Polynomial Optimization Problems

— Approximation Algorithms and Applications

LI, Zhening

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy
in
Systems Engineering and Engineering Management

The Chinese University of Hong Kong
July 2011

Thesis/Assessment Committee

Professor Duan Li (Chair)

Professor Shuzhong Zhang (Thesis Advisor)

Professor Anthony Man-Cho So (Committee Member)

Professor Yinyu Ye (External Examiner)

Abstract

Polynomial optimization problem is to optimize a generic multivariate polynomial function, subject to some suitable polynomial equality and inequality constraints. Such problem formulation dates back to the 19th century, when the relationship between nonnegative polynomials and sum of squares were discussed by Hilbert. Polynomial optimization is one of the fundamental problems in the field of optimization, and has applications in a large range of areas, including biomedical engineering, control theory, graph theory, investment science, material science, numerical linear algebra, quantum mechanics, signal processing, speech recognition, etc. This thesis presents a study of some important subclasses of polynomial optimization problems arising from various applications. The focus is on optimizing a high degree polynomial function, over some commonly encountered constraint sets, such as the Euclidean ball, the Euclidean sphere, the intersection of co-centered ellipsoids, the binary hypercube, as well as a combination of them. Specifically, five classes of models are discussed, i.e., optimizing a multilinear function with quadratic constraints, a homogeneous polynomial with quadratic constraints, a general polynomial with convex constraints, a general polynomial with binary constraints, and a homogeneous polynomial with binary and spherical constraints. All the problems under consideration are NP-hard in general. The main contribution of this thesis is on the design and analysis of polynomial-time approximation algorithms with guaranteed worst-case performance ratios. These approximation ratios are dependent on the problem dimensions only, and the new results improve some of the existing results in the literature. In each class of these optimization models, some application examples are discussed and results of numerical experiments are reported, revealing good practical performance of the proposed algorithms for solving some randomly generated test instances.

摘要

多项式优化问题是基于某种多项式等式和多项式不等式约束条件下以多元多项式函数为目标的最优化问题。这类问题的研究可追述到十九世纪，希尔伯特讨论了非负多项式函数和多项式平方和函数之间的关系。多项式优化问题是最优化领域最根本的问题之一，其应用领域非常广泛，其中包括了生物医学工程、控制论、图论、投资理论、材料科学、数值代数、量子力学、信号处理、语音识别等等。本论文着重研究多项式优化问题中与各类应用领域相关的一些重要的子问题。研究的优化问题重点放在以高次多项式函数为目标，约束集为一些通常出现的多项式约束集合，例如：欧几里德球体、欧几里德球面、同心椭圆的交集、高维立方体的顶点集、以及它们的各种组合。具体来说，本文研究五类多项式优化模型，它们是二次函数约束下的多线性函数优化、二次函数约束下的齐次多项式优化、凸约束下的一般多项式优化、二元约束下的多项式优化，以及二元和球面约束下的齐次多项式优化。所有涉及的优化问题都为NP困难。本论文的主要贡献在于设计和分析多项式时间的近似算法。这些算法得到的解都有一定的近似比值保证，且近似比值仅仅与优化问题的维数有关，新结果改进了现有文献的结果。在每类优化模型的研讨中，本论文列举了一些应用实例，并汇报了数值实验结果，报告显示这类算法在解决一些随机产生的案例时执行效果非常好。

Acknowledgement

I would never have finished this work, or even started it, if it were not for my father, who instilled in me a passion for learning and striving. Almost eight years have passed since my father's sudden passing away, and he has always stayed in my heart, supporting and encouraging me during the whole journey of my doctoral work. All the difficulties have become quite manageable, and my confidence in myself has never been lost.

It is a great fortune to have Professor Shuzhong Zhang as my doctoral advisor, who got me excited about various interesting research topics, as well as moralities and other human qualities. I would like to express my heartfelt thanks to him, for his guidance, constant support and numerous helps when in need, as well as many enjoyable hours of discussions that we have had over the last four years. I am greatly indebted to him.

I would like to express my gratitude to those who gave me the possibility to complete this work, especially to my former colleague, Professor Simai He, from whom I have been benefitted tremendously for his insightful comments, valuable suggestions and efforts in discussing technical problems. I would also like to thank Professors Duan Li, Anthony So, and Yinyu Ye for serving on my thesis committee.

I would also like to extend my gratitude to the professors, supporting staff members and the fellow postgraduate students at Department of Systems Engineering and Engineering Management for providing me with technical and non-technical supports. It has been a great pleasure to work with them. Besides, this work would not have been possible without the general support from The Chinese University of Hong Kong.

Finally, I would like to express my deepest gratitude to the three most important women in my family, whose love and support enabled me to complete this work. To my dear mother: thank you for continuously supporting and taking care of me; To my dear wife: thank you for your patience, trust and encouragement; And to my dear daughter: thank you for bringing me so much indelible happiness, I owe you a great deal!

This work is dedicated to my father

Contents

Chapter 1

Introduction

Polynomial optimization problem is the following generic optimization model

$$\begin{aligned} (POP) \quad \min \quad & p(\boldsymbol{x}) \\ \text{s.t.} \quad & f_i(\boldsymbol{x}) \leq 0,\ i = 1, 2, \ldots, m_1, \\ & g_j(\boldsymbol{x}) = 0,\ j = 1, 2, \ldots, m_2, \\ & \boldsymbol{x} = (x_1, x_2, \cdots, x_n)^{\mathrm{T}} \in \mathbb{R}^n, \end{aligned}$$

where $p(\boldsymbol{x})$, $f_i(\boldsymbol{x})\,(i = 1, 2, \ldots, m_1)$ and $g_j(\boldsymbol{x})\,(j = 1, 2, \ldots, m_2)$ are some multivariate polynomial functions. This problem is a fundamental model in the field of optimization, and has applications in a wide range of areas. Many algorithms have been proposed for subclasses of (POP), and specialized software packages have been developed.

1.1 History

The modern history of polynomial optimization may date back to the 19th century when the relationship between nonnegative polynomial function and the sum of squares of polynomials was studied. Given a multivariate polynomial function that takes only nonnegative values over the real numbers, can it be represented as a sum of squares of polynomial functions? Hilbert [54] gave a concrete answer in 1888, which asserted that the only cases for a nonnegative polynomial to be a sum of squares are: univariate polynomials; multivariate quadratic polynomials; and bivariate quartic polynomials. Later, in Hilbert's 17th problem—one of the famous 23 Hilbert problems addressed in a celebrated speech in 1900 by Hilbert, a nonnegative polynomial entails expression of definite rational functions as quotients of sums of squares. Given a multivariate

polynomial function that takes only nonnegative values over the real numbers, can it be represented as a sum of squares of rational functions? This was solved in the affirmative, by Artin [8] in 1927. A continuous and constructive algorithm was later found by Delzell [30] in 1984. About 10 years ago, Lasserre [70, 71] and Parrilo [93, 94] proposed a method called the sum of squares (SOS) to solve general polynomial optimization problem. The method is based on the fact that deciding whether a given polynomial is a sum of squares can be reduced to the feasibility of a semidefinite program (SDP). The SOS approach has a strong theoretical appeal, as it can in principle solve any polynomial optimization problem to any given accuracy.

1.2 Applications

Polynomial optimizations have wide applications—just to name a few examples: biomedical engineering, control theory, graph theory, investment science, material science, numerical linear algebra, quantum mechanics, signal processing, speech recognition. It is basically impossible to list, even very partially, the success stories of (POP), simply due to its sheer size in the literature. To motivate our study, below we shall nonetheless mention some sample applications to illustrate the usefulness of (POP).

Polynomial optimizations have immediate applications in investment science. For instance, the celebrated mean-variance model was proposed by Markowitz [81] early in 1952, where the portfolio selection problem is modeled by minimizing the variance of the investments subject to its target return. In control theory, Roberts and Newmann [107] studied polynomial optimization of stochastic feedback control for stable plants. In diffusion magnetic resonance imaging (MRI), Barmpoutis et al. [14] presented a case for the fourth order tensor approximation. In fact, there are a large class of (POP) arising from tensor approximations and decompositions, which are originated from applications in psychometrics and chemometrics (see an excellent survey by Kolda and Bader [68]). Polynomial optimizations have also applications in sinal processing. Maricic et al. [79] proposed a quartic polynomial model for blind channel equalization in digital communication, and Qi and Teo [101] conducted global optimization for high degree polynomial minimization models arising from signal processing. In quantum physics, Dahl et al. [27] proposed a polynomial optimization model to verify whether a physical system is entangled or not, which is an important problem in quantum physics.

