal{M}$, see Corollary 8.4.2 below. Since this is a local convergence analysis, it of course does not fully explain the strong global performance of both Algorithms 8.2 and 8.3 in the rank-one case as seen in Figures 8.1– 8.3 and Tables 8.2 and 8.3.

In general, using the shrinkage operator cannot be guaranteed to converge to a local solution. We have already noted that a matrix of rank larger than two cannot be a fixed point of the shrinkage operator (unless all nonzero singular values are the same). One could construct examples where in a fixed point of (8.7) X has the same rank as Y, but generically this seems extremely unlikely. Therefore, the convergence of Phase I is in general a less relevant question. The main problem is in which situations it provides a correct rank estimate, at least locally. Except for the rank-one case we have no precise arguments, but we give a qualitative explanation at the end of Section 8.4.1.

8.4.1 Local Convergence of Phase II

Algorithm 8.3 is nothing else than the method of alternating projections for finding a matrix in the intersection $\mathcal{B} \cap \mathcal{R}_r$ of the closed sets

$$\mathcal{B} = \{X \in \mathcal{M} : \|X\|_F = 1\}$$

and

$$\mathcal{R}_r = \{X \in \mathbb{R}^{m\times n} : \operatorname{rank}(X) \leq r\}.$$

The understanding of local convergence of this method for nonconvex closed sets has made substantial progress during the last years [8], [109], [110], [132]. In these papers one finds very abstract result in the language of variational analysis or differential geometry. However, it can happen for concrete problems, that the validation of the requirements is not much less challenging than reproving the result with the concrete information at hand. For instance, the statement of Theorem 8.4.1 below essentially follows from [109, Theorem 5.16], but we believe the theorem does make a contribution by providing a direct proof based on elementary linear algebra, and leading to an estimate for the typical convergence factor. In fact, projection on low-rank matrices has been used frequently as a motivating example for alternating projections (often in company with strictly affine constraints) [8], [109], [110], [118], but for the local convergence very abstract general theorems like the one cited are invoked, whose fully general proof is much longer than possibly necessary in this setting. In Remark 8.4.3 we discuss the main assumption (8.13) in the context of the available literature in a bit more detail.

To state the result, we assume that $X_* \in \mathcal{T}_r \cap \mathcal{B}$ has exactly rank r. By semi-continuity of rank, all matrices in a neighborhood (in $\mathbb{R}^{m\times n}$) of X have rank at least r. Therefore, we can locally regard Algorithm 8.3 as an alternating projection between $\mathcal{B}$ and the smooth manifold of matrices of fixed rank r. Letting $X_* = U_*\Sigma_*V_*^\top$ be a thin SVD of X_* where Σ_* contains positive singular values, the tangent space to that manifold at X_* is [83]

$$T_{X_*}\mathcal{R}_r = \left\{ [U_*\ U_*^\perp] \begin{bmatrix} A & B \\ C & 0 \end{bmatrix} [V_*\ V_*^\perp] : A \in \mathbb{R}^{r\times r},\ B \in \mathbb{R}^{r\times(m-r)},\ C \in \mathbb{R}^{(n-r)\times r} \right\}. \tag{8.12}$$

For our analysis we make the following genericity assumption:

$$T_{X_*}\mathcal{R}_r \cap T_{X_*}\mathcal{B} = \{0\}. \tag{8.13}$$

Here $T_{X_*}\mathcal{B}$ is the tangent space of $\mathcal{B}$, which is the orthogonal complement of X_* within $\mathcal{M}$. Since $T_{X_*}\mathcal{R}_r$ contains X_*, (8.13) is expressed in terms of $\mathcal{M}$ as

$$T_{X_*}\mathcal{R}_r \cap \mathcal{M} = \operatorname{span}\{X_*\}. \tag{8.14}$$

We remark that (8.13) ultimately implies that X_* is an isolated point of $\mathcal{R}_r \cap \mathcal{B}$, so actually it then holds $\mathcal{R}_r \cap \mathcal{M} = \operatorname{span}\{X_*\}$. Then we have the following result.

Theorem 8.4.1. *Assume $X_* \in \mathcal{R}_r \cap \mathcal{B}$ has rank r, and that this rank is used in Algorithm 8.3. If (8.13) holds, then for $X \in \mathcal{B}$ close enough to X_* the new iterate $X_{new} = \frac{\mathcal{P}_{\mathcal{M}}(\mathcal{T}_r(X))}{\|\mathcal{P}_{\mathcal{M}}(\mathcal{R}_r(X))\|_F}$ constructed by the algorithm is uniquely defined, and*

$$\frac{\|X_{new} - X_*\|_F}{\|X - X_*\|_F} \leq \cos\theta + O(\|X - X_*\|_F^2),$$

where $\theta \in (0, \frac{\pi}{2}]$ is the subspace angle between $T_{X_*}\mathcal{B}$ and $T_{X_*}\mathcal{R}_r$, defined by

$$\cos\theta = \max_{\substack{X\in T_{X_*}\mathcal{B} \\ Y\in T_{X_*}\mathcal{R}_r}} \frac{|\langle X, Y\rangle_F|}{\|X\|_F\|Y\|_F}.$$

As a consequence, the iterates produced by Algorithm 8.3 are uniquely defined and converge to X_* at a linear rate for close enough starting values.

In accordance with our observations in Section 8.3.4, we also have a result for Phase I in the rank-one case.

Corollary 8.4.2. *If $r = 1$, the statement of Theorem 8.4.1 also holds for Algorithm 8.2. In fact, both algorithms produce the same iterates in some neighborhood of X_*.*

Here it is essential that the value of shift τ is bounded below by a fixed fraction of the Frobenius norm of X, as it is the case for in Algorithm 8.2, a lower bound being $\delta/\sqrt{\min(m,n)}$.

Proof of Theorem 8.4.1. Since X and X_* are in $\mathcal{B}$, we can write

$$X - X_* = E + O(\|X - X_*\|_F^2)$$

with $E \in T_{X_*}\mathcal{B}$. We partition the principal error E as

$$E = [U_* \; U_*^\perp] \begin{bmatrix} A & B \\ C & D \end{bmatrix} [V_* \; V_*^\perp]^\top. \tag{8.15}$$

Of course, $\|E\|_F^2 = \|A\|_F^2 + \|B\|_F^2 + \|C\|_F^2 + \|D\|_F^2$, and due to (8.12), our assumption implies

$$\|D\|_F^2 \geq \sin^2\theta \cdot \|E\|_F^2. \tag{8.16}$$

Since X_* has rank r, it follows that all matrices in some neighborhood have a unique best rank-r approximation (by perturbation arguments for the singular values). In this neighborhood X_{new} is uniquely defined. To relate $\|E\|$ to $\|\mathcal{T}_r(X) - X_*\|$ we consider the two matrices

$$F = \begin{bmatrix} I & C\Sigma_*^{-1} \\ -C\Sigma_*^{-1} & I \end{bmatrix} [U_* \; U_*^\perp]^\top,$$

and

$$G = [V_* \; V_*^\perp] \begin{bmatrix} I & -\Sigma_*^{-1}B \\ \Sigma_*^{-1}B & I \end{bmatrix},$$

both of which are orthogonal up to $O(\|E\|_F^2)$:

$$\|F^\top F - I\|_F = O(\|E\|_F^2), \quad \|G^\top G - I\|_F = O(\|E\|_F^2).^{1}$$

Therefore, denoting by $\tilde{F}, \tilde{G}$ the orthogonal polar factors of F, G, respectively, we also have[2]

$$F = \tilde{F} + O(\|E\|_F^2), \quad G = \tilde{G} + O(\|E\|_F^2). \tag{8.17}$$

One now verifies that

$$\tilde{F}X\tilde{G} = FXG + O(\|E\|_F^2) = \begin{bmatrix} \Sigma_* + A & 0 \\ 0 & D \end{bmatrix} + O(\|E\|_F^2),$$

[1] Here and in the following, the error constant behind $O(\|E\|_F)$ depends mainly on the condition of Σ_*, which can be very large, but is fixed in this type of local analysis.

[2] From a polar decomposition $Z^\top = UP$ one gets $Z^\top Z - I = (Z^\top - U)(P + I)U^\top$, and since the singular values of $(P + I)U^\top$ are all at least 1, it follows that $\|Z - U^\top\|_F \leq \|Z^\top Z - I\|_F$.

or, since $\tilde{F}$ and $\tilde{G}$ are orthogonal,

$$X = \tilde{F}^\top \begin{bmatrix} \Sigma_* + A & 0 \\ 0 & D \end{bmatrix} \tilde{G}^\top + O(\|E\|_F^2).$$

For E small enough, the best rank-r approximation of the principal part is obtained by deleting D. Hence,

$$\begin{aligned} \mathcal{T}_r(X) &= \mathcal{T}_r\left(\tilde{F}^\top \begin{bmatrix} \Sigma_* + A & 0 \\ 0 & D \end{bmatrix} \tilde{G}^\top\right) + O(\|E\|_F^2) \\ &= \tilde{F}^\top \begin{bmatrix} \Sigma_* + A & 0 \\ 0 & 0 \end{bmatrix} \tilde{G}^\top + O(\|E\|_F^2). \end{aligned} \tag{8.18}$$

To get the last equality we have used results from matrix perturbation theory [169] (see also [157, Sec.V.4]), which shows that under the perturbation $O(\|E\|_F^2)$, the singular subspace corresponding to the r largest singular values of X gets perturbed by $O(\frac{\|E\|_2^2}{gap})$ where *gap* is the smallest distance between the singular values of $\Sigma_* + A$ and those of D. For $\|E\|_F$ sufficiently small such that $\|E\|_F = o(\sigma_{\min}(\Sigma_*))$, this bound is $O(\|E\|_2^2)$. Together with the fact that the perturbation in the singular values is bounded also by $O(\|E\|_2^2)$ (since the condition number of singular values is always 1), we obtain the final equality above.

Therefore, taking also (8.17) into account, we obtain

$$\begin{aligned} \|\mathcal{T}_r(X) - X_*\|_F &= \|\tilde{F}\mathcal{T}_r(X)\tilde{G} - FX_*G\|_F + O(\|E\|_F^2) \\ &= \left\| \begin{bmatrix} A + \Sigma_* & 0 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} \Sigma_* & -B \\ -C & C\Sigma_*^{-1}B \end{bmatrix} \right\|_F + O(\|E\|_F^2) \\ &= \left\| \begin{bmatrix} A & B \\ C & 0 \end{bmatrix} \right\|_F + O(\|E\|_F^2). \end{aligned}$$

Here we used $O(\|C\Sigma_*^{-1}B\|_F^2) = O(\|E\|_F^2)$, which holds since $\|\Sigma_*^{-1}\|_2$ can be regarded as a constant that does not depend on $\|E\|_F$. Since $O(\|E\|) = O(\|X - X_*\|)$, we arrive at

$$\begin{aligned} \frac{\|\mathcal{T}_r(X) - X_*\|_F}{\|X - X_*\|_F} &= \frac{\sqrt{\|E\|_F^2 - \|D\|_F^2} + O(\|X - X_*\|_F^2)}{\|E\|_F + O(\|X - X_*\|_F^2)} \\ &\leq \sqrt{1 - \sin^2\theta} + O(\|X - X_*\|_F^2) \\ &= \cos\theta + O(\|X - X_*\|_F^2), \end{aligned} \tag{8.19}$$

where we have used (8.16). Since $X_* \in \mathcal{M}$, it now follows that

$$\|P_{\mathcal{M}}(\mathcal{T}_r(X)) - X_*\|_F \leq \|\mathcal{T}_r(X) - X_*\|_F \leq \cos\theta \|X - X_*\|_F + O(\|X - X_*\|_F^3). \tag{8.20}$$

Finally, we consider the normalization step. Recalling $\|X_*\|_F = 1$, by a simple geometric argument on the unit sphere we obtain

$$\left\| \frac{Y}{\|Y\|_F} - X_* \right\|_F \leq \frac{1}{\cos\phi} \|Y - X_*\|_F, \tag{8.21}$$

where $\phi \in [0, \frac{\pi}{2}]$ such that $\sin\phi = \|Y - X_*\|_F$. By Taylor expansion, $\frac{1}{\sqrt{1-\xi^2}} = 1 + O(\xi^2)$. Substituting $\xi = \sin\phi$ and $Y = \mathcal{P}_{\mathcal{M}}(\mathcal{T}_r(X))$ in (8.21) gives

$$\|X_{\text{new}} - X_*\|_F \leq \|P_{\mathcal{M}}(\mathcal{T}_r(X)) - X_*\|_F + O(\|P_{\mathcal{M}}(\mathcal{T}_r(X)) - X_*\|_F^3).$$

Using (8.20), we arrive at

$$\|X_{\text{new}} - X_*\|_F \leq \cos\theta \|X - X_*\|_F + O(\|X - X_*\|_F^3),$$

completing the proof. □

From Theorem 8.4.1 we can obtain a rough estimate for the convergence factor that we can expect to observe in practice. Consider the "generic" case where the error term E in (8.15) is randomly distributed, that is, each element is of comparable absolute value. Then we have $\|D\|_F^2 \approx \frac{(n-r)^2}{n^2}\|E\|_F^2$, and plugging this into (8.19) gives $\frac{\|\mathcal{T}_r(X)-X_*\|_F}{\|X-X_*\|_F} \leq \frac{\sqrt{2nr+r^2}}{n} + O(\|X - X_*\|_F^2)$. This suggests that we typically expect a convergence factor $\approx O(\sqrt{\frac{r}{n}})$. This estimate reflects the experiments quite well; see Section 8.5.2.

The above proof provides some insight into the behavior of Algorithm 8.2 in Phase I. In this case $\mathcal{T}_r(X)$ in (8.18) is replaced by $\mathcal{S}_\tau(X)$. Provided again that we start with a matrix X close to X_* so that $\|D\|_2 \leq \tau$, the operation $\mathcal{S}_\tau(X)$ again removes the D term, emphasizing the components towards X_* just like in Phase II as shown above. However, now the $\Sigma_* + A$ term is also affected, and thus Phase I stagnates where the thresholding effect in $\Sigma_* + A$ is balanced with the error terms that come in from the projection $\mathcal{P}_\mathcal{M}$. Then the rank estimate r is of correct rank $\operatorname{rank}(X_*)$, but neither X nor Y in Algorithm 8.2 is close to X_*; reflecting the remark at the end of Section 8.3.1.

Remark 8.4.3. The conditions (8.13), resp. (8.14), allow for a simple proof but of course impose some restrictions, most obviously $d = \dim(\mathcal{M}) \leq (m-r)(n-r) + 1$. In a seminal attempt, Lewis and Malick [110] obtained the local convergence of the method of alternating projections between two smooth submanifolds $\mathcal{M}$ and $\mathcal{N}$ of $\mathbb{R}^n$ towards some $X_* \in \mathcal{M} \cap \mathcal{N}$ under the condition that $T_{X_*}\mathcal{M} + T_{X_*}\mathcal{N} = \mathbb{R}^n$ (transversality). This allows for a non-trivial intersection, but imposes lower bounds on the dimensions, in our case $d \geq (m-r)(n-r) + 1$. Andersson and Carlsson [8] relaxed the condition to $T_{X_*}(\mathcal{M} \cap \mathcal{N}) = T_{X_*}\mathcal{M} \cap T_{X_*}\mathcal{N}$, however, under the assumption that $\mathcal{M} \cap \mathcal{N}$ is a C^2 manifold. This does not impose restriction on the dimensions, and contains (8.14) as a special cases. Still these conditions can fail in the situation at hand ($\mathcal{M}$ a subspace of $\mathcal{R}^{m\times n}$, $\mathcal{N} = \mathcal{R}_r$), for instance when $M = T_{X_*}\mathcal{R}_r$, to mention one counter-example. Recently, Noll and Rondepierre [132] proved local convergence for alternating projections under very weak but abstract assumptions, which do not involve tangential conditions in first place. It may be that their result applies to our setting, but we have not been able to validate this.

We conclude with a sufficient condition for (8.13) which might be useful in some very structured cases. We denote by $\operatorname{ran}(X)$ and $\operatorname{ran}(X^\top)$ the column and row space of a matrix X, respectively.

Lemma 8.4.4. *Let $X_* \in \mathcal{M}$ have rank r. Assume there exists a $(d-1)$-dimensional subspace $\tilde{\mathcal{M}} \subseteq \mathcal{M}$ complementary to $\operatorname{span}\{X_*\}$ with the following property: for every $\tilde{X} \in \tilde{\mathcal{M}}$ it holds $\operatorname{ran}(X_*) \cap \operatorname{ran}(\tilde{X}) = 0$ and $\operatorname{ran}(X_*^\top) \cap \operatorname{ran}(\tilde{X}^\top) = 0$. Then (8.13) holds.*

Proof. Consider $X = \alpha X_* + \beta\tilde{X} \in \mathcal{M}$ with $\tilde{X} \in \tilde{\mathcal{M}}$. Let P and Q denote the orthogonal projections on $\operatorname{ran}(X_*)^\perp$ and $\operatorname{ran}(X_*^\top)^\perp$, respectively. Then $X \in T_{X_*}\mathcal{R}_r$ if and only if

$$0 = PXQ^\top = \beta P\tilde{X}Q^\top.$$

It holds $P\tilde{X}Q^\top \neq 0$. To see this we note that $P\tilde{X}Q^\top = 0$ would mean $\operatorname{ran}(\tilde{X}Q^\top) \subseteq \operatorname{ran}(X_*)$, which by assumption implies $\tilde{X}Q^\top = 0$. But then $\operatorname{ran}(\tilde{X}^\top) \subseteq \operatorname{ran}(X_*^\top)$, a contradiction. Hence $X \in T_{X_*}\mathcal{R}_r$ if and only if $\beta = 0$, which proves the equivalent condition (8.14). □

8.5 Experiments

8.5.1 Synthetic Averaged Examples

We fix $m = n = 20$, $d = 5$, and a set of ranks $(r_1, \ldots, r_d)$. We then randomly generate d random matrices $M_1, \ldots, M_d$ of corresponding rank by forming random matrices $U_\ell \in \mathbb{R}^{m \times r_\ell}$ and $V_\ell \in \mathbb{R}^{n \times r_\ell}$ with orthonormal columns, obtained from the QR factorization of random Gaussian matrices using MATLAB's `randn` and setting $M_\ell = U_\ell V_\ell^\top$. To check the average rate of success of Algorithm 8.5, we run it 100 times and calculate

- the average sum of ranks $\sum_{\ell=1}^{d} \operatorname{rank}(Y_\ell)$ found by Phase I of the algorithm,
- the average truncation error $\left(\sum_{\ell=1}^{d} \|X_\ell - \mathcal{T}_{r_\ell}(X_\ell)\|_F^2\right)^{1/2}$ after Phase I,
- the average truncation error $\left(\sum_{\ell=1}^{d} \|X_\ell - \mathcal{T}_{r_\ell}(X_\ell)\|_F^2\right)^{1/2}$ after Phase II,
- the average iteration count (# of SVDs computed) in each Phase.

Table 8.2 shows the results for some specific choices of ranks. The principal parameters in the algorithm are $maxit = 1000$ and $restartit = 50$ (in both Phase I and Phase II). The maximum number of iterations was practically never reached. The rank guesses in Phase I were accepted after they remained unchanged for $changeit = 100$ loops. This is still very conservative. For comparison, recall that Figure 8.1 (ranks $(1, 2, 3, 4, 5)$) was generated with same parameter $maxit = 1000$, but $changeit = 50$. Phase II was terminated using $tol = 10^{-14}$, and never took 1000 iterations. From Table 8.2 we see that the ranks are sometimes estimated incorrectly, although this does not necessarily tarnish the final outcome.

Table 8.2: Synthetic results, random initial guess.

exact ranks	av. sum(ranks)	av. Phase I err (iter)	av. Phase II err (iter)
(1 , 1 , 1 , 1 , 1)	5.05	2.59e-14 (55.7)	7.03e-15 (0.4)
(2 , 2 , 2 , 2 , 2)	10.02	4.04e-03 (58.4)	1.04e-14 (9.11)
(1 , 2 , 3 , 4 , 5)	15.05	6.20e-03 (60.3)	1.38e-14 (15.8)
(5 , 5 , 5 , 10 , 10)	35.42	1.27e-02 (64.9)	9.37e-14 (50.1)
(5 , 5 , 10 , 10 , 15)	44.59	2.14e-02 (66.6)	3.96e-05 (107)

A simple way to improve the rank estimate is to repeat Phase I with several initial matrices, and adopt the one that results in the smallest rank. Table 8.3 shows the results obtained in this way using five random initial guesses.