Gurvits [42] showed that the entanglement verification is NP-hard in general. In fact, the model discussed in [27] is related to the nonnegative quadratic mappings studied by Luo et al. [76].

Among generic polynomial functions, homogeneous polynomials play an important role in approximation theory (see e.g., two recent papers by Kroó and Szabados [69] and Varjú [117]). Essentially their results state that the homogeneous polynomial functions are fairly 'dense' among continuous functions in a certain well-defined sense. As such, optimizations of homogeneous polynomials become important. As an example, Ghosh et al. [39] formulated a fiber detection problem in diffusion MRI by maximizing a homogenous polynomial function subject to the Euclidean spherical constraint, i.e.,

$$\begin{aligned} (H_S) \quad & \max \quad f(\boldsymbol{x}) \\ & \text{s.t.} \quad \|\boldsymbol{x}\|_2 = 1, \, \boldsymbol{x} \in \mathbb{R}^n. \end{aligned}$$

The constraint of (H_S) is a typical polynomial equality constraint. In this case, the degree of the homogeneous polynomial $f(\boldsymbol{x})$ may be high. This particular model (H_S) is widely appeared in the following examples. In material sciences, Soare et al. [110] proposed some 4th, 6th and 8th order homogeneous polynomials to model the plastic anisotropy of orthotropic sheet metal. In statistics, Micchelli and Olsen [82] considered a maximum-likelihood estimation model in speech recognition. In numerical linear algebra, (H_S) is the formulation of an interesting problem: the eigenvalues of tensors (see Qi [99, 100] and Ni et al. [91]). Another widely used application of (H_S) is regarding to the best rank-one approximation of higher order tensors (see [67, 68]).

In fact, Markowitz's mean-variance model [81] mentioned previously is also optimization on a homogeneous polynomial, in particular, a quadratic form. Recently, an intensified discussion on investment models involving more than the first two moments (for instance to include the skewness and the kurtosis of the investment returns) have been another source of inspiration underlying polynomial optimizations. Mandelbrot and Hudson [78] made a strong case against a 'normal view' of the investment returns. The use of higher moments in portfolio selection becomes quite necessary. Along that line, several authors proposed investment models incorporating the higher moments, e.g., de Athayde and Flôre [10], Prakash et al. [96], Jondeau and Rockinger [60], and Kleniati et al. [64]. However, in those models, the polynomial functions involved are

no longer homogeneous. In particular, a very general model in [64] is

$$\begin{array}{ll}\max & \alpha\sum_{i=1}^n \mu_i x_i - \beta\sum_{i,j=1}^n \sigma_{ij}x_i x_j + \gamma\sum_{i,j,k=1}^n \varsigma_{ijk}x_i x_j x_k - \delta\sum_{i,j,k,\ell=1}^n \kappa_{ijk\ell}x_i x_j x_k x_\ell \\ \text{s.t.} & \sum_{i=1}^n x_i = 1,\ \boldsymbol{x}\geq \mathbf{0},\ \boldsymbol{x}\in\mathbb{R}^n,\end{array}$$

where (μ_i), (σ_{ij}), (ς_{ijk}) and $(\kappa_{ijk\ell})$ are the first four central moments of the given n assets. The nonnegative parameters $\alpha, \beta, \gamma, \delta$ measure the investor's preference to the four moments, and they sum up to one, i.e., $\alpha+\beta+\gamma+\delta = 1$. Besides investment science, many other important applications of polynomial function optimization involve an objective that is intrinsically inhomogeneous. The other example is the least square formulation to the sensor network localization problem proposed in Luo and Zhang [77]. Specifically, the problem takes the form of

$$\begin{array}{ll}\min & \sum_{i,j\in S}\left(\|\boldsymbol{x}^i-\boldsymbol{x}^j\|^2 - {d_{ij}}^2\right)^2 + \sum_{i\in S, j\in A}\left(\|\boldsymbol{x}^i-\boldsymbol{a}^j\|^2 - {d_{ij}}^2\right)^2 \\ \text{s.t.} & \boldsymbol{x}^i\in\mathbb{R}^3,\ i\in S,\end{array}$$

where A and S denote the set of anchor nodes and sensor nodes respectively, d_{ij} $(i\in S, j\in S\cup A)$ are (possibly noisy) distance measurements, $\boldsymbol{a}^j$ $(j\in A)$ denote the known positions of anchor nodes, while $\boldsymbol{x}^i$ $(i\in S)$ represent the positions of sensor nodes to be estimated.

Apart from the continuous models discussed above, polynomial optimizations over variables in discrete values, in particular binary variables, are also widely studied. For example, maximize a polynomial function over variables picking from 1 or -1, i.e.,

$$\begin{array}{lll}(P_B) & \max & p(\boldsymbol{x}) \\ & \text{s.t.} & x_i\in\{1,-1\},\ i=1,2,\ldots,n.\end{array}$$

This type of problem can be found in a great variety of application domains. Indeed, (P_B) has been investigated extensively in the quadratic case, due to its connections to various graph partitioning problems, e.g., the maximum cut problem [40]. If the degree of the polynomial goes higher, the following hypergraph max-cover problem is also well studied. Given a hypergraph $H=(V,E)$ with V being the set of vertices and E the set of hyperedges (or subsets of V), and each hyperedge $e\in E$ is associated with a real-valued weight $w(e)$. The problem is to find a subset S of the vertices set V, such that the total weight of the hyperedges covered by S is maximized. Denoting $x_i\in\{0,1\}$ $(i=1,2,\ldots,n)$ to indicate whether or not vertex i is selected in S. The problem thus is $\max_{\boldsymbol{x}\in\{0,1\}^n}\sum_{e\in E} w(e)\prod_{i\in e} x_i$. By a simple variable transformation $x_i\to(x_i+1)/2$, the problem is transformed to (P_B), and vice versa.

Note that the model (P_B) is a fundamental problem in integer programming. As such it has received attention in the literature (see e.g., [43, 44]). It is also known as the Fourier support graph problem. Mathematically, a polynomial function $p : \{-1, 1\}^n \to \mathbb{R}$ has Fourier expansion $p(\boldsymbol{x}) = \sum_{S \subset \{1,2,\cdots,n\}} \hat{p}(S) \prod_{i \in S} x_i$, which is also called the Fourier support graph. By assuming that $p(\boldsymbol{x})$ has only succinct (polynomially many) non-zero Fourier coefficient $\hat{p}(S)$, can we compute the maximum value of $p(\boldsymbol{x})$ over the discrete hypercube $\{1, -1\}^n$, or alternatively can we find a good approximate solution in polynomial-time? The latter question actually motivates the discrete polynomial optimization models studied in this thesis. In general, (P_B) is closely related to finding the maximum weighted independent set in a graph. In fact, any instance of (P_B) can be transformed into the maximum weighted independent set problem, which is also the most commonly used technique in the literature for solving (P_B) (see e.g., [12, 106]). The transformation uses the concept of a *conflict graph* of a 0-1 polynomial function, for details, one is referred to [21, 9]. Beyond its connection to the graph problems, (P_B) also has applications in neural networks [58, 21, 6], error-correcting codes [21, 97], etc. In fact, Bruck and Blaum [21] reveal the natural equivalence within the model (P_B), maximum likelihood decoding of error-correcting codes, and finding the global maximum of a neural network. Recently Khot and Naor [63] show that it has applications in the problem of refutation of random k-CNF formulas [32, 35, 33, 34].

If the objective polynomial function in (P_B) is homogeneous, likewise, the homogeneous quadratic case has been studied extensively, e.g., [40, 88, 90, 5]. Homogeneous cubic polynomial case is also discussed by Khot and Naor [63]. Another interesting problem of this class is the $\infty \mapsto 1$-norm of a matrix $\boldsymbol{F} = (F_{ij})$, studied by Alon and Naor [5], i.e.,

$$\begin{array}{rll} \|\boldsymbol{F}\|_{\infty \mapsto 1} = & \max & \sum_{1 \le i \le n_1, 1 \le j \le n_2} F_{ij} x_i y_j \\ & \text{s.t.} & \boldsymbol{x} \in \{1, -1\}^{n_1}, \boldsymbol{y} \in \{1, -1\}^{n_2}. \end{array}$$

It is quite natural to extend the problem of $\infty \mapsto 1$-norm to higher order tensors. In particular, the $\infty \mapsto 1$-norm of a d-th order tensor $\boldsymbol{F} = (F_{i_1 i_2 \cdots i_d})$ can be defined as

$$\begin{array}{ll} \max & \sum_{1 \le i_1 \le n_1, 1 \le i_2 \le n_2, \cdots, 1 \le i_d \le n_d} F_{i_1 i_2 \cdots i_d} x^1_{i_1} x^2_{i_2} \cdots x^d_{i_d} \\ \text{s.t.} & \boldsymbol{x}^k \in \{1, -1\}^{n_k},\ k = 1, 2, \ldots, d. \end{array}$$

Another generalization of the matrix $\infty \mapsto 1$-norm is to extend the entry F_{ij} of the

matrix $\boldsymbol{F}$ to a symmetric matrix $\boldsymbol{A}_{ij} \in \mathbb{R}^{m\times m}$, i.e., the problem of

$$\begin{array}{ll} \max & \lambda_{\max}\left(\sum_{1\le i\le n_1, 1\le j\le n_2} x_i y_j \boldsymbol{A}_{ij}\right) \\ \text{s.t.} & \boldsymbol{x}\in\{1,-1\}^{n_1},\ \boldsymbol{y}\in\{1,-1\}^{n_2}, \end{array}$$

where $\lambda_{\max}$ indicates the largest eigenvalue of a matrix. If the matrix $\boldsymbol{A}_{ij} \in \mathbb{R}^{m_1\times m_2}$ is not restricted to be symmetric, we may instead maximize the largest singular value, i.e.,

$$\begin{array}{ll} \max & \sigma_{\max}\left(\sum_{1\le i\le n_1, 1\le j\le n_2} x_i y_j \boldsymbol{A}_{ij}\right) \\ \text{s.t.} & \boldsymbol{x}\in\{1,-1\}^{n_1},\ \boldsymbol{y}\in\{1,-1\}^{n_2}. \end{array}$$

These two problems are actually equivalent to

$$\begin{array}{ll} \max & \sum_{1\le i\le n_1, 1\le j\le n_2, 1\le k,\ell\le m} F_{ijk\ell}\, x_i y_j z_k z_\ell \\ \text{s.t.} & \boldsymbol{x}\in\{1,-1\}^{n_1},\ \boldsymbol{y}\in\{1,-1\}^{n_2}, \\ & \|\boldsymbol{z}\|_2 = 1,\ \boldsymbol{z}\in\mathbb{R}^m \end{array}$$

and

$$\begin{array}{ll} \max & \sum_{1\le i\le n_1, 1\le j\le n_2, 1\le k\le m_1, 1\le \ell\le m_2} F_{ijk\ell}\, x_i y_j z_k w_\ell \\ \text{s.t.} & \boldsymbol{x}\in\{1,-1\}^{n_1},\ \boldsymbol{y}\in\{1,-1\}^{n_2}, \\ & \|\boldsymbol{z}\|_2 = \|\boldsymbol{w}\|_2 = 1,\ \boldsymbol{z}\in\mathbb{R}^{m_1},\ \boldsymbol{w}\in\mathbb{R}^{m_2} \end{array}$$

respectively, where $\boldsymbol{F} = (F_{ijk\ell})$ is a fourth order tensor, whose (i,j,k,ℓ)-th entry is the (k,ℓ)-th entry of the matrix $\boldsymbol{A}_{ij}$. These two special models of (POP) extends polynomial integer programming problems to the mixed integer programming problems, which is also an important subclass of (POP) studied in this thesis.

1.3 Algorithms

Polynomial optimization problems are typically non-convex and highly nonlinear. In most cases, (POP) is NP-hard, even for very special instances, such as maximizing a cubic polynomial over a sphere (see Nestorov [90]), maximizing a quadratic form in binary variables (see e.g., Goemans and Williamson [40]), etc. The reader is referred to de Klerk [65] for a survey on the computational complexity issues of polynomial optimization over some simple constraint sets. In the case that the constraint set is a simplex and the objective polynomial has a fixed degree, it is possible to derive polynomial-time approximation schemes (PTAS) (see de Klerk et al. [66]), albeit the result is viewed mostly as a theoretical one. Almost in all practical situations, the

problem is difficult to solve, theoretically as well as numerically. However, the search for general and efficient algorithms for polynomial optimization has been a priority for many mathematical optimizers and researchers in various applications.

Perhaps the very first attempt for solving polynomial optimization problems is taking them as nonlinear programming problems, and many existing algorithms and software packages are available, including KNITRO, BARON, IPOPT, SNOPT, and Matlab optimization toolbox. However, these algorithms and solvers are not tailor made for polynomial optimization problems, and so the performance may vary greatly from problem instance to instance. One direct approach is to apply the method of Lagrange multipliers to reach a set of multivariate polynomial equations, which is the Karush-Kuhn-Tucker (KKT) system that provides the necessary conditions for optimality (see e.g., [122, 39, 57]). In [39], the authors develop special algorithms for that purpose, such as subdivision methods proposed by Mourrain and Pavone [84], and generalized normal forms algorithms designed by Mourrain and Trébuchet [85]. However, the shortcomings of these methods are apparent if the degree of the polynomial is high. Generic solution methods based on nonlinear programming and global optimization have been studied and tested (see e.g., Qi [98] and Qi et al. [102], and the references therein). Recently, a tensor eigenvalue based method for a global polynomial optimization problem was also studies by Qi et al. [103]. Moreover, Parpas and Rustem [92], and Maringer and Parpas [80] proposed diffusion-based methods to solve the non-convex polynomial optimization models arising from portfolio selection involving higher moments. For polynomial integer programming models, e.g., (P_B), the most commonly used technique in the literature is transforming them to the maximum weighted independent set problems (see e.g., [12, 106]), by using the concept of a conflict graph of a 0-1 polynomial function.

Sum of squares (SOS) approach has been one major systematic approach for solving general polynomial optimization problems. The approach was proposed by Lasserre [70, 71] and Parrilo [93, 94], and significant research on the SOS method has been conducted in recent ten years. The SOS method has a strong theoretical appeal, by constructing a sequence of semidefinite programming (SDP) relaxations of the given polynomial optimization problem in such a way that the corresponding optimal values are monotone and converge to the optimal value of the original problem. Thus it can in principle solve any instance of (POP) to any given accuracy. For univariate polynomial optimization,

Nesterov [89] showed that the SOS method in combination with the SDP solution has a polynomial-time complexity. This is also true for unconstrained multivariate quadratic polynomial and bivariate quartic polynomial when the nonnegativity is equivalent to the sum of squares. In general, however, the SDP problems required to be solved by the SOS method may grow very large, and is not practical when the program dimension goes high. At any rate, thanks to the recently developed efficient SDP solvers (e.g., SeDuMi of Sturm [112], SDPT3 of Toh et al. [115]), the SOS method appears to be attractive. Henrion and Lasserre [52] developed a specialized tool known as GloptiPoly (the latest version, GloptiPoly 3, can be found in Henrion et al. [53]) for finding a global optimal solution of polynomial optimization problems on the SOS method, based on Matlab and SeDuMi. For an overview on the recent theoretical developments, we refer to the excellent survey by Laurent [72].