We observe that the problem becomes more difficult when the ranks vary widely. As mentioned in Section 8.3.1, choosing the initial guesses as in [138] also worked fine, but not evidently better than random initial guesses as in Table 8.3. From the first rows in both tables we validate once again that for the rank-one case, Phase II is not really necessary – Phase I is recovering a rank-one basis reliably.

Comparison with Tensor CP Algorithm

As we describe in Appendix 8.6, if the subspace is spanned by rank-one matrices, then the CP decomposition (if successfully computed; the rank is a required input) of a tensor with slices

Table 8.3: Synthetic results, random initial guess from subspace repeated 5 times.

exact ranks	av. sum(ranks)	av. Phase I err (iter)	av. Phase II err (iter)
(1 , 1 , 1 , 1 , 1)	5.00	6.77e-15 (709)	6.75e-15 (0.4)
(2 , 2 , 2 , 2 , 2)	10.00	4.04e-03 (393)	9.57e-15 (9.0)
(1 , 2 , 3 , 4 , 5)	15.00	5.82e-03 (390)	1.37e-14 (18.5)
(5 , 5 , 5 , 10 , 10)	35.00	1.23e-02 (550)	3.07e-14 (55.8)
(5 , 5 , 10 , 10 , 15)	44.20	2.06e-02 (829)	8.96e-06 (227)

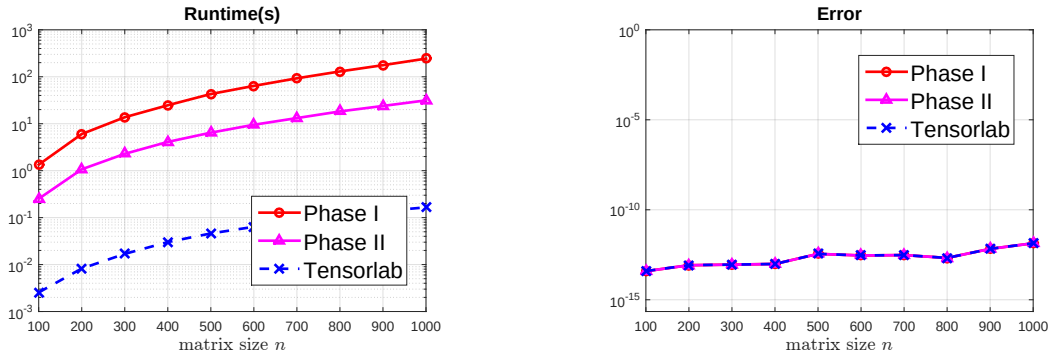


Figure 8.4: Rank-1 basis matrices $r = 1$, fixed $d = 10$, varying $m = n$ between 50 and 500. The accuracy is not identical but nearly the same. Tensorlab performs well.

M_k, where $M_1, \ldots, M_d$ is any basis of $\mathcal{M}$, provides a desired rank-one basis. Here we compare our algorithm with the CP-based approach. Specifically, we compare with the method `cpd` in Tensorlab [154] with the exact decomposition rank (the dimension d of $\mathcal{M}$) as input. By default, this method is based on alternating least-squares with initial guess obtained by an attempt of simultaneous diagonalization. When applied to a rank-one basis problem, `cpd` often gives an accurate CP decomposition with no ALS iteration.

As seen from the tables above, given a rank-one basis problem, our Algorithm 8.5 will typically terminate after Phase I. On the other hand, since we assume a rank-one basis to exist (otherwise the CP approach is not necessarily meaningful for finding a subspace basis), we can also use the alternating projection algorithm from Phase II with rank one directly from random initializations. In summary, we obtain three error curves: one for tensorlab, one for soft thresholding (Phase I) and one for alternating projection (Phase II). The errors are computed as in the experiments in Section 8.3.2 via the subspace angle.

We also present the runtime to show that our algorithms are not hopelessly slow in the special rank-one case. Just running Phase II results in an algorithm faster than Algorithm 8.5, but it is still slower than `cpd`. Note that while Tensorlab is a highly tuned toolbox, we did not try too hard to optimize our code regarding the choice of parameters and memory consumption. More importantly, unlike `cpd` our algorithm does not require the rank r and is applicable even when $r > 1$.

Growing matrix size n We first vary the matrix size n, fixing the other parameters. The runtime and accuracy are shown in Figure 8.4. We observe that if the CP rank is known, the CP-based algorithm is both fast and accurate.

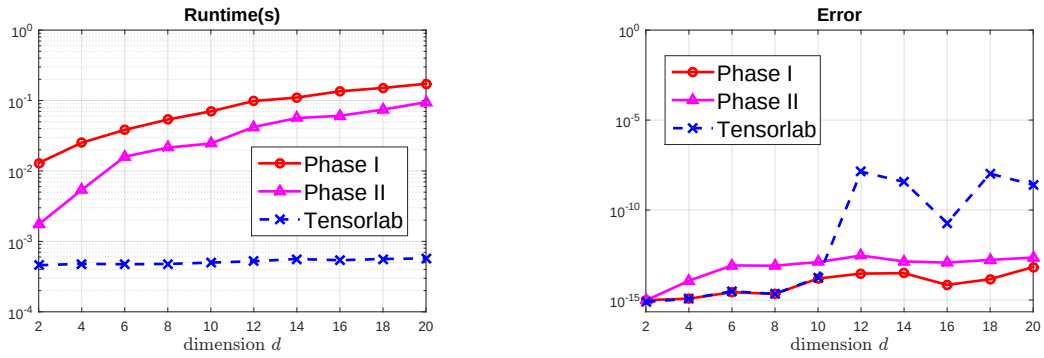


Figure 8.5: Rank-1 basis matrices $r = 1$, Fixed $m = n = 10$, varying d between 2 and 20. Our Algorithm gives better accuracy when $d > n$.

Growing dimension d We next vary the dimension d, in particular allowing it to exceed n (but not n^2). In this case, linear dependencies among the left factors $\boldsymbol{a}_\ell$ and right factors $\boldsymbol{b}_\ell$, respectively, of a rank-one basis $\boldsymbol{a}_1\boldsymbol{b}_1^\top, \ldots, \boldsymbol{a}_d\boldsymbol{b}_d^\top$ must necessarily occur. It is known that in this scenario obtaining an exact CP decomposition via simultaneous diagonalization, as in part attempted by `cpd`, becomes a much more subtle problem. And indeed, we observe that for $d > n$ the accuracy of Tensorlab deteriorates, while our methods do not. The runtime and accuracy for $n = 10$ are shown in Figure 8.5. However, this effect was less pronounced for larger $m = n$.

We conclude that the CP-based algorithm is recommended if (i) the basis matrices are known to be rank-1, and (ii) the dimension is lower than $\min(m, n)$.

8.5.2 Quality of the Convergence Estimate

In Section 8.4 we analyzed the convergence of Algorithm 8.3 in Phase II and showed that, when the error term is randomly distributed the convergence factor would be roughly $\sqrt{\frac{r}{n}}$, recall the remark after Theorem 8.4.1. Here we illustrate with experiments how accurate this estimate is.

In Figure 8.6 we plot a typical convergence of $\|\mathcal{T}_r(X) - X\|_F$ as the iterations proceed. We generated test problems (randomly as before) varying n on the left ($n = 10, 100$) and varying r on the right ($r = 2, 10$). The dashed lines indicate the convergence estimate $(\sqrt{\frac{r}{n}})^\ell$ after the ℓth iteration. Observe that in both cases the estimated convergence factors reflect the actual convergence reasonably well, and in particular we verify the qualitative tendency that (i) for fixed matrix size n, the convergence is slower for larger rank r, and (ii) for fixed rank r, the convergence is faster for larger n.

8.5.3 Image Separation

One well known use of the SVD is for data and image compression, although currently it is no longer used for the JPEG format or other modern image formats. It is known that most images can be compressed significantly without losing the visible quality by using a low-rank approximation of the matrix that represents the image.

Since each grayscale image can be expressed as a matrix, here we apply our algorithm to a set of four incomprehensible images (shown as "mixed" in Figure 8.7) that are obtained as a linear combination of four images (the famous woman.jpg in MATLAB, along with photos of Audrey Hepburn, Angelina Jolie and Arnold Schwarzenegger, taken from the labeled faces in the wild dataset [86]), with the singular values for each image truncated to have exact rank

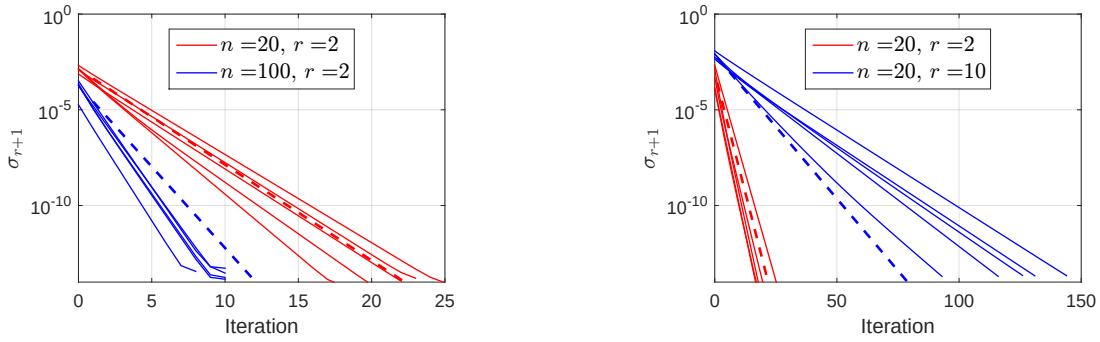


Figure 8.6: Convergence of $\|\mathcal{T}_r(X)-X\|_F$ as the iterations proceed in Phase II. The convergence factor is faster for larger matrices when the rank is fixed (left), and slower for higher rank when the matrix size is fixed (right), reflecting Theorem 8.4.1.

15. Since each original image is low-rank, we can recover them by our algorithm as shown in Figure 8.7.

We note that such a problem has been considered extensively in the image processing literature [1], [21], [177], which is known as *image separation*, and we make no claims regarding the actual usefulness of our approach in image processing; in particular, our approach would not work well if the matrices of the images are not low-rank but have gradually decaying singular values. This example is included simply for an illustrative visualization of the recovery of the low-rank matrix basis by Algorithm 8.5.

8.5.4 Computing Exact Eigenvectors of a Multiple Eigenvalue

Eigenvectors of a multiple eigenvalue are not unique. For example, the identity matrix I has any vector as an eigenvector. However, among the many possibilities one might naturally wish to obtain "nice" eigenvectors: for example, the columns of I might be considered a good set of "nice" eigenvectors for I, as they require minimum storage.

Numerically, the situation is even more complicated: a numerically stable algorithm computes eigenpairs $(\hat{\lambda}, \hat{\boldsymbol{x}})$ with residual $A\hat{\boldsymbol{x}} - \hat{\lambda}\hat{\boldsymbol{x}} = O(u\|A\|)$, where u is the unit roundoff. Since the eigenvector condition number is $O(\frac{1}{gap})$ [156, Sec. 1.3] where *gap* is the distance between λ and the rest of the eigenvalues, the accuracy of a computed eigenvector is typically $O(\frac{u\|A\|}{gap})$. This indicates the difficulty (or impossibility in general) of computing accurate eigenvectors for near-multiple eigenvalues in finite precision arithmetic. The common compromise is to compute a subspace corresponding to a cluster of eigenvalues, which is stable provided the cluster is well separated from the rest [14, Sec. 4.8].

Here we shall show nonetheless that it is sometimes possible to compute exact eigenvectors of (near) multiple eigenvalues, if additional property is present that the eigenvectors are low-rank when matricized. As we discussed in the introduction, this also lets us compress the memory to store the information. Below we illustrate how this can be done with examples.

Eigenvectors of a Multiple Eigenvalue of a Circulant Matrix

As is well known, the eigenvector matrix of a circulant matrix is the FFT matrix [73, Sec. 4.8]. One can easily verify that each column of an $n^2 \times n^2$ FFT matrix F is rank-one when matricized to $n \times n$, exemplifying a strong low-rank property.

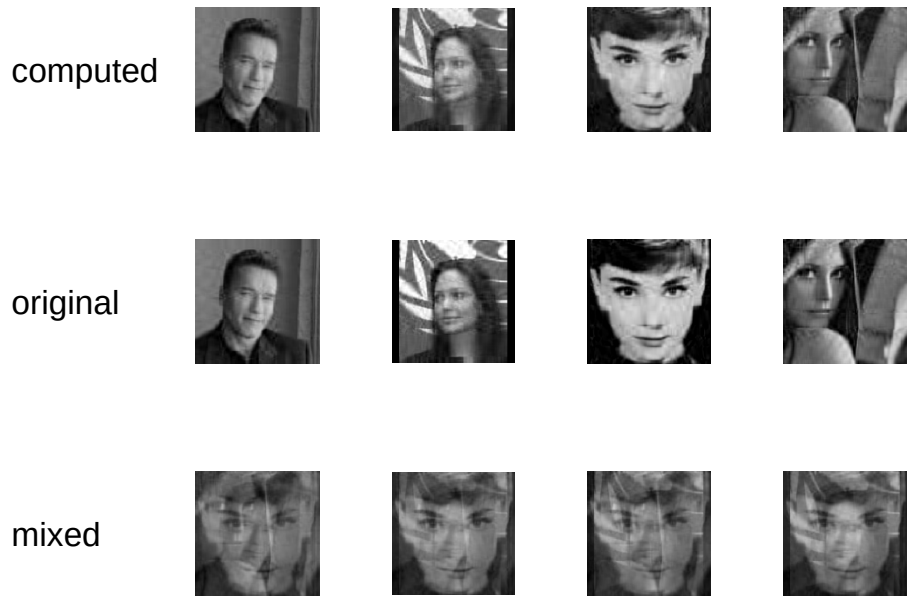


Figure 8.7: We start with the bottom images, which are obtained as random linear combinations of those in the middle row. This gives 4 matrices of size 200×200, and we apply our algorithm to obtain the top three images. Note how well the top images recover the original ones.

Let us consider a circulant matrix $A \in \mathbb{C}^{n^2 \times n^2}$ defined by

$$A = \frac{1}{n} F \Lambda F^*, \tag{8.22}$$

where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_{n^2})$.

Suppose for the moment that one is oblivious of the circulant structure (or perhaps more realistically, we can think of a matrix A that is not circulant but has d eigenvectors consisting of columns of F; such a matrix gives similar results) and attempts to compute the d smallest eigenvalues of A by a standard algorithm such as QR.

For the reason explained above, the numerically computed eigenvectors $\hat{\boldsymbol{x}}_i$ obtained by MATLAB's `eig` have poor accuracy. For concreteness suppose that $\lambda_1 = \lambda_2 = \cdots = \lambda_d$ and $\lambda_{d+i} = \lambda_{d+i-1} + 1$ for integers i, and we look for the eigenvectors corresponding to the first d eigenvalues. With $n = 10$ and $d = 5$, the smallest angle between $\hat{\boldsymbol{x}}_i$ and the first d columns of the Fourier matrix were $O(1)$ for each i (it should be 0 if $\hat{\boldsymbol{x}}_i$ was exact). Nonetheless, the subspace spanned by the d computed eigenvectors $[\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_d]$ has accuracy $O(u)$, as there is sufficient gap between λ_k and λ_{d+1}. We therefore run our algorithm with the $n \times n$ matrix subspace

$$\mathcal{M} = \mathrm{span}\{\mathrm{mat}(\hat{\boldsymbol{x}}_1), \ldots, \mathrm{mat}(\hat{\boldsymbol{x}}_d)\}.$$

Our algorithm correctly finds the rank ($= 1$), and finds the eigenvectors $[x_1, \ldots, x_d]$, each of which is numerically rank-one and has $O(u)$ angle with a column of the Fourier matrix. This is an example where by exploiting structure we achieve high accuracy that is otherwise impossible with a backward stable algorithm; another established example being the singular values for bidiagonal matrices [52, Ch. 5].

For example, we let $n^2 = 20^2$ and compute the smallest 5 eigenvalues of a circulant matrix $A = \frac{1}{n} F \mathrm{diag}(1+\epsilon_1, 1+\epsilon_2, 1+\epsilon_3, 1+\epsilon_4, 1+\epsilon_5, 6, \ldots, n^2) F^*$ where $\epsilon_i = O(10^{-10})$ was taken randomly. The matrix A therefore has a cluster of five eigenvalues near 1. The "exact" eigenvectors are the first five columns of the FFT matrix.

Table 8.4: Accuracy (middle columns) and memory usage for computed eigenvectors of a $20^2 \times 20^2$ circulant matrix.

	v_1	v_2	v_3	v_4	v_5	memory
`eig`	4.2e-01	1.2e+00	1.4e+00	1.4e+00	1.5e+00	$O(n^2)$
`eig`+Alg. 8.5	1.2e-12	1.2e-12	1.2e-12	1.2e-12	2.7e-14	$O(n)$

Note that our algorithm recovers the exact eigenvector of a near-multiple eigenvalue with accuracy $O(10^{-12})$. Furthermore, the storage required to store the eigenvectors has been reduced from $5n^2$ to $5n$.

Matrices with Low-Rank Eigenvectors

Of course, not every vector has low-rank structure when matricized. Nonetheless, we have observed that in many applications, the eigenvectors indeed have a low-rank structure that can be exploited. This observation may lead to the ability to deal with problems of scale much larger than previously possible.

Circulant matrices are an important example, as we have seen above (which clearly includes symmetric tridiagonal, symmetric banded, etc). We have observed that a sparse perturbation of a circulant matrix also has such structure.

Other examples come from graph Laplacians. We have numerically observed that typically the Laplacian matrix of the following graphs have eigenvectors (corresponding to the smallest nonzero eigenvalues) that are low-rank: binary tree, cycle, path graph and the wheel graph all have rank 3 irrelevant of the size, the lollipop graph has rank 4 (regardless of the ratio of the complete/path parts), and the ladder graph has rank 2 and circular ladder (rank 2) regardless of the size, and barbell always has rank 5. Clearly, not every graph has such structure: a counterexample is a complete graph. Our empirical observation is that sparse graphs tend to have low-rank structure in the eigenvectors.

Note that the low-rankness of the eigenvector depends also on the ordering of the vertices of the graph[3]. An ordering that seemed natural have often exhibited low-rank property.

Our algorithm does not need to know a priori that a low-rank structure is present, as its phase I attempts to identify whether a low-rank basis exists. We suspect that identifying and exploiting such structure will lead to significant improvement in both accuracy and efficiency (both in speed and memory). Identifying the conditions under which such low-rank structure is present is left as an open problem. We expect and hope that the low-rank matrix basis problem will find use in applications beyond those described above.

8.6 Finding Rank-One Bases via Tensor Decomposition

In this section, we describe the rank-one basis problem as a tensor decomposition problem. Recall that in this problem, we are promised that the given subspace $\mathcal{M}$ is spanned by rank-one matrices. Thus we can apply Algorithm 8.3 (Phase II) with the precise rank guess directly. Alternatively, we can also stop after Algorithm 8.2 (Phase I), which in practice performs well (see Section 8.5.1). The following tensor decomposition viewpoint leads to further algorithms.

Let $M_1, \ldots, M_d$ be an arbitrary basis of $\mathcal{M}$, and let $\mathcal{T}$ be the $m \times n \times d$ tensor whose 3-slices are $M_1, \ldots, M_d$. The fact that $\mathcal{M}$ possesses a rank-one basis is *equivalent* to the existence of d

[3]We thank Yuichi Yoshida for this observation.