On the other side, the intractability of general polynomial optimizations therefore motivates the search for suboptimal, or more formally, approximate solutions. In the case that the objective polynomial is quadratic, a well known example is the semidefinite programming relaxation and randomization approach for the max-cut problem due to Goemans and Williamson [40], where essentially a 0.878-approximation ratio of the model $\max_{\boldsymbol{x}\in\{1,-1\}^n} \boldsymbol{x}^{\mathrm{T}}\boldsymbol{F}\boldsymbol{x}$ is shown with $\boldsymbol{F}$ being the Laplacian of a given graph. Note that the approach in [40] has been generalized subsequently by many authors, including Nesterov [88], Ye [118, 119], Nemirovski et al. [87], Zhang [120], Charikar and Wirth [24], Alon and Naor [5], Zhang and Huang [121], Luo et al. [75], and He et al. [50]. In particular, when the matrix $\boldsymbol{F}$ is only known to be positive semidefinite, Nestrov [88] derived a 0.636-approximation bound for $\max_{\boldsymbol{x}\in\{1,-1\}^n} \boldsymbol{x}^{\mathrm{T}}\boldsymbol{F}\boldsymbol{x}$. For general diagonal-free matrix $\boldsymbol{F}$, Charikar and Wirth [24] derived an $\Omega(1/\log n)$-approximation bound, while its inapproximate results are also discussed by Arora et al. [7]. For the matrix $\infty \mapsto 1$-norm problem $\max_{\boldsymbol{x}\in\{1,-1\}^{n_1},\boldsymbol{y}\in\{1,-1\}^{n_2}} \boldsymbol{x}^{\mathrm{T}}\boldsymbol{F}\boldsymbol{y}$, Alon and Naor [5] derived a 0.56-approximation bound. Remark that all these approximation bounds remain hitherto the best available ones. In continuous polynomial optimizations, Nemirovski et al. [87] proposed an $\Omega(1/\log m)$-approximation bound for maximizing a quadratic form over the intersection of m co-centered ellipsoids. Their models are further studied and generalized by Luo et al. [75] and He et al. [50].

Among all the successful approximation stories mentioned above, the objective polynomials are all quadratic. However, there are only a few approximation results in the

literature when the degree of the objective polynomial is greater than two. Perhaps the very first one is due to de Klerk et al. [66] in deriving a PTAS of optimizing a fixed degree homogenous polynomial over a simplex, and it turns out to be a PTAS of optimizing a fixed degree even form (homogeneous polynomial with only even exponents) over the spherical constraint. Later, Barvinok [15] showed that optimizing a certain class of polynomials over the spherical constraint also admits a randomized PTAS. Note that the results in [66, 15] apply only when the objective polynomial has some special structure. A quite general result is due to Khot and Naor [63], where they showed how to estimate the optimal value of the problem $\max_{x\in\{1,-1\}^n} \sum_{1\le i,j,k\le n} F_{ijk}x_ix_jx_k$ with (F_{ijk}) being square-free, i.e., $F_{ijk} = 0$ whenever two of the indices are equal. Specifically, they presented a polynomial-time randomized procedure to get an estimated value that is no less than $\Omega\left(\sqrt{\frac{\log n}{n}}\right)$ times the optimal value. Two recent papers (Luo and Zhang [77], and Ling et al. [73]) discussed polynomial optimization problems with the degree of objective polynomial being four, and start a whole new research on approximation algorithms for high degree polynomial optimizations, which are essentially the main subject in this thesis. Luo and Zhang [77] considered quartic optimization, and showed that optimizing a homogenous quartic form over the intersection of some cocentered ellipsoids is essentially equivalent to its (quadratic) SDP relaxation problem, which is itself also NP-hard. However, this gives a handle on the design of approximation algorithms with provable worst-case approximation ratios. Ling et al. [73] considered a special quartic optimization model. Basically, the problem is to minimize a biquadratic function over two spherical constraints. In [73], approximate solutions as well as exact solutions using the SOS method are considered. The approximation bounds in [73] are indeed comparable to the bound in [77], although they are dealing with two different models. Very recently, Zhang et al. [123] and Ling al. [74] further studied biquadratic function optimization over quadratic constraints. The relations with its bilinear SDP relaxation are discussed, based on which they derived some data dependent approximation bounds.

In the meanwhile, when the objective function of (POP) is a high degree inhomogeneous polynomial, we have not seen any approximation results so far, even in the relative sense (for a discussion on relative approximation algorithms, see Section 2.3). As a matter of fact, so far all the successful polynomial-time approximation algorithms with provable approximation ratios in the literature, e.g., the quadratic, cubic and

quartic models mentioned above are all dependent on the homogeneity in a crucial way. Technically, a homogenous polynomial function allows one to *scale* the overall function value along a given direction, which is an essential operation in proving the quality bound of the approximation algorithms. Thus, extending the solution methods and the corresponding analysis from *homogeneous* polynomial optimizations to the general *inhomogeneous* polynomials is not straightforward. These trigger us to search for approximate solutions of (POP) with an inhomogeneous polynomial objective, which is one of the targets to achieve in this thesis.

1.4 Main Contributions

This thesis is concerned with some important and widely used subclasses of polynomial optimization problems, including optimization of a multilinear function with quadratic constraints, a homogeneous polynomial with quadratic constraints, a general polynomial with convex constraints, a general polynomial with binary constraints, and a homogeneous polynomial with binary and spherical constraints. The detailed description of the problems studied is listed in Section 2.1.3. All these problems are NP-hard in general, and the focus is on the design and analysis of polynomial-time approximation algorithms with provable worst-case performance ratios. We also discuss the applications of these models, and the numerical performance of the proposed algorithms. Specifically, our contributions are highlighted as follows.

1. We propose approximation algorithms for optimization of any fixed degree homogeneous polynomial with quadratic constraints, which is the first such result for approximation algorithms of polynomial optimization problems with an arbitrary degree. The approximation ratios depend only on the dimensions of the problems concerned. Compared with any existing results for high degree polynomial optimizations, our approximation ratios improve the previous ones, when specialized to their particular degrees.

2. We establish systematic link identities between multilinear functions and homogeneous polynomials, and thus establish the same approximation ratios for homogeneous polynomial optimizations with their multilinear form relaxation problems.

3. We propose a general scheme to handle inhomogeneous polynomial optimizations

through the method of homogenization, and thus establish the same approximation ratios (in relative sense) for inhomogeneous polynomial optimizations with their homogeneous polynomial relaxation problems. It is the first approximation bound of approximation algorithms for general inhomogeneous polynomial optimizations with a high degree.

4. We propose several decomposition routines for polynomial optimizations over different types of constraint sets, and derive approximation bounds for multilinear function optimizations with their lower degree relaxation problems.

5. With the availability of our proposed approximation algorithms, we illustrate some potential modeling opportunities with the new optimization models.

This thesis is organized as follows. First in Chapter 2, we introduce the notations and models, as well as providing necessary preparations for better understanding the whole thesis. Then from Chapter 3 to Chapter 7, we discuss five subclasses of polynomial optimization problems, with each subclass in one chapter (the detail description of these subclasses is introduced in Section 2.1). In each of these five chapters, polynomial-time approximation algorithms with provable worst-case performance ratios will be proposed to solve the models concerned, followed by a discussion on their applications and/or a report on numerical performance of the algorithms proposed. Finally, in Chapter 8, we summarize the main results in this thesis, and discuss some recent developments and future research topics.

Chapter 2

Notations and Preliminaries

2.1 Notations and Models

Throughout this thesis, we exclusively use the boldface letters to denote vectors, matrices, and tensors in general (e.g., the decision variable $\boldsymbol{x}$, the data matrix $\boldsymbol{Q}$, and the tensor form $\boldsymbol{F}$), while the usual non-bold letters are reserved for scalars (e.g., x_1 being the first component of the vector $\boldsymbol{x}$, Q_{ij} being one entry of the matrix $\boldsymbol{Q}$).

2.1.1 Objective Functions

The objective functions of the optimization models studied in this thesis are all multivariate polynomial functions. The following multilinear tensor function (or multilinear form) plays a major role in the discussion

$$\text{Function T} \quad F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) := \sum_{1 \le i_1 \le n_1, 1 \le i_2 \le n_2, \cdots, 1 \le i_d \le n_d} F_{i_1 i_2 \cdots i_d} x^1_{i_1} x^2_{i_2} \cdots x^d_{i_d},$$

where $\boldsymbol{x}^k \in \mathbb{R}^{n_k}$ for $k = 1, 2, \ldots, d$; and the letter 'T' signifies the notion of *tensor*. In the shorthand notation we denote $\boldsymbol{F} = (F_{i_1 i_2 \cdots i_d}) \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ to be a d-th order tensor, and F to be its corresponding multilinear form. The meaning for multilinear states if one fixed $(\boldsymbol{x}^2, \boldsymbol{x}^3, \cdots, \boldsymbol{x}^d)$ in the function F, then it is a linear function of $\boldsymbol{x}^1$, and so on.