(and not less) triplets of vectors $(\boldsymbol{a}_\ell, \boldsymbol{b}_\ell, \boldsymbol{c}_\ell)$ where $\boldsymbol{a}_\ell \in \mathbb{R}^m, \boldsymbol{b}_\ell \in \mathbb{R}^n, \boldsymbol{c}_\ell \in \mathbb{R}^d$, such that

$$M_k = \sum_{\ell=1}^{d} c_k(\ell) \boldsymbol{a}_\ell \boldsymbol{b}_\ell^\top, \quad k = 1, \ldots, d. \tag{8.23}$$

Namely, if such triplets $(\boldsymbol{a}_\ell, \boldsymbol{b}_\ell, \boldsymbol{c}_\ell)$ exist, then the assumed linear independence of the M_k automatically implies that rank-one matrices $\boldsymbol{a}_\ell \boldsymbol{b}_\ell^\top$ belong to $\mathcal{M}$. Using the outer product of vectors (denoted by $\circ$), we may express this relation in terms of the tensor $\mathcal{T}$ as

$$\mathcal{T} = \sum_{\ell=1}^{d} \boldsymbol{a}_\ell \circ \boldsymbol{b}_\ell \circ \boldsymbol{c}_\ell. \tag{8.24}$$

This type of tensor decomposition into a sum of outer products is called the *CP decomposition*, and is due to Hitchcock [85] (although the term CP decomposition appeared later). In general, the smallest d required for a representation of the form (8.24) is called the (canonical) rank of the tensor $\mathcal{T}$. We refer to [101] and references therein for more details. In summary, we have the following trivial conclusion.

Proposition 8.6.1. *The d-dimensional matrix space $\mathcal{M} = \operatorname{span}(M_1, \ldots, M_d)$ possesses a rank-one basis if and only if the tensor $\mathcal{T}$ whose 3-slices are the $M_1, \ldots, M_d$ has (canonical) rank d. Any CP decomposition (8.24) of $\mathcal{T}$ provides a rank-one basis $\boldsymbol{a}_1 \boldsymbol{b}_1^\top, \ldots, \boldsymbol{a}_d \boldsymbol{b}_d^\top$ of $\mathcal{M}$.*

We remark that computing the rank of a general third-order tensor is known to be NP-hard [82], [84]. Therefore, it is NP-hard to check whether a matrix space $\mathcal{M}$ admits a rank-one basis. Nevertheless, we might try to find a rank-one basis by trying to calculate a CP decomposition (8.24) from linearly independent $M_1, \ldots, M_d$.

Chapter 9

Introduction to Compressed Sensing

Compressed sensing, the subject of recovering a sparse signal vector $\boldsymbol{x} \in \mathbb{R}^n$ from its measurement vector $\boldsymbol{y} = A\boldsymbol{x}$ for some known matrix $A \in \mathbb{R}^{m \times n}$ with $m \ll n$, has been intensively studied in the last decade [24], [53], [64]. In this chapter, we briefly review the literature of compressed sensing. Our presentation is based on the monograph [64]. For further details, refer to [25], [64].

9.1 ℓ_1 Minimization and Null Space Property

A straightforward formulation for compressed sensing is as follows:

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{z}\|_0 \\ \text{subject to} \quad & A\boldsymbol{z} = \boldsymbol{y}, \end{aligned} \tag{P_0}$$

where $\|\boldsymbol{z}\|_0$ is the number of nonzero entries in $\boldsymbol{z}$. This formulation is exact but not useful since solving this optimization is NP-hard [127].

The basic idea of ℓ_1 *minimization* (also known as *basis pursuit*) is relaxing the ℓ_0-norm by its convex envelope, the ℓ_1-norm. Therefore ℓ_1 minimization is to solve the following optimization:

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{z}\|_1 \\ \text{subject to} \quad & A\boldsymbol{z} = \boldsymbol{y}. \end{aligned} \tag{P_1}$$

Note that (P_1) is polynomial-time solvable, since it reduces to linear programming.

Then the crucial question is when does the solution of (P_1) coincide with that of (P_0). A well-known condition is the *null space property* (NSP). We say that a matrix A satisfies the NSP with respect to $S \subseteq [n]$ if

$$\|\boldsymbol{v}_S\|_1 < \|\boldsymbol{v}_{\overline{S}}\|_1 \tag{9.1}$$

for any $\boldsymbol{v} \in \ker A \backslash \{\boldsymbol{0}\}$. Similarly, we say that A satisfies the NSP with sparsity s if the NSP with respect to S holds for any S with $|S| \leq s$. The following is fundamental to the ℓ_1 minimization.

Theorem 9.1.1. *For any vector $\boldsymbol{x}$ with support S, $\boldsymbol{x}$ is the unique optimal solution of ℓ_1 minimization (P_1) with $\boldsymbol{y} = A\boldsymbol{x}$ if and only if A satisfies the NSP with respect to S.*

Proof. Let us fix S and $\boldsymbol{v} \in \ker A \setminus \{\boldsymbol{0}\}$ arbitrary. Then $\boldsymbol{v}_S$ is the unique optimal solution to $\min\{\|\boldsymbol{z}\| : A\boldsymbol{z} = A\boldsymbol{v}_S\}$. Since $A\boldsymbol{v} = \boldsymbol{0}$, $\boldsymbol{v}_{\overline{S}}$ is also feasible. By the uniqueness of the optimal solution, we have $\|\boldsymbol{v}_S\|_1 < \|\boldsymbol{v}_{\overline{S}}\|_1$, which establishes the NSP.

Conversely, assume that the NSP with respect to S is satisfied. Let $\boldsymbol{z}$ be a feasible solution of ℓ_1 minimization (P_1) and $\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{z}$. Then $\boldsymbol{v} \in \ker A$ since $A\boldsymbol{v} = A\boldsymbol{x} - A\boldsymbol{z} = \boldsymbol{y} - \boldsymbol{y} = \boldsymbol{0}$. We also have

$$\|\boldsymbol{x}\|_1 \leq \|\boldsymbol{x} - \boldsymbol{z}_S\|_1 + \|\boldsymbol{z}_S\|_1 = \|\boldsymbol{v}_S\|_1 + \|\boldsymbol{z}_S\|_1 < \|\boldsymbol{v}_{\overline{S}}\|_1 + \|\boldsymbol{z}_S\|_1 = \|\boldsymbol{z}_{\overline{S}}\|_1 + \|\boldsymbol{z}_S\|_1 = \|\boldsymbol{z}\|_1,$$

which shows the unique minimality of $\boldsymbol{x}$. □

9.2 Stable and Robust Null Space Property

In real-world applications such as image compression or sound signal processing, the original vector $\boldsymbol{x}$ is not exactly sparse but close to a sparse vector. In such cases, we want to recover $\boldsymbol{x}$ with an error controlled by its distance to s-sparse vectors, and this feature is called *stability* [64] or *instance optimality* [48] of a recovery scheme. Here, a vector is said to be *s-sparse* if its support has the cardinality at most s. For $q > 0$, an integer $s \in \mathbb{Z}_+$, and a vector $\boldsymbol{x} \in \mathbb{R}^n$, we define

$$\sigma_s(\boldsymbol{x})_q = \min\{\|\boldsymbol{x} - \boldsymbol{x}'\|_q \mid \boldsymbol{x}' \in \mathbb{R}^n, \|\boldsymbol{x}'\|_0 \leq s\}. \tag{9.2}$$

That is, $\sigma_s(\boldsymbol{x})_q$ is the distance of $\boldsymbol{x}$ to the closest s-sparse vector in the ℓ_q norm. A common choice of q is $q = 1$ or $q = 2$, but other choices of q are also interesting. It is known that a slightly stronger version of the NSP guarantees good error performance of ℓ_1 minimization in the ℓ_1 metric. For $0 < \rho < 1$ and $s \in \mathbb{Z}_+$, a matrix A is said to satisfy the *(ρ, s)-stable null space property* (SNSP) if

$$\|\boldsymbol{v}_S\|_1 < \rho\|\boldsymbol{v}_{\overline{S}}\|, \tag{9.3}$$

for any $\boldsymbol{v} \in \ker A \setminus \{\boldsymbol{0}\}$ and any S with $|S| \leq s$.

Theorem 9.2.1 (Candès, Romberg, and Tao [35])**.** *Suppose that a matrix A satisfies the (ρ, s)-SNSP (9.3). Let $\boldsymbol{x}$ be a vector and $\boldsymbol{z}$ be the optimal solution of ℓ_1 minimization (P_1) with $\boldsymbol{y} = A\boldsymbol{x}$. Then we have*

$$\|\boldsymbol{x} - \boldsymbol{z}\|_1 \leq \frac{2(1+\rho)}{1-\rho}\sigma_s(\boldsymbol{x})_1. \tag{9.4}$$

We can go further by considering more severe scenarios that the observations are noisy; i.e., $\boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{e}$ for some error vector $\boldsymbol{e}$. In order to handle such noisy observations, we can extend ℓ minimization to *robust ℓ_1 minimization*.

$$\text{minimize} \quad \|\boldsymbol{z}\|_1 \quad \text{subject to} \quad \|A\boldsymbol{z} - \boldsymbol{y}\|_2 \leq \eta, \tag{$P_{1,\eta}$}$$

where η is a parameter. Remark that ($P_{1,\eta}$) cannot be reduced to linear programming as in (P_1). However, ($P_{1,\eta}$) is a convex problem and therefore still solvable by, for example, second order cone programming (SOCP).

Being parallel to the standard NSP (9.1), one can define a robust version of the NSP. Let $\rho, \tau > 0$, and $s \in \mathbb{Z}_+$. A matrix $A \in \mathbb{R}^{m\times n}$ is said to satisfy the *(ρ, τ, s)-robust null space property* (RNSP) if

$$\|\boldsymbol{v}_S\|_1 \leq \rho\|\boldsymbol{v}_{\overline{S}}\|_1 + \tau\|A\boldsymbol{v}\|_2 \tag{9.5}$$

for any $\boldsymbol{v} \in \mathbb{R}^n$ and $S \subseteq [n]$ with $|S| \leq s$. Note that the (ρ, τ, s)-RNSP implies the NSP with sparsity s for any $0 < \rho < 1$ and $\tau > 0$. The RNSP ensures that the distance between the original signal $\boldsymbol{x}$ and the solution $\boldsymbol{z}$ of robust ℓ_1 minimization is bounded.

Theorem 9.2.2 (Candès, Romberg, and Tao [35]). *Suppose that a matrix A satisfies the (ρ, τ, s)-RNSP (9.5). Let $\boldsymbol{x}$ be a vector and $\boldsymbol{z}$ be an optimal solution of robust ℓ_1 minimization ($P_{1,\eta}$) with $\boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{e}$ and $\|\boldsymbol{e}\|_2 \leq \eta$. Then we have*

$$\|\boldsymbol{x} - \boldsymbol{z}\|_1 \leq \frac{2(1+\rho)}{1-\rho}\sigma_s(\boldsymbol{x})_1 + \frac{4\tau}{1-\rho}\eta. \tag{9.6}$$

9.3 Coherence and Restricted Isometry Property

In this section, we review examples of matrices that satisfy the NSP (9.1) and the RNSP (9.5). The main ingredient is relating the NSP and RNSP to properties of measurement matrices called coherence and restricted isometry property.

9.3.1 Coherence

Let $A \in \mathbb{R}^{m\times n}$ be a matrix with ℓ_2-normalized columns $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$. The *coherence* of a matrix $A \in \mathbb{R}^{m\times n}$ is defined as $\mu = \max_{i,j:i\neq j} |\langle \boldsymbol{a}_i, \boldsymbol{a}_j\rangle|$. The *$\ell_1$-coherence function* of A is defined as

$$\mu_1(s) := \max_{i\in[n]} \max \left\{ \sum_{j\in S} |\langle \boldsymbol{a}_i, \boldsymbol{a}_j\rangle| : S \subseteq [n], |S| = s, i \notin S \right\} \tag{9.7}$$

for $s = 1, \ldots, n-1$. Evidently we have $\mu = \mu_1(1) \leq \mu_1(2) \leq \cdots \leq \mu_1(n-1)$ and $\mu \leq \mu_1(s) \leq s\mu$.

A matrix with small coherence approximately acts as an isometry for any sparse vector. Indeed

$$(1 - \mu_1(s-1))\|\boldsymbol{x}\|_2 \leq \|A\boldsymbol{x}\|_2 \leq (1 + \mu_1(s-1))\|\boldsymbol{x}\|_2 \tag{9.8}$$

for any s-sparse vector $\boldsymbol{x}$. Also, a matrix with small coherence satisfies the NSPs.

Theorem 9.3.1. *Let A be a matrix with ℓ_2-normalized columns. If*

$$\mu_1(s) + \mu_1(s-1) < 1, \tag{9.9}$$

then A satisfies the NSP with sparsity s. This condition holds if

$$\mu < \frac{1}{2s-1}. \tag{9.10}$$

More generally, if $\mu_1(s) < 1/2$, then A satisfies the (ρ, τ, s)-RNSP with

$$\rho = \frac{\mu_1(s)}{1 - \mu_1(s-1)} \text{ and } \tau = \frac{s}{1 - \mu_1(s-1)}. \tag{9.11}$$

The above theorem prompts us to construct a matrix with small coherence. In the literature, various construction of such matrices has been known (see [64]).

Coherence is a handy concept; one can easily compute the coherence of a given matrix and analysis with coherence is usually simpler than that with restricted isometry property. Moreover, there exist several deterministic ways to construct matrices with small coherence. On the other hand, the sample complexity of sensing with small coherence matrices can be suboptimal. In fact, to ensure the coherence condition (9.10) above, we need $\Omega(s^2)$ samples, whereas reconstruction with $O(s\log(s/n))$ samples is possible as we see in the following.

9.3.2 Restricted Isometry Property

The *restricted isometry property* (RIP) was introduced by Candes and Tao [37]. Let A be an $m \times n$ matrix with ℓ_2 normalized columns. The basic idea is a refinement of the near-isometry inequality (9.8). For $s \in \mathbb{Z}_+$, the *sth restricted isometry constant* δ_s is defined as the smallest value satisfying

$$(1 - \delta_s)\|\boldsymbol{x}\|_2 \leq \|A\boldsymbol{x}\|_2 \leq (1 + \delta_s)\|\boldsymbol{x}\|_2. \tag{9.12}$$

for any s-sparse vector $\boldsymbol{x}$. Equivalently,

$$\delta_s = \max_{S \subseteq [n]: |S| \leq s} \|I - A_S^\top A_S\|, \tag{9.13}$$

where A_S is the column submatrix of A whose columns are indexed by S. Evidently we have $\delta_1 \leq \delta_2 \leq \cdots \leq \delta_n$. Similarly to coherence, the small restricted isometry constant implies the NSPs.

Theorem 9.3.2 (Candès [32])**.** *If a matrix A satisfies $\delta_{2s} < \sqrt{2} - 1$, then the NSP (9.1) with sparsity s is satisfied. Moreover, under the same condition, the (ρ, τ)-RNSP (9.5) holds, where ρ and τ are constants depending only on δ_{2s}.*

We remark that the constant $\sqrt{2} - 1 \approx 0.4142$ has been improved to $4/\sqrt{46} \approx 0.6246$ by [9].

How can we construct a matrix with a small restricted isometry constant? In fact, computing the restricted isometry constant of a given matrix with given sparsity is NP-hard [164]. The only known method to obtain such a matrix is resorting to *random matrices*. For example, *Gaussian matrices*, random matrices whose entries are independent standard Gaussian random variables, satisfy the required condition with high probability. Similarly, *Rademacher matrices*, random matrices whose entries are random sign ± 1 with the equal probability, also have small restricted isometry constants. The following theorem establishes the fundamental fact that these families of random matrices yield reconstruction with $O(s \log(s/n))$ sample complexity.

Theorem 9.3.3. *Let A be an $m \times n$ Gaussian matrix, $s \geq 1$, and δ_s be the sth restricted isometry constant of $\frac{1}{\sqrt{m}}A$. Then there exists a constant $C > 0$ such that for any $\delta > 0$,*

$$\Pr(\delta_s \leq \delta) \geq 1 - 2\exp\left(-\frac{\delta^2 m}{2C}\right), \tag{9.14}$$

provided that $m \geq 2C\delta^{-2} s \log(en/s)$. The same bound holds even if A is a Rademacher matrix.

Note that each column of $\frac{1}{\sqrt{m}}A$ has the unit ℓ_2-norm in expectation. Combining theorems above, we can reconstruct any s-sparse vector with ℓ_1 minimization (P_1), using a Gaussian or Rademacher matrix with $O(s \log(s/n))$ columns.

9.4 Instance Optimality

In the previous section, we saw that given $m = O(s \log(s/n))$ samples from a random sensing matrix, we can reconstruct a sparse signal with high probability. In this section we see that this sample complexity is optimal in the sense specified below.

Let us first define the notion of instance optimality. For $p > 0$, a sensing matrix $A \in \mathbb{R}^{m \times n}$ and a reconstruction map $\Delta : \mathbb{R}^m \to \mathbb{R}^n$ are said to be *ℓ_p-instance optimal* of order s with constant $C > 0$ if

$$\|\boldsymbol{x} - \Delta(A\boldsymbol{x})\|_p \leq C\sigma_s(\boldsymbol{x})_p, \tag{9.15}$$

for any $\boldsymbol{x} \in \mathbb{R}^n$. Note that a reconstruction map Δ is not restricted to a polynomial time algorithm but can be any map.

In the previous sections, we have already seen that if A is a random Gaussian matrix with $m = O(s\log(s/n))$ and Δ is ℓ_1 minimization (P_1), then (A, Δ) is ℓ_1-instance optimal (see Theorem 9.2.1). Indeed the converse also holds; i.e., *any* ℓ_1-instance optimal reconstruction method requires $\Omega(s\log(s/n))$ samples.

Theorem 9.4.1 ([63]). *Suppose that a matrix $A \in \mathbb{R}^{m\times n}$ and a map $\Delta : \mathbb{R}^m \to \mathbb{R}^n$ are ℓ_1-instance optimal with constant $C > 0$ of order s. Then*

$$m \geq cs\log(en/s) \tag{9.16}$$

for some constant $c > 0$ depending only on C.

On the other hand, any ℓ_2-instance optimal reconstruction method must require much many samples.

Theorem 9.4.2 ([48]). *Suppose that a matrix $A \in \mathbb{R}^{m\times n}$ and a map $\Delta : \mathbb{R}^m \to \mathbb{R}^n$ are ℓ_2-instance optimal with constant $C > 0$ of order s. Then*

$$m \geq cn \tag{9.17}$$

for some constant $c > 0$ depending only on C.

Chapter 10

Nonconvex Compressed Sensing with the Sum of Squares Method

10.1 Introduction

Let us recall stable signal recovery in Chapter 9. The objective of stable signal recovery in ℓ_q quasi-norm is, given a measurement vector $\boldsymbol{y} = A\boldsymbol{x}$ generated from an unknown signal vector $\boldsymbol{x} \in \mathbb{R}^n$, to recover a vector $\boldsymbol{z} \in \mathbb{R}^n$ such that $\|\boldsymbol{z} - \boldsymbol{x}\|_q = O(\sigma_s(\boldsymbol{x})_q)$. It is desirable to take a small q, for example, in the sparse noise model; that is, $\boldsymbol{x} = \boldsymbol{x}' + \boldsymbol{e}$, where $\boldsymbol{x}'$ is the best s-sparse approximation to $\boldsymbol{x}$ and $\boldsymbol{e}$ is an s'-sparse noise with $s' = O(1)$. To see this, let us compare the cases $q = 1/2$ and $q = 1$ as an illustrative example. Note that $\|\boldsymbol{z} - \boldsymbol{x}\|_1 \leq \|\boldsymbol{z} - \boldsymbol{x}\|_{1/2} \leq n\|\boldsymbol{z} - \boldsymbol{x}\|_1$ and $\sigma_s(\boldsymbol{x})_{1/2} = \|\boldsymbol{e}\|_{1/2} \leq s'\|\boldsymbol{e}\|_1 = s'\sigma_s(\boldsymbol{x})_1$. Hence, $\|\boldsymbol{z} - \boldsymbol{x}\|_{1/2} = O(\sigma_s(\boldsymbol{x})_{1/2})$ implies $\|\boldsymbol{z} - \boldsymbol{x}\|_1 = O(\sigma_s(\boldsymbol{x})_1)$, whereas $\|\boldsymbol{z} - \boldsymbol{x}\|_1 = O(\sigma_s(\boldsymbol{x})_1)$ only implies a weaker bound $\|\boldsymbol{z} - \boldsymbol{x}\|_{1/2} = O(n\sigma_s(\boldsymbol{x})_{1/2})$. It is reported that the noise vector $\boldsymbol{e}$ is often sparse in practice [46], [106]. Intuitively speaking, smaller q works better in the sparse noise model because $\|\boldsymbol{z} - \boldsymbol{x}\|_q^q \to \|\boldsymbol{z} - \boldsymbol{x}\|_0$ and $\sigma_s(\boldsymbol{x})_q \to s'$ as $q \to 0$, and hence $\boldsymbol{z}$ converges to an $(s + O(s'))$-sparse vector.