Closely related with the tensor form $\boldsymbol{F}$ is a general d-th degree homogeneous polynomial function $f(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{R}^n$. We call the tensor form $\boldsymbol{F} = (F_{i_1 i_2 \cdots i_d})$ *supersymmetric* (see [67]), if any of its components $F_{i_1 i_2 \cdots i_d}$ is invariant under all permutations of $\{i_1, i_2, \cdots, i_d\}$. As any homogeneous quadratic function uniquely determines

a symmetric matrix, a given d-th degree homogeneous polynomial function $f(\boldsymbol{x})$ also uniquely determines a super-symmetric tensor form. In particular, if we denote a d-th degree homogeneous polynomial function

$$\text{Function H} \quad f(x) := \sum_{1 \le i_1 \le i_2 \le \cdots \le i_d \le n} F'_{i_1 i_2 \cdots i_d} x_{i_1} x_{i_2} \cdots x_{i_d},$$

then its corresponding super-symmetric tensor form can be written as $\boldsymbol{F} = (F_{i_1 i_2 \cdots i_d}) \in \mathbb{R}^{n^d}$, with $F_{i_1 i_2 \cdots i_d} \equiv F'_{i_1 i_2 \cdots i_d} / |\Pi(i_1, i_2, \cdots, i_d)|$, where $|\Pi(i_1, i_2, \cdots, i_d)|$ is the number of distinctive permutations of the indices $\{i_1, i_2, \cdots, i_d\}$. This super-symmetric tensor representation is indeed unique. Let F be its corresponding multilinear form defined by the super-symmetric tensor $\boldsymbol{F}$, then we have $f(\boldsymbol{x}) = F(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x}}_{d})$. The letter 'H' here is used to emphasize that the polynomial function in question is *homogeneous*.

We shall also consider in this this the following mixed form

$$\text{Function M} \quad f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s) := F(\underbrace{\boldsymbol{x}^1, \boldsymbol{x}^1, \cdots, \boldsymbol{x}^1}_{d_1}, \underbrace{\boldsymbol{x}^2, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^2}_{d_2}, \cdots, \underbrace{\boldsymbol{x}^s, \boldsymbol{x}^s, \cdots, \boldsymbol{x}^s}_{d_s}),$$

where $d_1 + d_2 + \cdots + d_s = d$, $\boldsymbol{x}^k \in \mathbb{R}^{n_k}$ for $k = 1, 2, \ldots, s$, d-th order tensor form $\boldsymbol{F} \in \mathbb{R}^{{n_1}^{d_1} \times {n_2}^{d_2} \times \cdots \times {n_s}^{d_s}}$; and the letter 'M' signifies the notion of *mixed* polynomial form. We may without loss of generality assume that $\boldsymbol{F}$ has partial symmetric property, namely for any fixed $(\boldsymbol{x}^2, \boldsymbol{x}^3, \cdots, \boldsymbol{x}^s)$, $F(\underbrace{\cdot, \cdot, \cdots, \cdot}_{d_1}, \underbrace{\boldsymbol{x}^2, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^2}_{d_2}, \cdots, \underbrace{\boldsymbol{x}^s, \boldsymbol{x}^s, \cdots, \boldsymbol{x}^s}_{d_s})$ is a super-symmetric d_1-th order tensor form, and so on.

Beyond the homogeneous polynomial functions described above, we also study in this thesis the generic multivariate inhomogeneous polynomial function. An n-dimensional d-th degree polynomial function can be explicitly written as a summation of homogenous polynomial functions in decreasing degrees as follows

$$\text{Function P} \quad p(\boldsymbol{x}) := \sum_{k=1}^{d} f_k(\boldsymbol{x}) + f_0 = \sum_{k=1}^{d} F_k(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x}}_{k}) + f_0,$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $f_0 \in \mathbb{R}$, and $f_k(\boldsymbol{x}) = F_k(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x}}_{k})$ is a homogenous polynomial function of degree k for $k = 1, 2, \ldots, d$; and letter 'P' signifies the notion of *polynomial*. One natural way to deal with inhomogeneous polynomial function is through *homogenization*; that is, we introduce a new variable, to be denoted by x_h in this thesis, which is actually set to be 1, to yield a homogeneous form

$$p(\boldsymbol{x}) = \sum_{k=1}^{d} f_k(\boldsymbol{x}) + f_0 = \sum_{k=1}^{d} f_k(\boldsymbol{x}) {x_h}^{d-k} + f_0 {x_h}^d = f(\bar{\boldsymbol{x}}),$$

where $f(\bar{\boldsymbol{x}})$ is an $(n+1)$-dimensional d-th degree homogeneous polynomial function, with variable $\bar{\boldsymbol{x}} \in \mathbb{R}^{n+1}$. Throughout this thesis, the 'bar' notation over boldface lowercase letters, e.g., $\bar{\boldsymbol{x}}$, is reserved for an $(n+1)$-dimensional vector, with the underlying letter $\boldsymbol{x}$ referring to the vector of its first n components, and the subscript 'h' (the subscript of x_h) referring to its last component. For instance, if $\bar{\boldsymbol{x}} = (x_1, x_2, \cdots, x_n, x_{n+1})^{\mathrm{T}} \in \mathbb{R}^{n+1}$, then $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^{\mathrm{T}} \in \mathbb{R}^n$ and $x_h = x_{n+1} \in \mathbb{R}$.

Throughout we adhere to the notation F for a multilinear form (Function T) defined by a tensor form $\boldsymbol{F}$, and f for a homogenous polynomial (Function H) or a mixed homogeneous form (Function M), and p for a generic (inhomogeneous) polynomial function (Function P). Without loss of generality we assume that $n_1 \le n_2 \le \cdots \le n_d$ in the tensor form $\boldsymbol{F} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, and $n_1 \le n_2 \le \cdots \le n_s$ in the tensor form $\boldsymbol{F} \in \mathbb{R}^{{n_1}^{d_1} \times {n_2}^{d_2} \times \cdots \times {n_s}^{d_s}}$. We also assume at lease one component of the tensor form, $\boldsymbol{F}$ in Functions T, H, M, and $\boldsymbol{F}_d$ in Function P is nonzero to avoid triviality.

2.1.2 Constraint Sets

The most commonly used constraint sets for polynomial optimization problems are studied in this thesis. Specifically, we consider the following types of constraint sets:

$$
\begin{aligned}
&\text{Constraint B} && \left\{\boldsymbol{x} \in \mathbb{R}^n \,|\, {x_i}^2 = 1,\ i = 1, 2, \ldots, n\right\} =: \mathbb{B}^n;\\
&\text{Constraint } \bar{\text{B}} && \left\{\boldsymbol{x} \in \mathbb{R}^n \,|\, {x_i}^2 \le 1,\ i = 1, 2, \ldots, n\right\} =: \bar{\mathbb{B}}^n;\\
&\text{Constraint S} && \left\{\boldsymbol{x} \in \mathbb{R}^n \,|\, \|\boldsymbol{x}\| := \left({x_1}^2 + {x_2}^2 + \cdots + {x_n}^2\right)^{\frac{1}{2}} = 1\right\} =: \mathbb{S}^n;\\
&\text{Constraint } \bar{\text{S}} && \left\{\boldsymbol{x} \in \mathbb{R}^n \,|\, \|\boldsymbol{x}\| \le 1\right\} =: \bar{\mathbb{S}}^n;\\
&\text{Constraint Q} && \left\{\boldsymbol{x} \in \mathbb{R}^n \,|\, \boldsymbol{x}^{\mathrm{T}} \boldsymbol{Q}_i \boldsymbol{x} \le 1,\ i = 1, 2, \ldots, m\right\};\\
&\text{Constraint G} && \left\{\boldsymbol{x} \in \mathbb{R}^n \,|\, \boldsymbol{x} \in G\right\}.
\end{aligned}
$$