A common approach to stable signal recovery is relaxing the ℓ_0 quasi-norm by an ℓ_q quasi-norm for $0 < q \leq 1$:

$$\text{minimize } \|\boldsymbol{z}\|_q \quad \text{subject to} \quad A\boldsymbol{z} = \boldsymbol{y} \text{ and } \boldsymbol{z} \in \mathbb{R}^n. \tag{P_q}$$

For $q = 1$, this approach is called *basis pursuit*, which we saw in the previous chapter. If A satisfies some technical condition, called the *stable null space property*, then the solution $\boldsymbol{z}$ to (P_q) satisfies $\|\boldsymbol{z} - \boldsymbol{x}\|_q = O(\sigma_s(\boldsymbol{x})_q)$. It is folklore that the following result can be obtained by extending the results in [32], [36], which handle the $q = 1$ case.

Theorem 10.1.1. *Let $q \in (0, 1]$ and $s, n \in \mathbb{Z}_+$. Then, a random Gaussian or Rademacher matrix $A \in \mathbb{R}^{m \times n}$ with $m = \Theta(s \log(n/s))$ satisfies the following property with high probability: For any vectors $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} = A\boldsymbol{x}$, the solution $\boldsymbol{z}$ to (P_q) satisfies $\|\boldsymbol{z} - \boldsymbol{x}\|_q = O(\sigma_s(\boldsymbol{x})_q)$.*

When $q = 1$, i.e., in the basis pursuit, the program (P_q) can be represented as a linear program and we can find its solution in polynomial time. When $q < 1$, however, solving (P_q) is NP-hard in general and Theorem 10.1.1 is currently only of theoretical interest.

The main contribution of this chapter is showing that, even when $q < 1$, there is a recovery scheme whose theoretical guarantee almost matches that of Theorem 10.1.1:

Theorem 10.1.2 (Main theorem, informal version)**.** *Let $q \in (0, 1]$ be a constant of the form $q = 2^{-k}$ for some $k \in \mathbb{Z}_+$, $s, n \in \mathbb{Z}_+$, and $\epsilon > 0$. Then, a random (normalized) Rademacher*

matrix $A \in \mathbb{R}^{m \times n}$ *with* $m = O(s^{2/q} \log n)$ *satisfies the following property with high probability: For any vector* $\boldsymbol{y} = A\boldsymbol{x}$*, where* $\boldsymbol{x} \in [-1, 1]^n$ *is an unknown vector, we can recover* $\boldsymbol{z}$ *with* $\|\boldsymbol{z} - \boldsymbol{x}\|_q \leq O(\sigma_s(\boldsymbol{x})_q) + \epsilon$ *in polynomial time.*

To our knowledge, our algorithm is the first polynomial-time stable recovery scheme in ℓ_q quasi-norm for $q < 1$. Our algorithm is based on the sum-of-squares method, which we describe next. The assumption on $\boldsymbol{x}$ can be easily relaxed to $\boldsymbol{x} \in [-R, R]$ for a polynomial R in n, keeping the running time polynomial in n.

10.1.1 Sum of Squares Method

As we cannot exactly solve the non-convex program (P_q) when $q < 1$, we use the *sum-of-squares* (SoS) semidefinite programming hierarchy [107], [131], [136], [150], or simply the SoS method. Here, we adopt the description of the SoS method introduced in [16], [17]. The SoS method attempts to solve a *polynomial optimization problem* (POP), via convex relaxation called the *moment* relaxation. We specify an even integer d called the *degree* or the *level* as a parameter. The SoS method will find a linear operator $\mathcal{L}$ from the set of polynomials of degree at most d to $\mathbb{R}$, which shares some properties of expectation operators and satisfies the system of polynomials. Such linear operators are called *pseudoexpectations*.

Definition 10.1.3 (Pseudoexpectation)**.** Let $\mathbb{R}[\boldsymbol{z}]$ denote the polynomial ring over the reals in variables $\boldsymbol{z} = (z(1), \ldots, z(n))$ and let $\mathbb{R}[\boldsymbol{z}]_d$ denote the set of polynomials in $\mathbb{R}[\boldsymbol{z}]$ of degree at most d. A degree-d pseudoexpectation operator for $\mathbb{R}[\boldsymbol{z}]$ is a linear operator $\widetilde{\mathbf{E}} : \mathbb{R}[\boldsymbol{z}]_d \to \mathbb{R}$ satisfying that $\widetilde{\mathbf{E}}(1) = 1$ and $\widetilde{\mathbf{E}}(P^2) \geq 0$ for every polynomial P of degree at most $d/2$.

For notational simplicity, it is convenient to regard a pseudoexpectation operator as the "expectation" operator of the so-called *pseudodistribution*[1] $\mathcal{D}$. Note that it is possible that no actual distribution realizes a pseudoexpectation operator as the expectation operator (see, e.g. [47], [142]). Also note that we cannot sample from a pseudodistribution; only its low-degree moments are available. We use the subscript $\widetilde{\mathbf{E}}_{\mathcal{D}}$ to emphasize the underlying pseudodistribution. Also we write $\widetilde{\mathbf{E}}_{\boldsymbol{z} \sim \mathcal{D}} P(\boldsymbol{z})$ to emphasize the variable $\boldsymbol{z}$. We say that a degree-d pseudodistribution $\mathcal{D}$ satisfies the constraint $\{P = 0\}$ if $\widetilde{\mathbf{E}}\, PQ = 0$ for all Q of degree at most $d - \deg P$. We say that $\mathcal{D}$ satisfies $\{P \geq 0\}$ if it satisfies the constraint $\{P - S = 0\}$ for some SoS polynomial $S \in \mathbb{R}_d[\boldsymbol{z}]$.

A pseudoexpectation can be compactly represented by its moment matrix, an $n^{O(d/2)} \times n^{O(d/2)}$ symmetric matrix M containing all the values for monomials of degree at most d. The above condition $\widetilde{\mathbf{E}}\, P^2 \geq 0$ is equivalent to requiring that M is positive semidefinite, and the other conditions can be represented by linear constraints among entries of M. The following theorem states that we can efficiently optimize over pseudodistributions, basically via semidefinite programming.

Theorem 10.1.4 (The SoS method [107], [131], [136], [150])**.** *For every* $\epsilon > 0$*,* $d, n, m, M \in \mathbb{N}$ *and* n*-variate polynomials* $P_1, \ldots, P_m$ *in* $\mathbb{R}_d[\boldsymbol{z}]$*, whose coefficients are in* $0, \ldots, M$*, if there exists a degree-*d *pseudodistribution* $\mathcal{D}$ *for* $\boldsymbol{z}$ *satisfying the constraint* $P_i = 0$ *for every* $i \in [m]$*, then we can find in* $(n \cdot \operatorname{poly} \log(M/\epsilon))^{O(d)}$ *time a degree-*d *pseudodistribution* $\mathcal{D}'$ *for* $\boldsymbol{z}$ *satisfying* $P_i \leq \epsilon$ *and* $P_i \geq -\epsilon$ *for every* $i \in [m]$*.*

By using binary search, we can also handle objective functions. We ignore numerical issues since it will never affect our arguments (at least from a theoretical perspective). We simply assume that the SoS method finds an optimal degree-d pseudodistribution in $\operatorname{poly}(n^{O(d)})$ time.

[1] It is also called *locally consistent distribution* [47], [142].

The dual of the SoS method corresponds to the *SoS proof system* [76], an automizable and restricted proof system for refuting a system of polynomials. The SoS proof system are built on several deep results of real algebraic geometry, such as *positivestellensatz* and its variants [137], [144]. Roughly speaking, any fact that is provable in the (bounded-degree) SoS proof system is also reflected to pseudodistributions. This approach lies at the heart of recent surprising results [16], [133]; they imported known proofs for integrality gap instances into the SoS proof system and showed that (low-degree) SoS hierarchy solves the integrality gap instances.

10.1.2 Proof Outline

Now we give a proof outline of Theorem 10.1.2. First we recall basic facts in stable signal recovery using the non-convex program (P_q). For a set $S \subseteq [n]$, we denote by $\overline{S}$ the complement set $[n] \setminus S$. For a vector $\boldsymbol{v} \in \mathbb{R}^n$ and a set $S \subseteq [n]$, $\boldsymbol{v}_S$ denotes the vector restricted to S. We abuse this notation and regard $\boldsymbol{v}_S$ as the vector in $\mathbb{R}^S$ with $v_S(i) = v(i)$ for $i \in S$ as well as a vector in $\mathbb{R}^n$ with $v_S(i) = v(i)$ for $i \in S$ and $v_S(i) = 0$ for $i \in \overline{S}$. The following notion is important in stable signal recovery.

Definition 10.1.5 (Robust null space property)**.** Let $q \in (0, 1]$, $\rho, \tau > 0$, and $s \in \mathbb{Z}_+$. A matrix $A \in \mathbb{R}^{m \times n}$ is said to satisfy the (ρ, τ, s)*-robust null space property* (RNSP) in ℓ_q quasi-norm if

$$\|\boldsymbol{v}_S\|_q^q \leq \rho \|\boldsymbol{v}_{\overline{S}}\|_q^q + \tau \|A\boldsymbol{v}\|_2^q \tag{10.1}$$

holds for any $\boldsymbol{v} \in \mathbb{R}^n$ and $S \subseteq [n]$ with $|S| \leq s$.

If A satisfies the (ρ, τ, s)-robust null space property for $\rho < 1$, then the solution to the program (P_q) satisfies $\|\boldsymbol{z} - \boldsymbol{x}\|_q = O(\sigma_s(\boldsymbol{x})_q)$ and we establish Theorem 10.1.1. Indeed, a random Gaussian or Rademacher matrix will do as long as $m = \Omega(s \log n/s)$ [36].

The main idea of our algorithm is that we can convert the proof of Theorem 10.1.1 into a *sum of squares (SoS) proof*, a proof such that all the polynomials appearing there have bounded degree and all the inequalities involved are simply implied by the SoS partial order. Due to the duality between SoS proofs and the SoS semidefinite programming hierarchy (see, e.g. [133]), any consequence of a SoS proof derived from a given set of polynomial constraints is satisfied by the pseudodistribution obtained by the SoS method on these constraints.

Let us consider the following SoS version of (P_q).

$$\text{minimize } \widetilde{\mathbf{E}}_{\boldsymbol{z} \sim \mathcal{D}} \|\boldsymbol{z}\|_q^q \quad \text{subject to } \mathcal{D} \text{ satisfies } A\boldsymbol{z} = \boldsymbol{y}, \tag{$\widetilde{P}_q$}$$

where $\mathcal{D}$ is over pseudodistributions for the variable $\boldsymbol{z}$. In our algorithm, we solve ($\widetilde{P}_q$) with technical additional constraints using the SoS method, and then recover a vector from the obtained pseudoexpectation operator. A subtle issue here is that we need to deal with terms of the form $|\boldsymbol{z}(i)|^q$ for fractional q, which is not polynomial in $\boldsymbol{z}$. A remedy for this is adding auxiliary variables and constraints among them (details are explained later), and for now we assume that we can exactly solve the program ($\widetilde{P}_q$).

As we can only execute SoS proofs by using the SoS method, we need an SoS version of the robust null space property:

Definition 10.1.6 (Pseudo robust null space property, informal version)**.** Let $q \in (0, 1]$, $\rho, \tau > 0$, and $s \in \mathbb{Z}_+$. A matrix $A \in \mathbb{R}^{m \times n}$ is said to satisfy the (ρ, τ, s)*-pseudo robust null space property* in ℓ_q quasi-norm (ℓ_q-(ρ, τ, s)-PRNSP for short) if

$$\widetilde{\mathbf{E}}_{\boldsymbol{v} \sim \mathcal{D}} \|\boldsymbol{v}_S\|_q^q \leq \rho \, \widetilde{\mathbf{E}}_{\boldsymbol{v} \sim \mathcal{D}} \|\boldsymbol{v}_{\bar{S}}\|_q^q + \tau \left(\widetilde{\mathbf{E}}_{\boldsymbol{v} \sim \mathcal{D}} \|A\boldsymbol{v}\|_2^2 \right)^{q/2} \tag{10.2}$$

holds for any pseudodistribution $\mathcal{D}$ of degree at least $2/q$ and $S \subseteq [n]$ with $|S| \leq s$.

In the formal definition of PRNSP, we allow to impose some other technical constraints. Note that PRNSP is a stronger condition than the robust null space property because the latter considers only actual distributions consisting of single vectors.

If the measurement matrix $A \in \mathbb{R}^{m \times n}$ satisfies the PRNSP, then the resulting pseudodistribution $\mathcal{D}$ satisfies an SoS version of stable signal recovery. Moreover, we can round $\mathcal{D}$ to a vector close to the signal vector, as shown in the following theorem.

Theorem 10.1.7 (PRNSP $\Rightarrow$ stable signal recovery, informal version)**.** *Let $q \in (0, 1]$ be a constant of the form $q = 2^{-k}$ for some $k \in \mathbb{Z}_+$, $0 < \rho < 1$, $\tau > 0$, $s \in \mathbb{Z}_+$, and $\epsilon > 0$. If A satisfies the ℓ_q-(ρ, τ, s)-PRNSP, then, from the pseudodistribution $\mathcal{D}$ obtained by solving $(\widetilde{P}_q)$ with additional constraints depending on ϵ, we can compute a vector $\bar{\boldsymbol{z}} \in \mathbb{R}^n$ such that*

$$\|\bar{\boldsymbol{z}} - \boldsymbol{x}\|_q \leq \frac{4(1+\rho)}{1-\rho} \sigma_s(\boldsymbol{x})_q + \epsilon$$

in polynomial time.

We then show that there exists a family of matrices satisfying PRNSP, namely the family of Rademacher matrices.

Theorem 10.1.8 (Existence of matrices satisfying PRNSP, informal version)**.** *Let $q \in (0, 1]$ and $s \in \mathbb{Z}_+$. For some $0 < \rho < 1$ and $\tau = O(s)$, a (normalized) Rademacher matrix $A \in \mathbb{R}^{m \times n}$ with $m = \Omega(s^{2/q} \log n)$ satisfies the ℓ_q-(ρ, τ, s)-PRNSP with high probability.*

Theorem 10.1.2 is a consequence of Theorems 10.1.7 and 10.1.8.

10.1.3 Technical Contributions

In order to prove Theorem 10.1.7, we would like to follow the proof of Theorem 10.1.1 in the SoS proof system. However, many steps of the proof requires non-trivial modifications for the SoS proof system to go through. Let us explain in more detail below.

Handling ℓ_q quasi-norm The first obvious obstacle is that the program $(\widetilde{P}_q)$ contains non-polynomial terms such as $|z(i)|^q$ in the objective function. This is easily resolved by *lifting*; i.e., we introduce a variable $t_{z(i)}$, which is supposed to represent $|z(i)|^q$, and impose the constraints $t_{z(i)}^{2/q} = z(i)^2$ and $t_{z(i)} \geq 0$ (recall that q is of the form 2^{-k} for some $k \in \mathbb{Z}_+$).

Imposing the triangle inequality for ℓ_q^q metric The second problem is that the added variable $t_{z(i)}$ does not always behave as expected. When proving Theorem 10.1.1, we frequently use the triangle inequality $\|\boldsymbol{u} + \boldsymbol{v}\|_q^q \leq \|\boldsymbol{u}\|_q^q + \|\boldsymbol{v}\|_q^q$ for vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, or in other words, $\sum_i |u(i) + v(i)|^q \leq \sum_i |u(i)|^q + \sum_i |v(i)|^q$. Unfortunately, we cannot derive its SoS version $\widetilde{\mathbf{E}} \sum_i t_{u(i)+v(i)} \leq \widetilde{\mathbf{E}} \sum_i t_{u(i)} + \widetilde{\mathbf{E}} \sum_i t_{v(i)}$ in the SoS proof system. This is because, even if we impose the constraints $t_{u(i)}^{2/q} = u(i)^2$, $t_{v(i)}^{2/q} = v(i)^2$, and $t_{u(i)+v(i)}^{2/q} = (u(i) + v(i))^2$, these constraints do not involve any constraint among $t_{u(i)}$, $t_{v(i)}$, and $t_{u(i)+v(i)}$. (Instead, we can derive $\left(\widetilde{\mathbf{E}} \sum_i t_{u(i)+v(i)}^{2/q}\right)^{q/2} \leq \left(\widetilde{\mathbf{E}} \sum_i t_{u(i)}^{2/q}\right)^{q/2} + \left(\widetilde{\mathbf{E}} \sum_i t_{v(i)}^{2/q}\right)^{q/2}$ in the SoS proof system, which is not very useful.)

To overcome this issue, we use the fact that we only need the triangle inequality among $\boldsymbol{z} - \boldsymbol{x}$, $\boldsymbol{z}$, and $\boldsymbol{x}$, where $\boldsymbol{z}$ is a variable in $(\widetilde{P}_q)$ and $\boldsymbol{x}$ is the signal vector. Of course we do not know $\boldsymbol{x}$ in advance; we only know that $\boldsymbol{x}$ lies in $[-1, 1]^n$. We will discretize the region $[-1, 1]^n$ using a small $\delta \ll 1/n^{2/q}$ and explicitly add the constraints $\widetilde{\mathbf{E}}\, t_{z(i)-b} \leq \widetilde{\mathbf{E}}\, t_{z(i)} + b$ for each $i \in [n]$

and multiple $b \in [-1, 1]$ of δ. Instead of $\boldsymbol{x}$, we use an approximated vector $\boldsymbol{x}^L$ each coordinate of which is a multiple of δ with $\|\boldsymbol{x} - \boldsymbol{x}^L\|$ being small. The SoS proof system augmented with these triangle inequalities can now follow the proof of Theorem 10.1.1.

Rounding Rounding a pseudodistribution to a feasible solution is a challenging task in general. Fortunately, we can easily round the pseudodistribution $\mathcal{D}$ obtained by solving the program $(\widetilde{P}_q)$ to a vector $\bar{\boldsymbol{z}}$ close to the signal vector by fixing each coordinate $\bar{z}(i)$ to $\arg\min_b \widetilde{\mathbf{E}}_{\mathcal{D}} |z(i) - b|^q$, where $b \in [-1, 1]$ runs over multiples of δ.

Imposing the PRNSP In order to prove Theorem 10.1.8, we show that a (normalized) Rademacher matrix with a small coherence satisfies PRNSP. Here, the *coherence* of a matrix $A \in \mathbb{R}^{m \times n}$ with columns $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m$ is defined as $\mu = \max_{i,j} |\langle \boldsymbol{a}_i, \boldsymbol{a}_j \rangle|$. Again our approach is the following: rewrite a proof that matrices with a small coherence satisfy the robust null space property, into an SoS proof. Here is the last obstacle; it seems inevitable for us to add *exponentially many* extra variables and constraints, in order to convert known proofs. However, we will show that the number of extra variables is polynomially bounded and that the extra constraints admit a separation oracle, if A is a Rademacher matrix. Thanks to the ellipsoid method, we can obtain the desired SoS proof for Rademacher matrices in polynomial time.

10.1.4 Related Work

The notion of stable signal recovery in ℓ_1 norm is introduced in [32], [35] and it is shown that basis pursuit is a stable recover scheme for a random Gaussian matrix $A \in \mathbb{R}^{m \times n}$ provided that $m = \Omega(s \log(n/s))$. The notion of stable signal recovery is extended to ℓ_q quasi-norm in [48], [143]. Stable recovery schemes with almost linear encoding time and recovery time have been proposed by using sparse matrices [22], [23], [87]. See [64], [71] for a survey of this area.