The notion 'B' signifies the *binary* variables or *binary* constraints, and 'S' signifies the Euclidean *spherical* constraint, with '$\bar{\text{B}}$' (hypercube) and '$\bar{\text{S}}$' (the Euclidean ball) signifying their *convex hulls* respectively. The norm notation '$\|\cdot\|$' in this thesis is the 2-norm (the Euclidean norm) unless otherwise specified, including those for vectors, matrices and tensors. In particular, the norm of the tensor $\boldsymbol{F} = (F_{i_1 i_2 \cdots i_d}) \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is defined as

$$
\|\boldsymbol{F}\| := \sqrt{\sum_{1 \le i_1 \le n_1, 1 \le i_2 \le n_2, \cdots, 1 \le i_d \le n_d} {F_{i_1 i_2 \cdots i_d}}^2}.
$$

The notion 'Q' signifies the *quadratic* constraints, and we focus on convex quadratic constraints in this thesis, or specifically the case of co-centered ellipsoids, i.e., $\boldsymbol{Q}_i \succeq 0$

for $i = 1, 2, \ldots, m$ and $\sum_{i=1}^{m} \boldsymbol{Q}_i \succ 0$. A *general* convex compact set in $\mathbb{R}^n$ is also discussed in this thesis, which is denoted by the notion 'G'. Constraints $\bar{\mathrm{B}}$, $\bar{\mathrm{S}}$, Q and G are convex, while Constraints B and S are non-convex. It is obvious that Constraint G is a generalization of Constraint Q, and Constraint Q is a generalization of Constraint $\bar{\mathrm{S}}$ and Constraint $\bar{\mathrm{B}}$ as well.

2.1.3 Models and Organization

All the polynomial optimization models discussed in this thesis are maximization problems, and the results for most of their minimization counterparts can be similarly derived. The names of all the models simply combine the names of the objective functions described in Section 2.1.1, and the names of the constraint sets described in Section 2.1.2, with the names of the constraints in the subscription. For examples, model (T_S) is to maximize a multilinear tensor function (Function T) under the spherical constraints (Constraint S), model (M_{BS}) is to maximize a mixed polynomial form (Function M) under binary constraints (Constraint B), mixed with variables under spherical constraints (Constraint S), etc.

In Chapter 3, we discuss the models for optimizing a multilinear form with quadratic constraints, including (T_S) and (T_Q). In Chapter 4, we discuss the models for optimizing a homogeneous polynomial or a mixed form with quadratic constraints, including (H_S), (H_Q), (M_S) and (M_Q). General polynomial optimization models including (P_S), (P_Q) and (P_G) are discussed in Chapter 5. Chapter 6 talk about binary integer programming models, including (T_B), (H_B), (M_B), and (P_B). Chapter 7 talk about mixed integer programming models, including (T_{BS}), (H_{BS}) and (M_{BS}). All these models are listed below for a quick reference.

Chapter 3:

$$
\begin{array}{lll}
(T_S) & \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\
& \text{s.t.} & \boldsymbol{x}^k \in \mathbb{S}^{n_k},\ k = 1, 2, \ldots, d; \\
(T_Q) & \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\
& \text{s.t.} & (\boldsymbol{x}^k)^{\mathrm{T}} \boldsymbol{Q}_{i_k}^k \boldsymbol{x}^k \le 1,\ k = 1, 2, \ldots, d,\ i_k = 1, 2, \ldots, m_k, \\
& & \boldsymbol{x}^k \in \mathbb{R}^{n_k},\ k = 1, 2, \ldots, d.
\end{array}
$$

Chapter 4:

$$
\begin{aligned}
(H_S)\quad & \max \quad f(\boldsymbol{x}) \\
& \text{s.t.} \quad \boldsymbol{x} \in \mathbb{S}^n; \\
(H_Q)\quad & \max \quad f(\boldsymbol{x}) \\
& \text{s.t.} \quad \boldsymbol{x}^{\mathrm{T}} \boldsymbol{Q}_i \boldsymbol{x} \le 1,\ i = 1, 2, \ldots, m, \\
& \qquad \boldsymbol{x} \in \mathbb{R}^n; \\
(M_S)\quad & \max \quad f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s) \\
& \text{s.t.} \quad \boldsymbol{x}^k \in \mathbb{S}^{n_k},\ k = 1, 2, \ldots, s; \\
(M_Q)\quad & \max \quad f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s) \\
& \text{s.t.} \quad (\boldsymbol{x}^k)^{\mathrm{T}} \boldsymbol{Q}_{i_k}^k \boldsymbol{x}^k \le 1,\ k = 1, 2, \ldots, s,\ i_k = 1, 2, \ldots, m_k, \\
& \qquad \boldsymbol{x}^k \in \mathbb{R}^{n_k},\ k = 1, 2, \ldots, s.
\end{aligned}
$$

Chapter 5:

$$
\begin{aligned}
(P_S)\quad & \max \quad p(\boldsymbol{x}) \\
& \text{s.t.} \quad \boldsymbol{x} \in \bar{\mathbb{S}}^n; \\
(P_Q)\quad & \max \quad p(\boldsymbol{x}) \\
& \text{s.t.} \quad \boldsymbol{x}^{\mathrm{T}} \boldsymbol{Q}_i \boldsymbol{x} \le 1,\ i = 1, 2, \ldots, m, \\
& \qquad \boldsymbol{x} \in \mathbb{R}^n; \\
(P_G)\quad & \max \quad p(\boldsymbol{x}) \\
& \text{s.t.} \quad \boldsymbol{x} \in G.
\end{aligned}
$$

Chapter 6:

$$
\begin{aligned}
(T_B)\quad & \max \quad F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\
& \text{s.t.} \quad \boldsymbol{x}^k \in \mathbb{B}^{n_k},\ k = 1, 2, \ldots, d; \\
(H_B)\quad & \max \quad f(\boldsymbol{x}) \\
& \text{s.t.} \quad \boldsymbol{x} \in \mathbb{B}^n; \\
(M_B)\quad & \max \quad f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s) \\
& \text{s.t.} \quad \boldsymbol{x}^k \in \mathbb{B}^{n_k},\ k = 1, 2, \ldots, s; \\
(P_B)\quad & \max \quad p(\boldsymbol{x}) \\
& \text{s.t.} \quad \boldsymbol{x} \in \mathbb{B}^n.
\end{aligned}
$$

Chapter 7:

$$
\begin{aligned}
(T_{BS}) \quad & \max \quad F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d, \boldsymbol{y}^1, \boldsymbol{y}^2, \ldots, \boldsymbol{y}^{d'}) \\
& \text{s.t.} \quad \boldsymbol{x}^k \in \mathbb{B}^{n_k},\ k = 1, 2, \ldots, d, \\
& \qquad \boldsymbol{y}^\ell \in \mathbb{S}^{m_\ell},\ \ell = 1, 2, \ldots, d'; \\
(H_{BS}) \quad & \max \quad f(\boldsymbol{x}, \boldsymbol{y}) \\
& \text{s.t.} \quad \boldsymbol{x} \in \mathbb{B}^n, \\
& \qquad \boldsymbol{y} \in \mathbb{S}^m; \\
(M_{BS}) \quad & \max \quad f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s, \boldsymbol{y}^1, \boldsymbol{y}^2, \cdots, \boldsymbol{y}^t) \\
& \text{s.t.} \quad \boldsymbol{x}^k \in \mathbb{B}^{n_k},\ k = 1, 2, \ldots, s, \\
& \qquad \boldsymbol{y}^\ell \in \mathbb{S}^{m_\ell},\ \ell = 1, 2, \ldots, t.
\end{aligned}
$$

As before, we also assume that the tensor forms of the objective functions in (H_{BS}) and (M_{BS}) to have partial symmetric property, $m_1 \leq m_2 \leq \cdots \leq m_{d'}$ in (T_{BS}), and $m_1 \leq m_2 \leq \cdots \leq m_t$ in (M_{BS}).

In each chapter mentioned above, we discuss the computational complexity of the models concerned, and focus on polynomial-time approximation algorithms with worst-case performance ratios, followed by discussions on their applications and/or numerical performance of the algorithms proposed. All the numerical computations are conducted using an Intel Pentium 4 CPU 2.80GHz computer with 2GB of RAM, and the supporting software Matlab 7.7.0 (R2008b). Let $d_1 + d_2 + \cdots + d_s = d$ and $d'_1 + d'_2 + \cdots + d'_t = d'$ in the above mentioned models. The degrees of the objective polynomials in these models, d and $d + d'$, are understood as fixed constants in our subsequent discussions. We are able to propose polynomial-time approximation algorithms for all these models, and the approximation ratios depend only on the dimensions (including the number of variables and the number of constraints) of the problems concerned.