Besides the fact that $\|\cdot\|_q^q$ converges to the ℓ_0 norm as $q \to 0$, an advantage of using the ℓ_q quasi-norm is that we can use a smaller measurement matrix for signal recovery (at least for the exact case). Chartrand and Staneva [41] showed that ℓ_q quasi-norm minimization is a stable recovery scheme for a certain matrix $A \in \mathbb{R}^{m \times n}$ with $m = \Omega(s + qs \log(n/s))$. Hence, as $q \to 0$, the required dimension of the measurement vector scales like $O(s)$.

In order to implement the ℓ_q quasi-norm minimization for $q < 1$, several heuristic methods have been proposed, including the gradient method [39], the iteratively reweighed least squares method (IRLS) [42], and an operator-splitting algorithm [40], and a method that computes a local minimum [68].

Recently, the sum-of-squares method has been found to be useful in several unsupervised learning tasks such as the planted sparse vector problem [17], dictionary learning [18], and tensor decompositions [18], [19], [69], [161]. Surveys [47], [142] provide detailed information on the Lasserre and other hierarchies. A handbook [10] contains various articles on the semidefinite programming, conic programming, and the SoS method, from both the theoretical and the practical point of view.

10.2 Preliminaries

Notation We write $P \succeq 0$ if P is a sum-of-squares polynomial, and similarly we write $P \succeq Q$ if $P - Q \succeq 0$.

We now see several useful lemmas for pseudodistributions. Most of these results were proved in the series of works by Barak and his coauthors [16]–[18]. We basically follow the terminology and notations in the recent survey by Barak and Steurer [15].

Lemma 10.2.1 (Pseudo Cauchy-Schwarz inequality [16]). *Let $d \in \mathbb{Z}_+$ be an integer. Let $P, Q \in \mathbb{R}[\boldsymbol{x}]_d$ be polynomials of degree d and $\mathcal{D}$ be a pseudodistribution of degree at least $2d$. Then, we have $\widetilde{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}} P(\boldsymbol{x})Q(\boldsymbol{x}) \le \left(\widetilde{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}} P(\boldsymbol{x})^2\right)^{1/2} \left(\widetilde{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}} Q(\boldsymbol{x})^2\right)^{1/2}$.*

Lemma 10.2.2 (Function pseudo Cauchy-Schwarz inequality [16]). *Let $d \in \mathbb{Z}_+$ be an integer. Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be vectors whose entries are polynomials in $\mathbb{R}[\boldsymbol{x}]_d$. Then for any pseudodistribution $\mathcal{D}$ of degree at least $2d$, we have $\widetilde{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}}\langle \boldsymbol{u}, \boldsymbol{v}\rangle \le \left(\widetilde{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}}\|\boldsymbol{u}\|_2^2\right)^{1/2} \left(\widetilde{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}}\|\boldsymbol{v}\|_2^2\right)^{1/2}$.*

Lemma 10.2.3 (Triangle inequality for pseudo ℓ^2-norm [16]). *Let $\boldsymbol{u}$, $\boldsymbol{v}$, and $\mathcal{D}$ be the same as above. Then, we have $\left(\widetilde{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}}\|\boldsymbol{u}+\boldsymbol{v}\|_2^2\right)^{1/2} \le \left(\widetilde{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}}\|\boldsymbol{u}\|_2^2\right)^{1/2} + \left(\widetilde{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}}\|\boldsymbol{v}\|_2^2\right)^{1/2}$.*

Lemma 10.2.4. *For any pseudodistribution $\mathcal{D}$ of degree at least 4, we have*

$$\widetilde{\mathbf{E}}_{(\boldsymbol{u},\boldsymbol{v})\sim\mathcal{D}}\langle \boldsymbol{u}, \boldsymbol{v}\rangle^2 \le \widetilde{\mathbf{E}}_{(\boldsymbol{u},\boldsymbol{v})\sim\mathcal{D}}\|\boldsymbol{u}\|_2^2\|\boldsymbol{v}\|_2^2.$$

Proof. By the Lagrange identity, we have

$$\|\boldsymbol{u}\|_2^2\|\boldsymbol{v}\|_2^2 - \langle \boldsymbol{u}, \boldsymbol{v}\rangle^2 = \sum_{i<j}(u(i)v(j) - u(j)v(i))^2.$$

Hence, the statement has an SoS proof of degree 4. □

10.3 Pseudo Robust Null Space Property

In this section, we prove Theorem 10.1.7. The formal statement of Theorem 10.1.7 is provided in Theorem 10.3.8. In the rest of this section, we fix $q \in (0, 1]$ and an error parameter $\epsilon > 0$. We also fix a matrix $A \in \mathbb{R}^{m\times n}$ and the unknown signal vector $\boldsymbol{x} \in [-1, 1]^n$, and hence the measurement vector $\boldsymbol{y} = A\boldsymbol{x}$.

Let $\delta > 0$ be a parameter chosen later, and L be a set of multiples of δ in $[-1, 1]$. Clearly $|L| = O(1/\delta)$, and there exists $\bar{x} \in L$ with $|x - \bar{x}| \le \delta$ for any $x \in [-1, 1]$. Let $\boldsymbol{x}^L \in \mathbb{R}^n$ be the point in L^n closest to $\boldsymbol{x}$. Our strategy is recovering $\boldsymbol{x}^L$ instead of $\boldsymbol{x}$. As we know that $\boldsymbol{x}^L \in L^n$ and each coordinate of $\boldsymbol{x}^L$ is discretized, we can add several useful constraints to the moment relaxation, such as triangle inequality.

For the measurement vector $\boldsymbol{y} = A\boldsymbol{x}$, we have

$$\|A\boldsymbol{x}^L - \boldsymbol{y}\|_2 = \|A(\boldsymbol{x}^L - \boldsymbol{x})\|_2 \le \sigma_{\max}(A)\|\boldsymbol{x}^L - \boldsymbol{x}\|_2 \le \sigma_{\max}(A)\sqrt{s}\delta.$$

To recover a vector close to $\boldsymbol{x}^L$ in ℓ_q quasi-norm from $\boldsymbol{y}$, we consider the following non-convex program:

$$\text{minimize } \|\boldsymbol{z}\|_q^q \quad \text{subject to } \|A\boldsymbol{z} - \boldsymbol{y}\|_2^2 \le \eta^2, \tag{$P_{q,\eta}$}$$

where $\eta = \sigma_{\max}(A)\sqrt{s}\delta$. Note that the desired vector $\boldsymbol{x}^L$ is a feasible solution of this program.

In Section 10.3.1, we show how to formulate the non-convex program ($P_{q,\eta}$) as a POP. Although the form of the POP depends on ϵ, we can solve its moment relaxation using the SoS method. Then, we show that, if A satisfies a certain property, which we call pseudo robust null space property, then the vector $\boldsymbol{z}$ "sampled" from the pseudodistribution obtained by the SoS method is close to the vector $\boldsymbol{x}^L$ in ℓ_q quasi-norm. Then in Section 10.3.2, we show how to round the pseudodistribution to an actual vector $\bar{\boldsymbol{z}}$ so that $\|\bar{\boldsymbol{z}} - \boldsymbol{x}\|_q = O(\sigma_s(\boldsymbol{x})_q) + \epsilon$ for small $\epsilon > 0$.

10.3.1 Formulating ℓ_q Quasi-Norm Minimization as a POP

To formulate ($P_{q,\eta}$) as a POP, we introduce additional variables. Formally, we define $\|\boldsymbol{z}\|_q^q$ as a shorthand for $\sum_{i=1}^n t_{z(i)}$, where $t_{z(i)}$ for $i \in [n]$ is an additional variable corresponding to $|z(i)|^q$. We also impose constraints $t_{z(i)} \geq 0$ and $t_{z(i)}^{2/q} = z(i)^2$ for each $i \in [n]$.

Unfortunately, these constraints are not sufficient to derive formula connecting $\widetilde{\mathbf{E}}\, t_{z(i)}$ and $z(i)$ in the SoS proof system because $\widetilde{\mathbf{E}}\, t_{z(i)}$ and $\widetilde{\mathbf{E}}\, t_{z(i)}^{2/q}$ are unrelated in general. To address this issue, we add several other variables and constraints. Specifically, for each $i \in [n]$ and $b \in L$, we add a new variable $t_{z(i)-b}$ along with the following constraints:

- $t_{z(i)-b} \geq 0$,
- $t_{z(i)-b}^{2/q} = (z(i) - b)^2$,
- $t_{z(i)-b} \leq t_{z(i)-b'} + |b - b'|^q \quad (b' \in L)$,
- $|b - b'|^q \leq t_{z(i)-b} + t_{z(i)-b'} \quad (b' \in L)$.

Intuitively, $t_{z(i)-b}$ represents $|z(i) - b|^q$. The last two constraints encode triangle inequalities; for example, the first one corresponds to $|z(i) - b|^q \leq |z(i) - b'|^q + |b - b'|^q$. This increases the number of variables and constraints by $O(n/\delta)$ and $O(n/\delta^2)$, respectively. For the sake of notational simplicity, we will denote $t_{z(i)-b}$ by $|z(i) - b|^q$ for $i \in [n]$ and $b \in L$ in POPs. Note that these constraints are valid in the sense that the desired solution $\boldsymbol{x}^L$ satisfies all of them.

Thus we arrive at the following POP:

$$\text{minimize } \|\boldsymbol{z}\|_q^q \quad \text{subject to } \boldsymbol{z} \text{ satisfies the system } \{\|A\boldsymbol{z} - \boldsymbol{y}\|_2^2 \leq \eta^2\} \cup \mathcal{P},$$

where $\mathcal{P}$ is a system of constraints. Here, $\mathcal{P}$ contains all the constraints on $t_{z(i)-b}$ mentioned above but not limited to. Indeed, in Section 10.4, we add other (valid) constraints to $\mathcal{P}$ so that the matrix A satisfies the pseudo robust null space property. From now on, we also fix $\mathcal{P}$.

Now, we consider the following moment relaxation:

$$\text{minimize } \widetilde{\mathbf{E}}_{\mathcal{D}}\|\boldsymbol{z}\|_q^q \quad \text{subject to } \mathcal{D} \text{ satisfies the system } \{\|A\boldsymbol{z} - \boldsymbol{y}\|_2^2 \leq \eta^2\} \cup \mathcal{P}, \qquad (\widetilde{P}_{q,\eta})$$

Proposition 10.3.1. *Let $n_{\mathcal{P}}$ be the number of variables in $\mathcal{P}$ and $d_{\mathcal{P}}$ be the maximum degree that appears in the system $\mathcal{P}$. If there is a polynomial-time separation oracle for constraints in $\mathcal{P}$, then we can solve the moment relaxation ($\widetilde{P}_{q,\eta}$) in $O(n/\delta + n_{\mathcal{P}})^{O(2/q+d_{\mathcal{P}})}$ time.*

Proof. Immediate from Theorem 10.1.4. □

Since $\boldsymbol{x}^L \in L^n$, the value $\widetilde{\mathbf{E}}\|\boldsymbol{z} - \boldsymbol{x}^L\|_q^q$ is well defined. Furthermore, $\mathcal{D}$ satisfies the following triangle inequalities:

$$\begin{aligned}
\widetilde{\mathbf{E}}\|\boldsymbol{z}\|_q^q &\leq \widetilde{\mathbf{E}}\|\boldsymbol{z} - \boldsymbol{x}^L\|_q^q + \|\boldsymbol{x}^L\|_q^q, \\
\widetilde{\mathbf{E}}\|\boldsymbol{z} - \boldsymbol{x}^L\|_q^q &\leq \widetilde{\mathbf{E}}\|\boldsymbol{z}\|_q^q + \|\boldsymbol{x}^L\|_q^q, \\
\|\boldsymbol{x}^L\|_q^q &\leq \widetilde{\mathbf{E}}\|\boldsymbol{z} - \boldsymbol{x}^L\|_q^q + \widetilde{\mathbf{E}}\|\boldsymbol{z}\|_q^q.
\end{aligned}$$

Now we formally define pseudo robust null space property.

Definition 10.3.2. Let $\rho, \tau > 0$ and $s \in \mathbb{Z}_+$. A matrix $A \in \mathbb{R}^{m \times n}$ is said to satisfy the (ρ, τ, s) *pseudo robust null space property* in ℓ_q quasi-norm (ℓ_q-(ρ, τ, s)-PRNSP for short) with respect to the system $\mathcal{P}$ if

$$\widetilde{\mathbf{E}}_{\mathcal{D}} \|\boldsymbol{v}_S\|_q^q \leq \rho \, \widetilde{\mathbf{E}}_{\mathcal{D}} \|\boldsymbol{v}_{\overline{S}}\|_q^q + \tau \left(\widetilde{\mathbf{E}}_{\mathcal{D}} \|A\boldsymbol{v}\|_2^2 \right)^{q/2}$$

holds for any pseudodistribution $\mathcal{D}$ of degree at least $2/q$ satisfying $\mathcal{P}$ and all $\boldsymbol{v}$ of the form $\boldsymbol{z} - \boldsymbol{b}$ ($\boldsymbol{b} \in L^n$), and for any $S \subseteq [n]$ with $|S| = s$.

Lemma 10.3.3. *Let $0 < \rho < 1$, $\tau > 0$, and $s \in \mathbb{Z}_+$. If the matrix $A \in \mathbb{R}^{m \times n}$ satisfies the ℓ_q-(ρ, τ, s)-PRNSP with respect to the system $\mathcal{P}$, then we have*

$$\widetilde{\mathbf{E}} \|\boldsymbol{z} - \boldsymbol{x}^L\|_q^q \leq \frac{1+\rho}{1-\rho} \left(\widetilde{\mathbf{E}} \|\boldsymbol{z}\|_q^q - \|\boldsymbol{x}^L\|_q^q + 2\|\boldsymbol{x}_{\overline{S}}^L\|_q^q \right) + \frac{2\tau}{1-\rho} \left(\widetilde{\mathbf{E}} \|A(\boldsymbol{z} - \boldsymbol{x}^L)\|_2^2 \right)^{q/2}$$

for any $S \subseteq [n]$ with $|S| \leq s$.

Proof. Let $\boldsymbol{v} := \boldsymbol{z} - \boldsymbol{x}^L$. Then from the PRNSP of A, we have

$$\widetilde{\mathbf{E}} \|\boldsymbol{v}_S\|_q^q \leq \rho \, \widetilde{\mathbf{E}} \|\boldsymbol{v}_{\overline{S}}\|_q^q + \tau \left(\widetilde{\mathbf{E}} \|A\boldsymbol{v}\|_2^2 \right)^{q/2}. \tag{10.3}$$

Note that

$$\|\boldsymbol{x}^L\|_q^q = \|\boldsymbol{x}_S^L\|_q^q + \|\boldsymbol{x}_{\overline{S}}^L\|_q^q \leq \widetilde{\mathbf{E}} \|(\boldsymbol{z} - \boldsymbol{x}^L)_S\|_q^q + \widetilde{\mathbf{E}} \|\boldsymbol{z}_S\|_q^q + \|\boldsymbol{x}_{\overline{S}}^L\|_q^q = \widetilde{\mathbf{E}} \|\boldsymbol{v}_S\|_q^q + \widetilde{\mathbf{E}} \|\boldsymbol{z}_S\|_q^q + \|\boldsymbol{x}_{\overline{S}}^L\|_q^q,$$
$$\widetilde{\mathbf{E}} \|\boldsymbol{v}_{\overline{S}}\|_q^q = \widetilde{\mathbf{E}} \|(\boldsymbol{z} - \boldsymbol{x}^L)_{\overline{S}}\|_q^q \leq \|\boldsymbol{x}_{\overline{S}}^L\|_q^q + \widetilde{\mathbf{E}} \|\boldsymbol{z}_{\overline{S}}\|_q^q.$$

Summing up these two inequalities, we get

$$\widetilde{\mathbf{E}} \|\boldsymbol{v}_{\overline{S}}\|_q^q \leq \widetilde{\mathbf{E}} \|\boldsymbol{z}\|_q^q - \|\boldsymbol{x}^L\|_q^q + \widetilde{\mathbf{E}} \|\boldsymbol{v}_S\|_q^q + 2\|\boldsymbol{x}_{\overline{S}}^L\|_q^q. \tag{10.4}$$

Combining (10.3) and (10.4), we get

$$\widetilde{\mathbf{E}} \|\boldsymbol{v}_{\overline{S}}\|_q^q \leq \frac{1}{1-\rho} \left(\widetilde{\mathbf{E}} \|\boldsymbol{z}\|_q^q - \|\boldsymbol{x}^L\|_q^q + 2\|\boldsymbol{x}_{\overline{S}}^L\|_q^q + \tau \left(\widetilde{\mathbf{E}} \|A\boldsymbol{v}\|_2^2 \right)^{q/2} \right).$$

Therefore,

$$\begin{aligned}
\widetilde{\mathbf{E}} \|\boldsymbol{v}\|_q^q &= \widetilde{\mathbf{E}} \|\boldsymbol{v}_S\|_q^q + \widetilde{\mathbf{E}} \|\boldsymbol{v}_{\overline{S}}\|_q^q \\
&\leq (1+\rho) \, \widetilde{\mathbf{E}} \|\boldsymbol{v}_{\overline{S}}\|_q^q + \tau \left(\widetilde{\mathbf{E}} \|A\boldsymbol{v}\|_2^2 \right)^{q/2} \\
&\leq \frac{1+\rho}{1-\rho} \left(\widetilde{\mathbf{E}} \|\boldsymbol{z}\|_q^q - \|\boldsymbol{x}^L\|_q^q + 2\|\boldsymbol{x}_{\overline{S}}^L\|_q^q \right) + \frac{2\tau}{1-\rho} \left(\widetilde{\mathbf{E}} \|A\boldsymbol{v}\|_2^2 \right)^{q/2}.
\end{aligned}$$

□

Corollary 10.3.4. *Let $0 < \rho < 1$, $\tau > 0$, and $s \in \mathbb{Z}_+$. If the matrix $A \in \mathbb{R}^{m \times n}$ satisfies the ℓ_q-(ρ, τ, s)-PRNSP with respect to the system $\mathcal{P}$, then we have*

$$\widetilde{\mathbf{E}} \|\boldsymbol{z} - \boldsymbol{x}^L\|_q^q \leq \frac{2(1+\rho)}{1-\rho} \sigma_s(\boldsymbol{x}^L)_q^q + \frac{2^{1+q}\tau}{1-\rho} \eta^q.$$

Proof. Note that $\widetilde{\mathbf{E}} \|\boldsymbol{z}\|_q^q - \|\boldsymbol{x}^L\|_q^q \leq 0$ and $\left(\widetilde{\mathbf{E}} \|A(\boldsymbol{z} - \boldsymbol{x}^L)\|_2^2 \right)^{q/2} \leq \left(\widetilde{\mathbf{E}} \|A\boldsymbol{z} - \boldsymbol{y}\|_2^2 \right)^{q/2} + \left(\|\boldsymbol{y} - A\boldsymbol{x}^L\|_2^2 \right)^{q/2} \leq 2\eta^q$ by Lemma 10.2.3 and the concavity of $|\cdot|^q$. The statement is immediate from the previous lemma with S being an index set of top-s largest absolute entries of $\boldsymbol{x}$. □

10.3.2 Rounding

Suppose that we have obtained a pseudodistribution $\mathcal{D}$ as a solution to ($\widetilde{P}_{q,\eta}$). Then, we use the following simple rounding scheme to extract an actual vector $\bar{\boldsymbol{z}} \in \mathbb{R}^n$: For each $i \in [n]$, find $b \in L$ that minimizes $\widetilde{\mathbf{E}}_{\boldsymbol{z}\sim\mathcal{D}}|z(i) - b|^q$ and set $\bar{z}(i) = b$. In other words, $\bar{\boldsymbol{z}}$ is a minimizer of $\widetilde{\mathbf{E}}_{\boldsymbol{z}\sim\mathcal{D}}\|\boldsymbol{z} - \boldsymbol{b}\|_q^q$, where $\boldsymbol{b}$ is over vectors in L^n. Clearly, we can find $\bar{\boldsymbol{z}}$ in $O(n/\delta)$ time. Also, we have the following guarantee on its distance to $\boldsymbol{x}^L$.