The remaining sections in this chapter discuss some necessary preparations, for the purpose of better understanding the main subjects in the thesis. The topics include elementary introductions of tensor operations, approximation algorithms, randomized algorithms, and semidefinite programming.

2.2 Tensor Operations

A tensor is a multidimensional array. More formally, a d-th order tensor is an element of the tensor product of d vector spaces, each of which has its own coordinate system.

Each entry of a d-th order tensor has d indices associated. A first order tensor is a vector, a second order tensor is a matrix, and tensors of order three or higher are called higher order tensors.

This section describe a few tensor operations commonly used in this thesis. For a general review of other tensor operations, the reader is referred to [68]. The tensor inner product is denoted by '$\bullet$', which is the summation of products of all corresponding entries. For example, if $\boldsymbol{F}^1, \boldsymbol{F}^2 \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, then

$$\boldsymbol{F}^1 \bullet \boldsymbol{F}^2 := \sum_{1 \le i_1 \le n_1, 1 \le i_2 \le n_2, \cdots, 1 \le i_d \le n_d} F^1_{i_1 i_2 \cdots i_d} \cdot F^2_{i_1 i_2 \cdots i_d}.$$

As mentioned before, the norm of the tensor is then defined as $\|\boldsymbol{F}\| := \sqrt{\boldsymbol{F} \bullet \boldsymbol{F}}$. Notice that the tensor inner product and tensor norm also apply to the vectors and the matrices since they are lower order tensors.

The modes of a tensor are referred to its coordinate systems. For example, the following fourth order tensor $\boldsymbol{G} \in \mathbb{R}^{2 \times 2 \times 3 \times 2}$, with its entries being

$$\begin{array}{llllll} G_{1111} = 1, & G_{1112} = 2, & G_{1121} = 3, & G_{1122} = 4, & G_{1131} = 5, & G_{1132} = 6, \\ G_{1211} = 7, & G_{1212} = 8, & G_{1221} = 9, & G_{1222} = 10, & G_{1231} = 11, & G_{1232} = 12, \\ G_{2111} = 13, & G_{2112} = 14, & G_{2121} = 15, & G_{2122} = 16, & G_{2131} = 17, & G_{2132} = 18, \\ G_{2211} = 19, & G_{2212} = 20, & G_{2221} = 21, & G_{2222} = 22, & G_{2231} = 23, & G_{2232} = 24, \end{array}$$

has 4 modes, to be named mode 1, mode 2, mode 3 and mode 4. In case a tensor is a matrix, it has only two modes, which we usually called them column and row. The indices for an entry of a tensor are a sequence of integers, each one assigning from one mode.

The first widely used tensor operation is tensor rewritten, which appears frequently in this thesis. Namely, by combining a set of modes into one mode, a tensor can be rewritten as a new tensor with a lower order. For example, by combining modes 3 and 4 together and put it into the last mode of the new tensor, tensor $\boldsymbol{G}$ can be rewritten as a third order tensor $\boldsymbol{G}' \in \mathbb{R}^{2 \times 2 \times 6}$, with its entries being

$$\begin{array}{llllll} G'_{111} = 1, & G'_{112} = 2, & G'_{113} = 3, & G'_{114} = 4, & G'_{115} = 5, & G'_{116} = 6, \\ G'_{121} = 7, & G'_{122} = 8, & G'_{123} = 9, & G'_{124} = 10, & G'_{125} = 11, & G'_{126} = 12, \\ G'_{211} = 13, & G'_{212} = 14, & G'_{213} = 15, & G'_{214} = 16, & G'_{215} = 17, & G'_{216} = 18, \\ G'_{221} = 19, & G'_{222} = 20, & G'_{223} = 21, & G'_{224} = 22, & G'_{225} = 23, & G'_{226} = 24. \end{array}$$

By combining modes 2, 3 and 4 together, tensor $\boldsymbol{G}$ is then rewritten as a 2×12 matrix

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \end{bmatrix};$$

and by combing all the modes together, tensor $\boldsymbol{G}$ becomes a 24-dimensional vector $(1, 2, \ldots, 24)^{\mathrm{T}}$, which is the same as vectorization of a tensor.

The other commonly used operation of tensor is modes switch, that is to switch the positions of two modes. This is very much like the transpose of a matrix, switching the positions of row and column. Accordingly, the modes switch will change the sequences of indices for the entries of a tensor. For example, by switching mode 1 and mode 3 of $\boldsymbol{G}$, tensor $\boldsymbol{G}$ is then changed to $\boldsymbol{G}'' \in \mathbb{R}^{3\times 2\times 2\times 2}$, with its entries defined by

$$G''_{ijk\ell} := G_{kji\ell} \quad \forall\, j, k, \ell = 1, 2,\ i = 1, 2, 3.$$

By default, among all the tensors discussed in this thesis, we assume their modes have been switched (in fact reordered), so that their dimensions are in a non-decreasing order.

Another widely used operation is multiplying a tensor by a vector. For example, tensor $\boldsymbol{G}$ has its associated multilinear function $G(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{w})$, where variables $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w} \in \mathbb{R}^2$ and $\boldsymbol{z} \in \mathbb{R}^3$. Four modes in $\boldsymbol{G}$ correspond to the four positions of variables in function G. For a given vector $\hat{\boldsymbol{w}} = (\hat{w}_1, \hat{w}_2)^{\mathrm{T}}$, its multiplication with $\boldsymbol{G}$ in mode 4 makes $\boldsymbol{G}$ to be $\boldsymbol{G}''' \in \mathbb{R}^{2\times 2\times 3}$, whose entries are defined by

$$G'''_{ijk} := G_{ijk1}\hat{w}_1 + G_{ijk2}\hat{w}_2 \quad \forall\, i, j = 1, 2,\ k = 1, 2, 3,$$

which is basically the inner product of the vectors $\hat{\boldsymbol{w}}$ and $\boldsymbol{G}_{ijk\cdot} := (G_{ijk1}, G_{ijk2})^{\mathrm{T}}$. For examples, if $\hat{\boldsymbol{w}} = (1, 1)^{\mathrm{T}}$, then $\boldsymbol{G}'''$ has entries

$$\begin{aligned} &G'''_{111} = 3, \quad G'''_{112} = 7, \quad G'''_{113} = 11, \quad G'''_{121} = 15, \quad G'''_{122} = 19, \quad G'''_{123} = 23, \\ &G'''_{211} = 27, \quad G'''_{212} = 31, \quad G'''_{213} = 35, \quad G'''_{221} = 39, \quad G'''_{222} = 43, \quad G'''_{223} = 47. \end{aligned}$$

Its corresponding multilinear function is in fact $G(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \hat{\boldsymbol{w}})$, with the underling variables $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$. Sometimes, we use $G(\cdot, \cdot, \cdot, \hat{\boldsymbol{w}})$ more often to denote this new multilinear function $G(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \hat{\boldsymbol{w}})$.

This type of multiplication can extend to a tensor with a matrix, even with a tensor. For example, if we multiply tensor $\boldsymbol{G}$ by a given matrix $\hat{\boldsymbol{Z}} \in \mathbb{R}^{3\times 2}$ in modes 3 and 4,

then we get a second order tensor (matrix) in $\mathbb{R}^{2\times 2}$, whose (i,j)-th entry is

$$\boldsymbol{G}_{ij\cdot\cdot} \bullet \hat{\boldsymbol{Z}} = \sum_{k=1}^{3}\sum_{\ell=1}^{2} G_{ijk\ell}\hat{Z}_{k\ell} \quad \forall\, i,j = 1,2.$$

Its corresponding multilinear function is denoted by $G(\cdot,\cdot,\hat{\boldsymbol{Z}})$. In general, if a d-th order tensor multiply by a d'-th order tensor ($d' \leq d$) in appropriate modes, then its product is a $(d-d')$-th order tensor. In particular, if $d = d'$, then this multiplication is simply the tensor inner product.