Lemma 10.3.5. *Let $0 < \rho < 1$, $\tau > 0$, and $s \in \mathbb{Z}_+$. If the matrix $A \in \mathbb{R}^{m\times n}$ satisfies the ℓ_q-(ρ, τ, s)-PRNSP with respect to the system $\mathcal{P}$, then the vector $\bar{\boldsymbol{z}} \in L^n$ satisfies*

$$\|\bar{\boldsymbol{z}} - \boldsymbol{x}^L\|_q^q \le 2\left(\frac{2(1+\rho)}{1-\rho}\sigma_s(\boldsymbol{x}^L)_q^q + \frac{2^{1+q}\tau}{1-\rho}\eta^q\right). \tag{10.5}$$

Proof. By Corollary 10.3.4, we have

$$\widetilde{\mathbf{E}}\|\boldsymbol{z} - \boldsymbol{x}^L\|_q^q \le \frac{2(1+\rho)}{1-\rho}\sigma_s(\boldsymbol{x}^L)_q^q + \frac{2^{1+q}\tau}{1-\rho}\eta^q.$$

Since $\boldsymbol{x}^L \in L^n$ holds and $\bar{\boldsymbol{z}}$ is a minimizer of $\widetilde{\mathbf{E}}\|\boldsymbol{z} - \boldsymbol{b}\|_q^q$, we also have

$$\widetilde{\mathbf{E}}\|\boldsymbol{z} - \bar{\boldsymbol{z}}\|_q^q \le \frac{2(1+\rho)}{1-\rho}\sigma_s(\boldsymbol{x}^L)_q^q + \frac{2^{1+q}\tau}{1-\rho}\eta^q.$$

Then by the triangle inequality,

$$\|\bar{\boldsymbol{z}} - \boldsymbol{x}^L\|_q^q \le \widetilde{\mathbf{E}}\|\boldsymbol{z} - \bar{\boldsymbol{z}}\|_q^q + \widetilde{\mathbf{E}}\|\boldsymbol{z} - \boldsymbol{x}^L\|_q^q \le 2\left(\frac{2(1+\rho)}{1-\rho}\sigma_s(\boldsymbol{x}^L)_q^q + \frac{2^{1+q}\tau}{1-\rho}\eta^q\right). \qquad \square$$

Then, we show that $\bar{\boldsymbol{z}}$ is close to the signal vector $\boldsymbol{x}$ in ℓ_q^q metric.

Lemma 10.3.6. *Let $0 < \rho < 1$, $\tau > 0$, and $s \in \mathbb{Z}_+$. If the matrix $A \in \mathbb{R}^{m\times n}$ satisfies the ℓ_q-(ρ, τ, s)-PRNSP with respect to the system $\mathcal{P}$, then the vector $\bar{\boldsymbol{z}}$ satisfies*

$$\|\bar{\boldsymbol{z}} - \boldsymbol{x}\|_q^q \le \frac{4(1+\rho)}{1-\rho}\sigma_s(\boldsymbol{x})_q^q + O(\epsilon)$$

by choosing

$$\delta = \min\left\{\frac{((1-\rho)\epsilon/\tau)^{1/q}}{\sigma_{\max}(A)\sqrt{s}}, ((1-\rho)\epsilon/n)^{1/q}\right\}.$$

Proof. From the choice of δ, we have $\eta = \sigma_{\max}(A)\sqrt{s}\delta \le (\epsilon(1-\rho)/\tau)^{1/q}$. Then from Lemma 10.3.5, we get

$$\|\bar{\boldsymbol{z}} - \boldsymbol{x}^L\|_q^q \le \frac{4(1+\rho)}{1-\rho}\sigma_s(\boldsymbol{x}^L)_q^q + O(\epsilon).$$

Note that we have $\|\boldsymbol{x}^L - \boldsymbol{x}\|_q^q \le n\delta^q$ and $|\sigma_s(\boldsymbol{x})_q^q - \sigma_s(\boldsymbol{x}^L)_q^q| = O(n\delta^q)$. Then, we have

$$\begin{aligned}\|\bar{\boldsymbol{z}} - \boldsymbol{x}\|_q^q &\le \|\bar{\boldsymbol{z}} - \boldsymbol{x}^L\|_q^q + \|\boldsymbol{x}^L - \boldsymbol{x}\|_q^q \le \frac{4(1+\rho)}{1-\rho}(\sigma_s(\boldsymbol{x})_q^q + n\delta^q) + O(\epsilon) + n\delta^q \\ &= \frac{4(1+\rho)}{1-\rho}\sigma_s(\boldsymbol{x})_q^q + O\Big(\epsilon + \frac{n\delta^q}{1-\rho}\Big) = \frac{4(1+\rho)}{1-\rho}\sigma_s(\boldsymbol{x})_q^q + O(\epsilon).\end{aligned}$$

$\square$

Finally, we show that $\bar{\boldsymbol{z}}$ is close to $\boldsymbol{x}$ in ℓ_q quasi-norm.

Corollary 10.3.7. *Let $0 < \rho < 1$, $\tau > 0$, and $s \in \mathbb{Z}_+$. If the matrix $A \in \mathbb{R}^{m \times n}$ satisfies the ℓ_q-(ρ, τ, s)-PRNSP with respect to the system $\mathcal{P}$, then the vector $\bar{\boldsymbol{z}}$ satisfies*

$$\|\bar{\boldsymbol{z}} - \boldsymbol{x}\|_q \leq \frac{4(1+\rho)}{1-\rho}\sigma_s(\boldsymbol{x})_q + O(\epsilon)$$

by choosing

$$\delta = O\left(\left(\frac{\epsilon(1-\rho)^{1/q+1}}{\tau n^{1/q}}\right)^{1/q} \sigma_{\max}(A)^{-1} s^{-1/2}\right).$$

Proof. Invoke Lemma 10.3.6 with

$$\epsilon' = \epsilon \cdot \left(\frac{1-\rho}{4(1+\rho)\sigma_s(\boldsymbol{x})_q^q}\right)^{1/q-1} = \Omega\left(\epsilon\left(\frac{1-\rho}{n}\right)^{1/q}\right).$$

Then, we get a vector $\bar{\boldsymbol{z}}$ such that

$$\begin{aligned}
\|\bar{\boldsymbol{z}} - \boldsymbol{x}\|_q &\leq \left(\frac{4(1+\rho)}{1-\rho}\sigma_s(\boldsymbol{x})_q^q + \epsilon'\right)^{1/q} \\
&\leq \frac{4(1+\rho)}{1-\rho}\sigma_s(\boldsymbol{x})_q + O\left(\epsilon'\left(\frac{4(1+\rho)}{1-\rho}\sigma_s(\boldsymbol{x})_q^q\right)^{1/q-1}\right) \\
&\leq \frac{4(1+\rho)}{1-\rho}\sigma_s(\boldsymbol{x})_q + O(\epsilon).
\end{aligned}$$

Note that the required δ is

$$\begin{aligned}
&\Omega\left(\min\left\{\left(\frac{(1-\rho)\epsilon'}{\tau}\right)^{1/q} \sigma_{\max}(A)^{-1} s^{-1/2}, \left(\frac{(1-\rho)\epsilon'}{n}\right)^{1/q}\right\}\right) \\
&= \Omega\left(\left(\frac{\epsilon(1-\rho)^{1/q+1}}{\tau n^{1/q}}\right)^{1/q} \sigma_{\max}(A)^{-1} s^{-1/2}\right).
\end{aligned}$$

□

Combining Proposition 10.3.1 and Corollary 10.3.7, we get the following theorem.

Theorem 10.3.8 (PRNSP ⇒ stable signal recovery, the formal version of Theorem 10.1.7)**.** *Let $0 < \rho < 1$, $\tau > 0$, and $s \in \mathbb{Z}_+$. If the matrix $A \in \mathbb{R}^{m \times n}$ satisfies the ℓ_q-(ρ, τ, s)-PRNSP with respect to the system $\mathcal{P}$ and $\mathcal{P}$ has a polynomial-time separation oracle, then we can compute a vector $\bar{\boldsymbol{z}} \in \mathbb{R}^n$ such that*

$$\|\bar{\boldsymbol{z}} - \boldsymbol{x}\|_q \leq \frac{4(1+\rho)}{1-\rho}\sigma_s(\boldsymbol{x})_q + \epsilon.$$

The running time is $(n/\delta + n_{\mathcal{P}})^{O(2/q+d_{\mathcal{P}})}$, where $n_{\mathcal{P}}$ is the number of variables in $\mathcal{P}$, $d_{\mathcal{P}}$ is the maximum degree that appears in constraints of $\mathcal{P}$, and $\delta = \Omega\left(\left(\frac{\epsilon(1-\rho)^{1/q+1}}{\tau n^{1/q}}\right)^{1/q} \sigma_{\max}(A)^{-1} s^{-1/2}\right)$.

10.4 Imposing PRNSP

A *normalized Rademacher matrix* $A \in \mathbb{R}^{m \times n}$ is a random matrix whose each entry is i.i.d. sampled from $\{\pm 1/\sqrt{m}\}$. In this section, we show that we can impose the PRNSP to the normalized Rademacher matrix by introducing some additional variables and constraints to pseudodistributions.

The following notion is important in our analysis.

Definition 10.4.1 (Coherence)**.** Let $A \in \mathbb{R}^{m\times n}$ be a matrix whose columns $\boldsymbol{a}_1, \dots, \boldsymbol{a}_n$ are ℓ^2-normalized. The *ℓ_q-coherence function* of A is defined as

$$\mu_q^q(s) := \max_{i\in[n]} \max \left\{ \sum_{j\in S} |\langle \boldsymbol{a}_i, \boldsymbol{a}_j\rangle|^q : S \subseteq [n], |S| = s, i \notin S \right\}$$

for $s = 1, \dots, n-1$.

Now we review properties of a normalized Rademacher matrix.

Lemma 10.4.2 ([2])**.** *Let $n \in \mathbb{Z}_+$ and $\epsilon > 0$. If $m = \Omega(\log n/\epsilon^2)$, then a normalized Rademacher matrix $A \in \mathbb{R}^{m\times n}$ with columns $\boldsymbol{a}_1, \dots, \boldsymbol{a}_n$ satisfies $|\langle \boldsymbol{a}_i, \boldsymbol{a}_j\rangle| \le \epsilon$ for all $i \neq j$ with high probability.*

Corollary 10.4.3. *Let $s, n \in \mathbb{Z}_+$, $\epsilon > 0$. If $m = \Omega((s/\epsilon)^{2/q} \log n)$, then a normalized Rademacher matrix $A \in \mathbb{R}^{m\times n}$ with columns $\boldsymbol{a}_1, \dots, \boldsymbol{a}_n$ satisfies $\mu_q^q(s) \le \epsilon$ with high probability.*

Lemma 10.4.4 ([148])**.** *A normalized Rademacher matrix $A \in \mathbb{R}^{m\times n}$ with $m \le n$ satisfies $\sigma_{\max}(A) = O(\frac{n\log m}{m})$.*

In what follows, we fix $q \in (0, 1]$, $s \in \mathbb{Z}_+$, $\epsilon > 0$, and a matrix $A \in \mathbb{R}^{m\times n}$ that satisfies the properties of Corollary 10.4.3 and Lemma 10.4.4. Let $\boldsymbol{a}_1, \dots, \boldsymbol{a}_n$ be the columns of A.

For $\boldsymbol{b} \in L^n$, we define $\boldsymbol{v}^{\boldsymbol{b}} = \boldsymbol{z} - \boldsymbol{b}$, where $\boldsymbol{z}$ is a variable in POP. To impose the PRNSP to A, for each $i \in [n]$ and $\boldsymbol{b} \in L^n$, we add a variable $t_{\langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle}$ and constraints $t^{2/q}_{\langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle} = \langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle^2$ and $t_{\langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle} \ge 0$. Intuitively, $t_{\langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle}$ represents $|\langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle|^q$, and as with the previous section, we will denote $t_{\langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle}$ by $|\langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle|^q$ in POPs for notational simplicity.

Furthermore, we add the following constraints for each $i \in [n]$ and $\boldsymbol{b} \in L^n$,

$$|v^{\boldsymbol{b}}(i)|^q \le |\langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle|^q + \sum_{j\neq i} |v^{\boldsymbol{b}}(j)|^q |\langle \boldsymbol{a}_j, \boldsymbol{a}_i\rangle|^q. \tag{10.6}$$

These constraints are valid since $v^{\boldsymbol{b}}(i)\boldsymbol{a}_i = A\boldsymbol{v}^{\boldsymbol{b}} - \sum_{j\neq i} v^{\boldsymbol{b}}(j)\boldsymbol{a}_j$ and $\langle \boldsymbol{a}_i, \boldsymbol{a}_i\rangle = 1$.

At first sight, the resulting moment relaxation is intractable since there are exponentially many new variables. However, this is not the case. To see this, note that

$$\langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle = \sum_{j=1}^n z(i)\langle \boldsymbol{a}_j, \boldsymbol{a}_i\rangle - \sum_{j=1}^n b(j)\langle \boldsymbol{a}_j, \boldsymbol{a}_i\rangle.$$

Also, the number of possible values of $\sum_{j=1}^n b(j)\langle \boldsymbol{a}_j, \boldsymbol{a}_i\rangle$ is $O(nm/\delta)$ because A is $\pm 1/\sqrt{m}$-valued and $b(j) \in [-1, 1]$ is a multiple of δ. Hence, identifying variables $t_{\langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle}$ and $t_{\langle A\boldsymbol{v}_{\boldsymbol{b}'}, \boldsymbol{a}_i\rangle}$ if $\sum_{j=1}^n b(j)\langle \boldsymbol{a}_j, \boldsymbol{a}_i\rangle = \sum_{j=1}^n b'(j)\langle \boldsymbol{a}_j, \boldsymbol{a}_i\rangle$, we only have to add $O(nm/\delta)$ variables.

Another issue is that we have added exponentially many constraints. The following lemma says that we can design a separation oracle for (10.6), and hence we can use the ellipsoid method to solve SoS relaxations involving these constraints.

Lemma 10.4.5. *There is a polynomial time algorithm that validates whether (10.6) is satisfied by a given pseudodistribution $\mathcal{D}$, and if not, it finds $i \in [n]$ and $\boldsymbol{b} \in L^n$ such that*

$$\widetilde{\mathbf{E}}|v^{\boldsymbol{b}}(i)|^q > \widetilde{\mathbf{E}}|\langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle|^q + \sum_{j\neq i} \widetilde{\mathbf{E}}|v^{\boldsymbol{b}}(j)|^q |\langle \boldsymbol{a}_j, \boldsymbol{a}_i\rangle|^q.$$

Proof. Since there are only polynomially many choices for the values of i, $b(i)$, and $\sum_{j=1}^n b(j)\langle \boldsymbol{a}_j, \boldsymbol{a}_i\rangle$, we can exhaustively check all of them. Hence, in what follows, we fix $i \in [n]$, $b_i := b(i)$, and $c := \sum_{j=1}^n b(j)\langle \boldsymbol{a}_j, \boldsymbol{a}_i\rangle$. Note that, by fixing c, we have also fixed the value of $\widetilde{\mathbf{E}}|\langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle|^q$ to $c' := \widetilde{\mathbf{E}}\,|\langle A\boldsymbol{z}, \boldsymbol{a}_i\rangle - c|^q$.

Now we want to check whether there exists $\boldsymbol{b} \in L^n$ with $b(i) = b_i$ and $\sum_{j=1}^n b(j)\langle \boldsymbol{a}_j, \boldsymbol{a}_i\rangle = c$ such that

$$\widetilde{\mathbf{E}}|z(i) - b_i|^q > c' + \sum_{j\neq i} \widetilde{\mathbf{E}}|z(j) - b(j)|^q |\langle \boldsymbol{a}_j, \boldsymbol{a}_i\rangle|^q.$$

We can solve this problem by dynamic programming using a table $T[\cdot,\cdot]$, where $T[k,a]$ stores the minimum value of $\sum_{j\in[k],j\neq i} \widetilde{\mathbf{E}}|z(j) - b(j)|^q|\langle \boldsymbol{a}_j, \boldsymbol{a}_i\rangle|^q$ conditioned on $\sum_{j\in[k],j\neq i} b(j)\langle \boldsymbol{a}_j, \boldsymbol{a}_i\rangle = a$. Since the size of T is $O(n \cdot nm/\delta)$, the dynamic programming can be solved in polynomial time. □

Let $\mathcal{P}'$ be the system of all the added constraints on $t_{z(i)-b}$ (given in Section 10.3) and constraints on $t_{\langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle}$. Then, A satisfies the PRNSP by imposing the system $\mathcal{P}'$.

Theorem 10.4.6 (The formal version of Theorem 10.1.8)**.** *The matrix A satisfies the ℓ_q-(ρ,τ,s)-PRNSP with respect to $\mathcal{P}'$ with $\rho = \frac{\mu_q^q(s)}{1-\mu_q^q(s-1)} = O(\epsilon)$ and $\tau = \frac{s}{1-\mu_q^q(s-1)} = O(s)$.*

Proof. The proof basically follows the standard proof in compressed sensing (see [64]). Let us fix $S \subseteq [n]$ of size s and $\boldsymbol{b} \in L^n$. Then for any pseudodistribution $\mathcal{D}$ satisfying the above constraints, we have

$$\widetilde{\mathbf{E}}|v^{\boldsymbol{b}}(i)|^q \le \widetilde{\mathbf{E}}|\langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle|^q + \widetilde{\mathbf{E}}\sum_{l\in\bar{S}}|v^{\boldsymbol{b}}(l)|^q|\langle \boldsymbol{a}_l, \boldsymbol{a}_i\rangle|^q + \widetilde{\mathbf{E}}\sum_{j\in S: j\neq i}|v^{\boldsymbol{b}}(j)|^q|\langle \boldsymbol{a}_j, \boldsymbol{a}_i\rangle|^q$$

for $i \in S$. Using the pseudo Cauchy Schwarz $k = -\log_2 q$ times, we get

$$\left(\widetilde{\mathbf{E}}|\langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle|^q\right)^{2/q} \le \widetilde{\mathbf{E}}\langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle^2.$$

By Lemma 10.2.4,

$$\widetilde{\mathbf{E}}\langle A\boldsymbol{v}^{\boldsymbol{b}}, \boldsymbol{a}_i\rangle^2 \le \widetilde{\mathbf{E}}\|A\boldsymbol{v}^{\boldsymbol{b}}\|_2^2\|\boldsymbol{a}_i\|_2^2 = \widetilde{\mathbf{E}}\|A\boldsymbol{v}^{\boldsymbol{b}}\|_2^2.$$

Thus we have

$$\widetilde{\mathbf{E}}|v^{\boldsymbol{b}}(i)|^q \le \left(\widetilde{\mathbf{E}}\|A\boldsymbol{v}^{\boldsymbol{b}}\|_2^2\right)^{q/2} + \widetilde{\mathbf{E}}\sum_{l\in\bar{S}}|v^{\boldsymbol{b}}(l)|^q|\langle \boldsymbol{a}_l, \boldsymbol{a}_i\rangle|^q + \widetilde{\mathbf{E}}\sum_{j\in S: j\neq i}|v^{\boldsymbol{b}}(j)|^q|\langle \boldsymbol{a}_j, \boldsymbol{a}_i\rangle|^q$$

Summing up these inequalities yields

$$\begin{aligned}\widetilde{\mathbf{E}}\|\boldsymbol{v}_S^{\boldsymbol{b}}\|_q^q &\le s\left(\widetilde{\mathbf{E}}\|A\boldsymbol{v}^{\boldsymbol{b}}\|_2^2\right)^{q/2} + \sum_{l\in\bar{S}}\widetilde{\mathbf{E}}|v^{\boldsymbol{b}}(l)|^q\sum_{i\in S}|\langle \boldsymbol{a}_l, \boldsymbol{a}_i\rangle|^q + \sum_{j\in S}\widetilde{\mathbf{E}}|v^{\boldsymbol{b}}(j)|^q\sum_{i\in S: i\neq j}|\langle \boldsymbol{a}_j, \boldsymbol{a}_i\rangle|^q \\ &\le s\left(\widetilde{\mathbf{E}}\|A\boldsymbol{v}^{\boldsymbol{b}}\|_2^2\right)^{q/2} + \mu_q^q(s)\,\widetilde{\mathbf{E}}\|\boldsymbol{v}_S^{\boldsymbol{b}}\|_q^q + \mu_q^q(s-1)\,\widetilde{\mathbf{E}}\|\boldsymbol{v}_S^{\boldsymbol{b}}\|_q^q.\end{aligned}$$

Rearranging this inequality finishes the proof. □

Combining Lemma 10.4.4 and Theorem 10.4.6, we immediately have the following.

Corollary 10.4.7. *The matrix A satisfies the ℓ_q-(ρ,τ,s)-PRNSP with respect to $\mathcal{P}'$ with $\rho = O(\epsilon)$ and $\tau = O(s)$. Moreover, $\sigma_{\max}(A) = O((n\log m)/m)$.*

Now we establish our main theorem.

Theorem 10.4.8 (Main theorem, the formal version of Theorem 10.1.2)**.** *Let $q \in (0, 1]$ be a constant of the form $q = 2^{-k}$ for some $k \in \mathbb{Z}_+$, $s, n \in \mathbb{Z}_+$, and $\epsilon > 0$. Then, a normalized Rademacher matrix $A \in \mathbb{R}^{m \times n}$ with $m = O(s^{2/q} \log n)$ satisfies the following property with high probability: For any vector $\boldsymbol{y} = A\boldsymbol{x}$, where $\boldsymbol{x} \in [-1, 1]^n$ is an unknown vector, we can recover $\bar{\boldsymbol{z}}$ with $\|\bar{\boldsymbol{z}} - \boldsymbol{x}\|_q \leq O(\sigma_s(\boldsymbol{x})_q) + \epsilon$. The running time is $(nm/\delta)^{O(2/q)}$, where $\delta = O\Big(m(\frac{\epsilon}{sn^{1/q}})^{1/q}/(\sqrt{s}n \log m)\Big)$.*

Proof. Solve the SoS relaxation ($\widetilde{P}_{q,\eta}$) with the system $\mathcal{P}'$. Then, with high probability, the matrix A satisfies the ℓ_q-(ρ, τ, s)-PRNSP with respect to $\mathcal{P}'$ for $\rho = O(1)$ and $\tau = O(s)$, and $\sigma_{\max}(A) = O((n \log m)/m)$ with high probability by Corollary 10.4.7. Then, we can recover a vector $\bar{\boldsymbol{z}} \in \mathbb{R}^n$ with the desired property by Theorem 10.3.8.

Note that the number $n_{\mathcal{P}'}$ of added variables in the system $\mathcal{P}'$ is $O(nm/\delta)$ and the maximum degree $d_{\mathcal{P}'}$ that appears in a constraint of $\mathcal{P}'$ is $2/q$. Hence, the running time becomes as stated. □

Chapter 11

Conclusion

In this dissertation, we have considered the three problems in machine learning and communication: monotone submodular function maximization, matrix completion, and compressed sensing. We now conclude this dissertation with a brief summary of our contribution and several open problems.

Submodularity over the Integer Lattice and Budget Allocation (Sections 3.3–3.5): We presented a novel generalization of the previous models in machine learning by exploiting the submodularity over the integer lattice. Our generalized models naturally capture not only the budget allocation problem but also generalized sensor placement and text summarization, in which we allow multiple choices for each element. Further investigation of the potential of submodularity over the integer lattice in machine learning is an obvious future direction.

Submodular Function Maximization over the Integer Lattice (Chapter 4): We devised several approximation algorithms for maximizing a monotone submodular function over the integer lattice under a certain constraint. For DR-submodular functions, we designed polynomial time $(1 - 1/\mathrm{e} - \epsilon)$-approximation algorithms for cardinality, polymatroid, and knapsack constraints. We also described a polynomial time $(1 - 1/\mathrm{e} - \epsilon)$-approximation algorithm for lattice-submodular function maximization with a cardinality constraint. Since our algorithm for maximizing a lattice submodular function subject to a knapsack constraint is a pseudopolynomial time algorithm, the first open problem is to improve the time complexity. Furthermore, maximization of a lattice submodular function subject to a polymatroid constraint is another open problem, because we know almost nothing about it.

The Diminishing Return Submodular Cover Problem (Chapter 5): We designed a bicriteria approximation algorithm for a generalization of submodular cover, namely, the diminishing submodular cover problem (DR-submodular cover). Our algorithm is comparable to the naive greedy algorithm in objective values, but exponentially faster in running time. We do not think that our approximation ratio $\rho\left(1 + \log \frac{d}{\beta}\right)$ is tight for DR-submodular cover in general, although the approximation ratio coincides with the tight log-factor if it is applied for set cover. Especially, up to our knowledge, there is no reason to believe that the linear dependence on a curvature ρ is optimal.

Faster Algorithm for Multicasting in Linear Deterministic Relay Network (Chapter 7): We presented a faster algorithm for finding a multicast code in LDRN using mixed matrix completion technique. Our algorithm achieves a "lower bound"; i.e., the simpler problem of unicast computations for all the sinks already takes the same time complexity and no

multicast algorithms are known to break this lower bound even in a classical wired network. Can we clarify this "lower bound" or can we design a multicast algorithm that is faster than unicast computations for every sinks? Also, as we noted in the last of this chapter, it might be possible to reduce the time complexity without unicast computation by exploiting the structure of the mixed matrices in our algorithm. More generally, designing a sparsity-aware algorithm for mixed matrix completion is an interesting future direction.

The Low-Rank Basis Problem for a Matrix Subspace (Chapter 8): We introduced a new problem, the low-rank basis problem for a matrix subspace and provided a heuristic algorithm for the problem. We pointed out that the low-rank basis problem has various applications in max-rank matrix completion, image separation, accurate eigenvector computation, etc. Although our algorithm works very well in practice, establishing theoretical guarantee of performance is obvious future work. One promising direction is to investigate a special case that a given subspace is spanned by a single low-rank matrix and random Gaussian matrices. Such a case is called the planted model, and indeed, Qu, Sun, and Wright [138] analyzed their algorithm under the planted sparse vector model. However, their analysis is quite complicated and it is nontrivial to generalize their result for matrices. Nonetheless, we are optimistic for giving rigorous analysis to our heuristic methods.

Nonconvex Compressed Sensing with the Sum of Squares Method (Chapter 10): We designed a polynomial time stable signal recovery algorithm in the ℓ_q metric, where $q = 1/2^k \leq 1$. Our algorithm heavily depends on the sum-of-squares (SoS) method. The SoS method is a very powerful relaxation scheme for nonconvex problems, but its potential and limitation are far from being fully understood. Our result shed the light on the potential of the SoS method in compressed sensing. We believe that further investigation of the SoS method in compressed sensing would be fruitful. From the technical point of view, our analysis relies on the coherence argument, but an RIP argument often yields better sample complexity as we saw in Chapter 9. However, known RIP arguments do not seem easy to be converted into an SoS proof. Therefore, the last open problem we pose is; can we find an SoS proof for RIP arguments or is it impossible to prove RIP results in the SoS proof system?

Bibliography

[1] V. Abolghasemi, S. Ferdowsi, and S. Sanei, "Blind separation of image sources via adaptive dictionary learning," *IEEE Transactions on Image Processing*, vol. 21, no. 6, pp. 2921–2930, 2012 (cited on p. 96).

[2] D. Achlioptas, "Database-friendly random projections: Johnson-Lindenstrauss with binary coins," *Journal of Computer and System Sciences*, vol. 66, no. 4, pp. 671–687, 2003 (cited on p. 115).

[3] ACM-SIGKDD and Netflix. (2007). Proceedings of kdd cup and workshop 2007, [Online]. Available: http://www.cs.uic.edu/~liub/KDD-cup-2007/proceedings.html (cited on pp. 3, 63).

[4] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, pp. 1204–1216, 2000 (cited on p. 67).

[5] N. Alon, I. Gamzu, and M. Tennenholtz, "Optimizing budget allocation among channels and influencers," in *Proceedings of the 21st International Conference on World Wide Web (WWW)*, ACM Press, 2012, pp. 381–388 (cited on pp. 1, 2, 18, 19, 48).

[6] A. Amaudruz and C. Fragouli, "Combinatorial algorithms for wireless information flow," in *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2009, pp. 555–564 (cited on p. 65).

[7] B. P. W. Ames and S. A. Vavasis, "Nuclear norm minimization for the planted clique and biclique problems," *Mathematical Programming*, vol. 129, no. 1, Ser. B, pp. 69–89, 2011 (cited on p. 63).

[8] F. Andersson and M. Carlsson, "Alternating projections on nontangential manifolds," *Constructive Approximation*, vol. 38, no. 3, pp. 489–525, 2013 (cited on pp. 89, 92).

[9] J. Andersson and J.-O. Strömberg, "On the theorem of uniform recovery of structured random matrices," *IEEE Transactions on Information Theory*, vol. 60, no. 3, pp. 1700–1710, 2014 (cited on p. 103).

[10] M. F. Anjos and J. B. Lasserre, *Handbook on Semidefinite, Conic and Polynomial Optimization*. Springer Science & Business Media, 2011 (cited on p. 109).

[11] A. S. Avestimehr, S. N. Diggavi, and D. N. C. Tse, "Wireless network information flow: a deterministic approach," *IEEE Transactions on Information Theory*, vol. 57, pp. 1872–1905, 2011 (cited on pp. 3, 65–67, 70).

[12] F. Bach, "Learning with submodular functions: a convex optimization perspective," *Foundations and Trends®in Machine Learning*, vol. 6, no. 2-3, pp. 145–373, 2013 (cited on p. 9).

[13] A. Badanidiyuru and J. Vondrák, "Fast algorithms for maximizing submodular functions," in *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014, pp. 1497–1514 (cited on pp. 1, 2, 28, 29, 51).

[14] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, Eds., *Templates for the solution of algebraic eigenvalue problems*, ser. Software, Environments, and Tools. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000, vol. 11, A practical guide (cited on p. 96).

[15] B. Barak and D. Steurer, "Sum-of-squares proofs and the quest toward optimal algorithms," in *Proceedings of International Congress of Mathematicians (ICM)*, 2014, pp. 509–533 (cited on p. 109).

[16] B. Barak, F. G. S. L. Brandao, A. W. Harrow, J. Kelner, D. Steurer, and Y. Zhou, "Hypercontractivity, sum-of-squares proofs, and their applications," in *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, 2012, pp. 307–326 (cited on pp. 106, 107, 109, 110).

[17] B. Barak, J. A. Kelner, and D. Steurer, "Rounding sum-of-squares relaxations," in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, 2014, pp. 31–40 (cited on pp. 106, 109).

[18] ——, "Dictionary learning and tensor decomposition via the sum-of-squares method," in *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, 2015, pp. 143–151 (cited on pp. 74, 109).

[19] B. Barak and A. Moitra, "Tensor prediction, Rademacher complexity and random 3-XOR," *CoRR*, vol. abs/1501.06521, 2015 (cited on p. 109).

[20] R. Barbosa, A. Ene, H. L. Nguyen, and J. Ward, "The power of randomization: distributed submodular maximization on massive datasets," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015 (cited on p. 1).

[21] A. J. Bell and T. J. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural Computation*, vol. 7, no. 6, pp. 1129–1159, 1995 (cited on p. 96).

[22] R. Berinde, P. Indyk, and M. Ruzic, "Practical near-optimal sparse recovery in the L1 norm," in *Proceedings of the 46th Annual Allerton Conference on Communication, Control, and Computing*, 2008, pp. 198–205 (cited on p. 109).

[23] R. Berinde and P. Indyk, "Sequential sparse matching pursuit," in *Proceedings of the 47th Annual Allerton Conference on Communication, Control, and Computing*, 2009, pp. 36–43 (cited on p. 109).

[24] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Review*, vol. 51, no. 1, pp. 34–81, 2009 (cited on p. 100).

[25] K. Bryan and T. Leise, "Making so with less: an introduction to compressed sensing," *SIAM Review*, vol. 55, no. 3, pp. 547–566, 2013 (cited on p. 100).

[26] N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz, "A tight linear time (1/2)-approximation for unconstrained submodular maximization," in *Proceedings of the IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, 2012, pp. 649–658 (cited on p. 28).

[27] N. Buchbinder and M. Feldman, "Deterministic algorithms for submodular maximization problems," *ArXiv*, 2015 (cited on p. 28).

[28] N. Buchbinder, M. Feldman, J. S. Naor, and R. Schwartz, "Submodular maximization with cardinality constraints," in *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014, pp. 1433–1452 (cited on p. 28).

[29] P. Bühlmann and S. van de Geer, *Statistics for high-dimensional data*, ser. Springer Series in Statistics. Springer, Heidelberg, 2011, Methods, theory and applications (cited on p. 74).

[30] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2008 (cited on pp. 3, 63, 75, 80, 83).

[31] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1740–1766, 2011 (cited on pp. 1, 2, 9, 17).

[32] E. J. Candès, "The restricted isometry property and its implications for compressed sensing," *Comptes Rendus Mathematique*, vol. 346, no. 9-10, pp. 589–592, 2008 (cited on pp. 74, 103, 105, 109).

[33] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, pp. 717–772, 2009 (cited on pp. 3, 63, 64).

[34] E. J. Candès, J. K. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006 (cited on p. 3).

[35] ——, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006 (cited on pp. 4, 101, 102, 109).

[36] E. J. Candès and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005 (cited on pp. 105, 107).

[37] ——, "Near-optimal signal recovery from random projections: universal encoding strategies?" *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5406–5425, 2006 (cited on pp. 74, 103).

[38] ——, "The power of convex relaxation: near-optimal matrix completion," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2053–2080, 2010 (cited on pp. 63, 64).

[39] R. Chartrand, "Exact reconstruction of sparse signals via nonconvex minimization," *Signal Processing Letters*, vol. 14, no. 10, pp. 707–710, (cited on p. 109).

[40] ——, "Fast algorithms for nonconvex compressive sensing: MRI reconstruction from very few data," in *Proceedings of the IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI)*, 2009, pp. 262–265 (cited on p. 109).

[41] R. Chartrand and V. Staneva, "Restricted isometry properties and nonconvex compressive sensing," *Inverse Problems*, vol. 24, no. 3, 2008 (cited on p. 109).

[42] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008, pp. 3869–3872 (cited on p. 109).

[43] C. Chekuri, J. Vondrák, and R. Zenklusen, "Dependent randomized rounding via exchange properties of combinatorial structures," in *Proceedings of IEEE 51st Annual Symposium on Foundations of Computer Science (FOCS)*, 2010, pp. 575–584 (cited on pp. 1, 2, 17, 28).

[44] S. S. Chen, D. L. Donoho, and M. a. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998 (cited on p. 4).

[45] Y. Chen, H. Shioi, C. A. F. Montesinos, L. P. Koh, S. Wich, and A. Krause, "Active detection via adaptive submodularity," in *Proceedings of the 31st International Conference of Machine Learning (ICML)*, 2014 (cited on p. 13).

[46] A. Cherian, S. Sra, and N. Papanikolopoulos, "Denoising sparse noise via online dictionary learning," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2060–2063, 2011 (cited on pp. 4, 105).

[47] E. Chlamtac and M. Tulsiani, "Convex relaxations and integrality gaps," in *Handbook on Semidefinite, Conic and Polynomial Optimization*, 2012 (cited on pp. 106, 109).

[48] A. Cohen, W. Dahmen, and R. DeVore, "Compressed sensing and best k-term approximation," *Journal of the American Mathematical Society*, vol. 22, no. 1, pp. 211–231, 2009 (cited on pp. 101, 104, 109).

[49] T. F. Coleman and A. Pothen, "The null space problem. I. Complexity," *SIAM Journal on Algebraic and Discrete Methods*, vol. 7, no. 4, pp. 527–537, 1986 (cited on pp. 75, 77).

[50] W. H. Cunningham, "Testing membership in matroid polyhedra," *Journal of Combinatorial Theory, Series B*, vol. 188, pp. 161–188, 1984 (cited on pp. 11, 38).

[51] ——, "Improved bounds for matroid partition and intersection algorithms," *SIAM Journal on Computing*, vol. 15, pp. 948–957, 1986 (cited on p. 69).

[52] J. W. Demmel, *Applied numerical linear algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997 (cited on p. 97).

[53] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006 (cited on pp. 3, 100).

[54] J. B. Ebrahimi and C. Fragouli, "Multicasting algorithms for deterministic networks," in *Proceedings of IEEE Information Theory Workshop (ITW)*, 2010, pp. 1–5 (cited on pp. 65, 66).

[55] ——, "Algebraic algorithms for vector network coding," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 996–1007, 2011 (cited on pp. 65–67).

[56] J. Edmonds, "Systems of distinct representatives and linear algebra," *Journal of Research of the National Bureau of Standards*, vol. B71, pp. 241–245, 1967 (cited on pp. 60, 61).

[57] ——, "Matroids and the greedy algorithm," *Mathematical Programming*, vol. 1, no. 1, pp. 127–136, 1971 (cited on pp. 11, 12).

[58] J. Edmonds and R. Giles, "A min-max relation for submodular functions on graphs," *Annals of Discrete Mathematics*, vol. 1, pp. 185–204, 1977 (cited on p. 67).

[59] M. Fazel, "Matrix rank minimization with applications," PhD thesis, Electrical Engineering Deptartment Stanford University, 2002 (cited on p. 74).

[60] M. Fazel, H. Hindi, and S. P. Boyd, "A rank minimization heuristic with application to minimum order system approximation," in *Proceedings of the 2001 American Control Conference*, 2001, pp. 4734–4739 (cited on p. 74).

[61] U. Feige, "A threshold of $\ln n$ for approximating set cover," *Journal of the ACM*, vol. 45, no. 4, pp. 634–652, 1998 (cited on pp. 1, 18, 51).

[62] U. Feige, V. S. Mirrokni, and J. Vondrák, "Maximizing non-monotone submodular functions," *SIAM Jounarl on Computing*, vol. 40, no. 4, pp. 1133–1153, 2011 (cited on p. 28).

[63] S. Foucart, A. Pajor, H. Rauhut, and T. Ullrich, "The gelfand widths of ℓ_p-balls for $0 < p \leq 1$," *Journal of Complexity*, vol. 26, no. 6, pp. 629–640, 2010 (cited on p. 104).

[64] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*. Springer Science+Business Media New York, 2013 (cited on pp. 100–102, 109, 116).

[65] S. Fujishige, "Polymatroidal dependence structure of a set of random variables," *Information and Control*, vol. 39, no. 1, pp. 55–72, 1978 (cited on p. 8).

[66] ——, *Submodular Functions and Optimization*, 2nd. Elsevier, 2005 (cited on pp. 1, 7, 9, 20, 28, 67).

[67] ——, "A note on polylinking flow networks," *Mathematical Programming*, vol. 137, no. 1, pp. 601–607, 2011 (cited on p. 67).

[68] D. Ge, X. Jiang, and Y. Ye, "A note on the complexity of L_p minimization," *Mathematical Programming*, vol. 129, no. 2, pp. 285–299, 2011 (cited on p. 109).

[69] R. Ge and T. Ma, "Decomposing overcomplete 3rd order tensors using sum-of-squares algorithms," *CoRR*, vol. abs/1504.05287, 2015 (cited on p. 109).

[70] J. F. Geelen, "Maximum rank matrix completion," *Linear Algebra and Its Applications*, vol. 288, pp. 211–217, 1999 (cited on pp. 3, 61).

[71] A. Gilbert and P. Indyk, "Sparse recovery using sparse matrices," *Proceedings of the IEEE*, vol. 6, pp. 937–947, 2010 (cited on p. 109).

[72] M. X. Goemans, S. Iwata, and R. Zenklusen, "A flow model based on polylinking system," *Mathematical Programming*, vol. 135, no. 1, pp. 1–23, 2012 (cited on pp. 9, 65–67, 69–71).

[73] G. H. Golub and C. F. Van Loan, *Matrix computations*, 4th, ser. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, 2013 (cited on pp. 80, 83, 85, 96).

[74] C. Gottschalk and B. Peis, "Submodular function maximization on the bounded integer lattice," *ArXiv preprint*, 2015 (cited on p. 28).

[75] M. Grant and S. Boyd. (Mar. 2014). Cvx: Matlab software for disciplined convex programming, version 2.1, [Online]. Available: http://cvxr.com/cvx (cited on p. 83).

[76] D. Grigoriev and N. Vorobjov, "Complexity of null-and positivstellensatz proofs," *Annals of Pure and Applied Logic*, vol. 113, no. 1, pp. 153–160, 2001 (cited on p. 107).

[77] M. Grötschel, L. Lovász, and A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization," *Combinatorica*, vol. 1, no. 2, pp. 169–197, 1981 (cited on p. 9).

[78] L. Gurvits, "Classical complexity and quantum entanglement," *Journal of Computer and System Sciences*, vol. 69, no. 3, pp. 448–484, 2004 (cited on p. 76).

[79] P. Hand and L. Demanet, "Recovering the sparsest element in a subspace," *ArXiv*, 2013 (cited on p. 74).

[80] N. J. A. Harvey, D. R. Karger, and K. Murota, "Deterministic network coding by matrix completion," in *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2005, pp. 489–498 (cited on pp. 3, 61, 62, 65, 68, 69).

[81] N. J. A. Harvey, D. R. Karger, and S. Yekhanin, "The complexity of matrix completion," in *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithm (SODA)*, 2006, pp. 1103–1111 (cited on p. 62).

[82] J. Håstad, "Tensor rank is NP-complete," *Journal of Algorithms*, vol. 11, no. 4, pp. 644–654, 1990 (cited on p. 99).

[83] U. Helmke and M. A. Shayman, "Critical points of matrix least squares distance functions," *Linear Algebra and its Applications*, vol. 215, pp. 1–19, 1995 (cited on p. 89).

[84] C. J. Hillar and L.-H. Lim, "Most tensor problems are NP-hard," *Journal of the ACM*, vol. 60, no. 6, Art. 45, 39, 2013 (cited on p. 99).

[85] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, pp. 164–189, 1927 (cited on p. 99).

[86] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, Oct. 2007 (cited on p. 95).

[87] P. Indyk and M. Ružić, "Near-optimal sparse recovery in the L1 norm," in *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2008, pp. 199–207 (cited on p. 109).

[88] G. Ivanyos, M. Karpinski, Y. Qiao, and M. Santha, "Generalized Wong sequences and their applications to Edmonds' problems," in *Proceedings of the 31st International Symposium on Theoretical Aspects of Computer Science (STACS)*, 2014, pp. 397–408 (cited on p. 76).

[89] G. Ivanyos, M. Karpinski, and N. Saxena, "Deterministic polynomial time algorithms for matrix completion problems," *SIAM Journal on Computing*, vol. 39, no. 8, pp. 3736–3751, 2010 (cited on pp. 3, 62).

[90] S. Iwata, "Submodular function minimization," *Mathematical Programming*, vol. 112, no. 1, pp. 45–64, 2007 (cited on p. 9).

[91] S. Iwata, L. Fleischer, and S. Fujishige, "A combinatorial strongly polynomial algorithm for minimizing submodular functions," *Journal of the ACM*, vol. 48, no. 4, pp. 761–777, 2001 (cited on p. 9).

[92] S. Iwata, S. Tanigawa, and Y. Yoshida, "Bisubmodular function maximization and extensions," *Mathematical Engineering Technical Reports*, no. September, 2013 (cited on p. 28).

[93] R. Iyer and J. Bilmes, "Submodular optimization with submodular cover and submodular knapsack constraints," in *Advances of Neural Information Processing Systems (NIPS)*, 2013, pp. 2436–2444 (cited on pp. 52, 55).

[94] P. Jain, P. Netrapalli, and S. Sanghavi, "Low-rank matrix completion using alternating minimization," in *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, 2013, pp. 665–674 (cited on p. 64).

[95] S. Jegelka, H. Lin, and J. A. Bilmes, "On fast approximate submodular minimization," in *Advances in Neural Information Processing Systems (NIPS)*, 2011, pp. 460–468 (cited on p. 9).

[96] M. Kapralov, I. Post, and J. Vondrák, "Online submodular welfare maximization: greedy is optimal," in *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2012, pp. 1216–1225 (cited on p. 20).

[97] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2003, pp. 137–146 (cited on pp. 1, 8, 13, 14).

[98] M. Kim and M. Médard, "Algebraic network coding approach to deterministic wireless relay networks," in *Proceedings of 48th Annual Allerton Conference on Communication, Control, and Computing*, IEEE, 2010, pp. 1518–1525 (cited on p. 65).

[99] Z. Király and E. R. Kovács, "Randomized and deterministic algorithms for network coding problems in wireless networks," *Information Processing Letters*, vol. 115, no. 4, pp. 507–511, 2015 (cited on p. 67).

[100] C.-W. Ko, J. Lee, and M. Queyranne, "An exact algorithm for maximum entropy sampling," *Operations Research*, vol. 43, no. 4, pp. 684–691, 1995 (cited on p. 1).

[101] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009 (cited on pp. 76, 99).

[102] J. Krarup and P. M. Pruzan, "The simple plant location problem: survey and synthesis," *European Journal of Operational Research*, vol. 12, no. 1, pp. 36–81, 1983 (cited on p. 1).

[103] A. Krause and D. Golovin, "Submodular function maximization," in *Tractability: Practical Approaches to Hard Problems*, Cambridge University Press, 2014, pp. 71–104 (cited on p. 7).

[104] A. Krause and J. Leskovec, "Efficient sensor placement optimization for securing large water distribution networks," *Journal of Water Resources Planning and Management*, vol. 134, no. 6, pp. 516–526, 2008 (cited on p. 14).

[105] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: theory, efficient algorithms and empirical studies," *The Journal of Machine Learning Research*, vol. 9, pp. 235–284, 2008 (cited on pp. 1, 13).

[106] J. N. Laska, M. A. Davenport, and R. G. Baraniuk, "Exact signal recovery from sparsely corrupted measurements through the pursuit of justice," *Conference Record of the 43rd Asilomar Conference on Signals, Systems and Computers (ACSSC)*, pp. 1556–1560, 2009 (cited on pp. 4, 105).

[107] J. B. Lasserre, "Global optimization with polynomials and the problem of moments," *SIAM Journal on Optimization*, vol. 11, no. 3, pp. 796–817, 2006 (cited on pp. 4, 106).

[108] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2007, pp. 420–429 (cited on pp. 56, 57).

[109] A. S. Lewis, D. R. Luke, and J. Malick, "Local linear convergence for alternating and averaged nonconvex projections," *Foundations of Computational Mathematics*, vol. 9, no. 4, pp. 485–513, 2009 (cited on p. 89).

[110] A. S. Lewis and J. Malick, "Alternating projections on manifolds," *Mathematics of Operations Research*, vol. 33, no. 1, pp. 216–234, 2008 (cited on pp. 89, 92).

[111] H. Lin and J. Bilmes, "Multi-document summarization via budgeted maximization of submodular functions," in *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2010, pp. 912–920 (cited on pp. 1, 13, 14, 48).

[112] ——, "A class of submodular functions for document summarization.," in *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2011, pp. 510–520 (cited on pp. 1, 13, 14).

[113] Y.-J. Liu, D. Sun, and K.-C. Toh, "An implementable proximal point algorithmic framework for nuclear norm minimization," *Mathematical Programming*, vol. 133, no. 1-2, Ser. A, pp. 399–436, 2012 (cited on p. 63).

[114] Z. Liu and L. Vandenberghe, "Interior-point method for nuclear norm approximation with application to system identification," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 3, pp. 1235–1256, 2009 (cited on p. 63).

[115] L. Lovász, "On determinants, matchings and random algorithms," *Fundamentals of Computation Theory, FCT*, pp. 565–574, 1979 (cited on pp. 3, 61).

[116] ——, "Submodular functions and convexity," in *Mathematical Programming The State of the Art*, Springer, 1983, pp. 235–257 (cited on p. 8).

[117] ——, "Singular spaces of matrices and their application in combinatorics," *Bulletin of the Brazilian Mathematical Society*, vol. 20, pp. 87–99, 1989 (cited on pp. 62, 76).

[118] D. R. Luke, "Prox-regularity of rank constraints sets and implications for algorithms," *Journal of Mathematical Imaging and Vision*, vol. 47, no. 3, pp. 231–238, 2013 (cited on p. 89).

[119] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993 (cited on p. 4).

[120] P. Milgrom and B. Strulovici, "Substitute goods, auctions, and equilibrium," *Journal of Economic Theory*, vol. 144, no. 1, pp. 212–247, 2009 (cited on p. 20).

[121] M. Minoux, "Accelerated greedy algorithms for maximizing submodular set functions," *Optimization Techniques, Lecture Notes in Control and Information Sciences*, vol. 7, pp. 234–243, 1978 (cited on pp. 16, 48, 56).

[122] V. Mirrokni and M. Zadimoghaddam, "Randomized composable core-sets for distributed submodular maximization," in *Proceedings of the 47th Annual ACM on Symposium on Theory of Computing (STOC)*, 2015, pp. 153–162 (cited on p. 1).

[123] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrak, and A. Krause, "Lazier than lazy greedy," in *Proceedings of AAAI Conference on Artificial Intelligence*, Sep. 2015 (cited on p. 1).

[124] K. Murota, *Discrete Convex Analysis*. Society for Industrial and Applied Mathematics, 2003 (cited on pp. 20, 28).

[125] ——, *Matrices and Matroids for System Analysis*, 2nd. Springer-Verlag, Berlin, 2009 (cited on pp. 60, 62, 67).

[126] K. Murota and M. Iri, "Structural solvability of systems of equations – a mathematical formulation for distinguishing accurate and inaccurate numbers in structural analysis of systems," *Japan Journal of Applied Mathematics*, vol. 2, pp. 247–271, 1985 (cited on p. 60).

[127] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM journal on Computing*, vol. 24, no. 2, pp. 227–234, 1995 (cited on pp. 4, 100).

[128] G. L. Nemhauser and L. A. Wolsey, "Best algorithms for approximating the maximum of a submodular set function," *Mathematics of Operations Research*, vol. 3, no. 3, pp. 177–188, 1978 (cited on pp. 1, 15, 16).

[129] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions - i," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978 (cited on pp. 1, 18).

[130] ——, "An analysis of approximations for maximizing submodular set functions - ii," *Mathematical Programming*, vol. 8, pp. 73–87, 1978 (cited on pp. 1, 13).

[131] Y. Nesterov, "Squared functional systems and optimization problems," in *High Performance Optimization*, Springer, 2000, pp. 405–440 (cited on pp. 4, 106).

[132] D. Noll and A. Rondepierre, "On local convergence of the method of alternating projections," *Foundations of Computational Mathematics*, 2015, published online (cited on pp. 89, 92).

[133] R. O'Donnell and Y. Zhou, "Approximability and proof complexity," in *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2013, pp. 1537–1556 (cited on p. 107).

[134] A. Ostfeld, J. G. Uber, E. Salomons, J. W. Berry, W. E. Hart, C. A. Phillips, J.-P. Watson, G. Dorini, P. Jonkergouw, Z. Kapelan, F. di Pierro, S.-T. Khu, D. Savic, D. Eliades, M. Polycarpou, S. R. Ghimire, B. D. Barkdoll, R. Gueli, J. J. Huang, E. A. McBean, W. James, A. Krause, J. Leskovec, S. Isovitsch, J. Xu, C. Guestrin, J. VanBriesen, M. Small, P. Fischbeck, A. Preis, M. Propato, O. Piller, G. B. Trachtman, Z. Y. Wu, and T. Walski, "The battle of the water sensor networks (BWSN): A design challenge for engineers and algorithms," *Journal of Water Resources Planning and Management*, vol. 134, no. 6, pp. 556–568, 2008 (cited on p. 57).

[135] J. Oxley, "Matroid applications," in *Matroid Applications*, N. White, Ed., vol. 40, Cambridge University Press, 1992, pp. 73–90 (cited on p. 78).

[136] P. A. Parrilo, "Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization," PhD thesis, California Institute of Technology, 2000 (cited on pp. 4, 106).

[137] M. Putinar, "Positive polynomials on compact semi-algebraic sets," *Indiana University Mathematics Journal*, vol. 42, no. 3, pp. 969–984, 1993 (cited on p. 107).

[138] Q. Qu, J. Sun, and J. Wright, "Finding a sparse vector in a subspace: linear sparsity using alternating directions," in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 3401–3409 (cited on pp. 74, 75, 79–83, 93, 119).

[139] B. Recht, "A simpler approach to matrix completion," *Journal of Machine Learning Research*, vol. 12, pp. 3413–3430, 2011 (cited on p. 63).

[140] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Review*, vol. 52, no. 3, pp. 471–501, 2010 (cited on pp. 63, 83).

[141] R. T. Rockafellar, *Convex Analysis*. Princeton University Press, 1996 (cited on p. 34).

[142] T. Rothvoß, "The lasserre hierarchy in approximation algorithms," *Lecture Notes for the MAPSP*, pp. 1–25, 2013 (cited on pp. 106, 109).

[143] R. Saab, R. Chartrand, and O. Yilmaz, "Stable sparse approximations via nonconvex optimization," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008, pp. 3885–3888 (cited on p. 109).

[144] K. Schmüdgen, "The k-moment problem for compact semi-algebraic sets," *Mathematische Annalen*, vol. 289, no. 1, pp. 203–206, 1991 (cited on p. 107).

[145] A. Schrijver, "A combinatorial algorithm minimizing submodular functions in strongly polynomial time," *Journal of Combinatorial Theory, Series B*, vol. 80, no. 2, pp. 346–355, 2000 (cited on p. 9).

[146] ——, *Combinatorial Optimization: Polyhedra and Efficiency*. Springer Science & Business Media, 2003, vol. 24 (cited on p. 1).

[147] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *Journal of the ACM*, vol. 27, pp. 701–717, 1980 (cited on p. 61).

[148] Y. Seginer, "The expected norm of random matrices," *Combinatorics, Probability and Computing*, vol. 9, no. 2, pp. 149–166, 2000 (cited on p. 115).

[149] A. Shioura, "On the pipage rounding algorithm for submodular function maximization — a view from discrete convex analysis—," *Discrete Mathematics, Algorithms and Applications*, vol. 1, no. 1, pp. 1–23, 2009 (cited on pp. 20, 28).

[150] N. Z. Shor, "An approach to obtaining global extremums in polynomial mathematical programming problems," *Cybernetics*, vol. 23, no. 5, pp. 695–700, 1987 (cited on pp. 4, 106).

[151] A. Singh, A. Guillory, and J. Bilmes, "On bisubmodular maximization," in *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*, 2012, pp. 1055–1063 (cited on p. 28).

[152] T. Soma, "Fast deterministic algorithms for matrix completion problems," *SIAM Journal on Discrete Mathematics*, vol. 28, no. 1, pp. 490–502, 2014 (cited on pp. 3, 62, 76).

[153] H. O. Song, R. Girshick, S. Jegelka, J. Mairal, Z. Harchaoui, and T. Darrell, "On learning to localize objects with minimal supervision," in *Proceedings of the 31st International Conference of Machine Learning (ICML)*, 2014 (cited on p. 13).

[154] L. Sorber, M. Van Barel, and L. De Lathauwer. (2013). Tensorlab v1.0, [Online]. Available: http://esat.kuleuven.be/sista/tensorlab (cited on p. 94).

[155] D. A. Spielman, H. Wang, and J. Wright, "Exact recovery of sparsely-used dictionaries," in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, 2013, pp. 3087–3090 (cited on p. 74).

[156] G. W. Stewart, *Matrix algorithms. Vol. II.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001, Eigensystems (cited on p. 96).

[157] G. W. Stewart and J. G. Sun, *Matrix perturbation theory*, ser. Computer Science and Scientific Computing. Academic Press, Inc., Boston, MA, 1990 (cited on p. 91).

[158] M. Streeter and D. Golovin, "An online algorithm for maximizing submodular functions," in *Advances in Neural Information Processing Systems (NIPS)*, 2009, pp. 1577–1584 (cited on p. 1).

[159] M. Sviridenko, "A note on maximizing a submodular set function subject to a knapsack constraint," *Operations Research Letters*, vol. 32, no. 1, pp. 41–43, 2004 (cited on pp. 1, 17, 18).

[160] M. Sviridenko, J. Vondrák, and J. Ward, "Optimal approximation for submodular and supermodular optimization with bounded curvature," in *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2015, pp. 1134–1148 (cited on p. 55).

[161] G. Tang and P. Shah, "Guaranteed tensor decomposition: A moment approach," *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pp. 1491–1500, 2015 (cited on p. 109).

[162] É. Tardos, C. A. Tovey, and M. A. Trick, "Layered augmenting path algorithms," *Mathematics of Operations Research*, vol. 11, no. 2, pp. 362–370, 1986 (cited on p. 11).

[163] The 8th Annual Water Distribution Systems Analysis Symposium. (2008). Bwsn utility program, [Online]. Available: http://www.water-simulation.com/wsp/about/bwsn/ (cited on p. 57).

[164] A. M. Tillmann and M. E. Pfetsch, "The computational complexity of the restricted isometry property, the nullspace property, and related concepts in compressed sensing," *IEEE Transactions on Information Theory*, vol. 60, no. 2, pp. 1248–1259, 2014 (cited on p. 103).

[165] W. T. Tutte, "The factorization of linear graphs," *Jornal of London Mathematical Society*, vol. 22, pp. 107–111, 1947 (cited on pp. 3, 60, 61).

[166] J. Vondrák, "Symmetry and approximability of submodular maximization problems," *SIAM Journal on Computing*, vol. 42, no. 1, pp. 265–304, 2013 (cited on p. 2).

[167] P.-J. Wan, D.-Z. Du, P. Pardalos, and W. Wu, "Greedy approximations for minimum submodular cover with submodular cost," *Computational Optimization and Applications*, vol. 45, no. 2, pp. 463–474, 2009 (cited on pp. 18, 52).

[168] J. Ward and S. Živný, "Maximizing bisubmodular and k-submodular functions," in *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2014, pp. 1468–1481 (cited on p. 28).

[169] P.-Å. Wedin, "Perturbation bounds in connection with singular value decomposition," vol. 12, pp. 99–111, 1972 (cited on p. 91).

[170] K. Wei, R. Iyer, and J. Bilmes, "Fast multi-stage submodular maximization," in *Proceedings of The 31st International Conference on Machine Learning (ICML)*, 2014 (cited on p. 1).

[171] D. Welsh, "On matroid theorems of Edmonds and Rado," *Journal of the London Mathematical Society*, vol. S2, pp. 251–256, 1970 (cited on p. 67).

[172] H. Whitney, "On the abstract properties of linear dependence," *American Journal of Mathematics*, vol. 57, no. 3, pp. 509–533, 1935 (cited on p. 10).

[173] L. A. Wolsey, "An analysis of the greedy algorithm for the submodular set covering problem," *Combinatorica*, vol. 2, no. 4, pp. 385–393, 1982 (cited on pp. 2, 13, 18, 52).

[174] S. M. S. T. Yazdi and S. A. Savari, "A max-flow/min-cut algorithm for linear deterministic relay networks," *IEEE Transactions on Information Theory*, vol. 57, no. 5, pp. 3005–3015, 2011 (cited on p. 67).

[175] ——, "A deterministic polynomial-time algorithm for constructing a multicast coding scheme for linear deterministic relay networks," *IEEE Transactions on Information Theory*, vol. 59, no. 11, pp. 7541–7552, 2013 (cited on pp. 3, 65, 66).

[176] Z. Zhang and R. W. Yeung, "A non-Shannon-type conditional inequality of information quantities," *IEEE Transactions on Information Theory*, vol. 43, no. 6, pp. 1982–1986, 1997 (cited on p. 8).

[177] X. Zhao, G. Zhou, W. Dai, T. Xu, and W. Wang, "Joint image separation and dictionary learning," in *Proccedings of 18th International Conference on Digital Signal Processing (DSP)*, 2013, pp. 1–6 (cited on p. 96).

[178] R. Zippel, *Efficient Polynomial Computations*. Kluwer Academic Publishers, Boston, 1993 (cited on p. 61).