2.3 Approximation Algorithms

Approximation algorithms are algorithms designed to find approximate solutions to optimization problems. In general, approximation algorithms are often associated with NP-hard problems, since it is unlikely that there exist polynomial-time exact algorithms for solving NP-hard problems, one then settles for polynomial-time sub-optimal solutions. Approximation algorithms are also used for problems where exact polynomial-time algorithms are possible but are too expensive to compute due to the size of the problem. Usually, an approximation algorithm is associated with an approximation ratio, which is a provable value measuring the quality of the solution found.

Approximation algorithms are widely used in combinatorial optimizations, typically in various graph problems. Let us describe a well known example, the *vertex cover* problem, to appreciate the notion of approximation algorithms. Given an undirected graph $G = (V, E)$, and a nonnegative cost associated with each vertex, find a minimum cost of vertices to cover all the edges, i.e., a set $V' \subset V$ such that every edge has at least one endpoint incident at V'.

The vertex cover problem is NP-hard (see e.g., [38]), even for the cardinality vertex cover, which is the case that the cost associated with each vertex is 1. There is a very simple algorithm for cardinality vertex cover problem. Pick any uncovered edge $e \in E$, select both of its two incident vertices, and then remove all the edges covered by these two vertices; The process is continued until every edge is removed, and output all the selected vertices. This process can be done in at most $|E|$ numbers of steps, which is polynomial-time of the input dimension $\max\{|V|, |E|\}$. The algorithm may not get an optimal cover, however we can show that the number of vertices selected by this algorithm is at most twice as the optimal value of cardinality vertex cover. If fact,

for any optimal cover, it must cover any edge picked (not removed) by the algorithm, and thus must include one of its two incident vertices, then this optimal cover must include half of the vertices selected by the algorithm. This is a typical approximation algorithm with approximation ratio 2.

We shall now define formally the approximation algorithms and approximation ratios. Throughout this thesis, for any maximization problem (P) defined as $\max_{\boldsymbol{x}\in X} p(\boldsymbol{x})$, we use $v(P)$ to denote its optimal value, and $\underline{v}(P)$ to denote the optimal value of its minimization counterpart, i.e.,

$$v(P) := \max_{\boldsymbol{x}\in X} p(\boldsymbol{x}) \quad \text{and} \quad \underline{v}(P) := \min_{\boldsymbol{x}\in X} p(\boldsymbol{x}).$$

Definition 2.3.1 *Approximation algorithm and approximation ratio:*

1. *A maximization problem* $\max_{\boldsymbol{x}\in X} p(\boldsymbol{x})$ *admits a polynomial-time approximation algorithm with approximation ratio* $\tau \in (0,1]$, *if* $v(P) \geq 0$ *and a feasible solution* $\hat{\boldsymbol{x}} \in X$ *can be found in polynomial-time such that* $p(\hat{\boldsymbol{x}}) \geq \tau\, v(P)$;

2. *A minimization problem* $\min_{\boldsymbol{x}\in X} p(\boldsymbol{x})$ *admits a polynomial-time approximation algorithm with approximation ratio* $\mu \in [1,\infty)$, *if* $\underline{v}(P) \geq 0$ *and a feasible solution* $\hat{\boldsymbol{x}} \in X$ *can be found in polynomial-time such that* $p(\hat{\boldsymbol{x}}) \leq \mu\, \underline{v}(P)$.

It is easy to see that the larger the τ, the better the ratio for a maximization problem, and the smaller the μ, the better the ratio for a minimization problem. In short the closer to one, the better the ratio. However, sometimes a problem may be very hard, such that there is no polynomial-time approximation algorithm which approximates the optimal value within any positive factor. A typical example of this type is also the vertex cover problem, although its cardinality version has a very simple 2-approximation algorithm. In those unfortunate cases, we have approximation algorithms with *relative* approximation ratios.

Definition 2.3.2 *Approximation algorithm and relative approximation ratio:*

1. *A maximization problem* $\max_{\boldsymbol{x}\in X} p(\boldsymbol{x})$ *admits a polynomial-time approximation algorithm with relative approximation ratio* $\tau \in (0,1]$, *if a feasible solution* $\hat{\boldsymbol{x}} \in X$ *can be found in polynomial-time such that* $p(\hat{\boldsymbol{x}}) - \underline{v}(P) \geq \tau\,(v(P) - \underline{v}(P))$, *or equivalently* $v(P) - p(\hat{\boldsymbol{x}}) \leq (1-\tau)\,(v(P) - \underline{v}(P))$;

2. *A minimization problem* $\min_{\boldsymbol{x}\in X} p(\boldsymbol{x})$ *admits a polynomial-time approximation algorithm with relative approximation ratio* $\mu \in [1, \infty)$, *if a feasible solution* $\hat{\boldsymbol{x}} \in X$ *can be found in polynomial-time such that* $v(P) - p(\hat{\boldsymbol{x}}) \geq (1/\mu)\,(v(P) - \underline{v}(P))$, *or equivalently* $p(\hat{\boldsymbol{x}}) - \underline{v}(P) \leq (1 - 1/\mu)\,(v(P) - \underline{v}(P))$.

Similar to the usual approximation ratio, the closer to one, the better the relative approximation ratios. For a maximization problem, if we know for sure that the optimal value of its minimization counterpart is nonnegative, then trivially a relative approximation ratio already implies a usual approximation ratio. This is not rare, as many optimization problems always have nonnegative objective functions in real applications, e.g., various graph partition problems. Of course there are several other ways in defining the approximation quality to measure the performance of the approximate solutions (see e.g., [61, 11]).

We would like to point out that the approximation ratios defined are for the worst-case scenarios, which might be hard or even impossible to find an example attaining exactly the ratio in applying the algorithms. Thus it does not mean an approximation algorithm with a better approximation ratio has better performance in practice than that with a worse ratio. In reality, many approximation algorithms have their approximation ratios far from one if they have one at all, which might approach zero when the dimensions of the problems become large. Perhaps it is more appropriate to view the approximation guarantee as a measure that forces us to explore deeper into the structure of the problem and discover more powerful tools to explore this structure. In addition, an algorithm with a theoretical assurance should be viewed as a useful guidance that can be fine tuned to suit the type of instances arising from that specific applications.

As mentioned in Section 2.1.3, all optimization models considered in this thesis are maximization problems. Thus we reserve the Greek letter τ, specialized to indicate the approximation ratio, which is a key ingredient throughout this thesis. All the approximation ratios presented in this thesis are in general not universal constants, and involve problem dimensions and Ω. Here $\Omega(f(n))$ signifies that there are positive universal constants α and n_0 such that $\Omega(f(n)) \geq \alpha f(n)$ for all $n \geq n_0$. As usual, $O(f(n))$ signifies that there are positive universal constants α and n_0 such that $O(f(n)) \leq \alpha f(n)$ for all $n \geq n_0$.

2.4 Randomized Algorithms

A randomized algorithm is an algorithm which employs a degree of randomness as part of its operation. The algorithm typically contains certain probability distribution as an auxiliary input to guide its executions, in the hope of achieving good performance on *average*, or with *high probability* to achieve good performance. Formally, the algorithm's performance will be a random variable, thus either the running time, or the output (or both) are random variables.

Historically, the first randomized algorithm was a method developed by Rabin [104] for the closest pair problem in computational geometry. The study of randomized algorithms was spurred by the 1977 discovery of a randomized primality test (determining the primality of a number) by Solovay and Strassen [111]. Soon afterwards Rabin [105] demonstrated that the 1976 Miller's primality test [83] can be turned into a randomized algorithm. At that time, no practical deterministic algorithm for primality was known.

A well known application and commonly used algorithm in which randomness can be useful is quicksort. Any deterministic version of this algorithm requires $O(n^2)$ time to sort n different numbers (to be denoted by set S), e.g., the straightforward one: comparing all the pairs requiring $\frac{n(n-1)}{2}$ time. However, if we assume the given n different numbers are in a sequence uniformly distributed on all the $n!$ number of distinctive sequences, then the quicksort algorithm sort this sequence in $O(n \log n)$ time. The algorithm chooses an element of S uniformly at random as a pivot, compares the pivot with other elements and groups them into two sets S_1 (those bigger than the pivot) and S_2 (those smaller than the pivot), and then applies the same process to sort S_1 and S_2; This process is continued until a sort realizes.

To see why quicksort will cost $O(n \log n)$ in average, let us without loss of generality assume $S = \{1, 2, \ldots, n\}$. Denote x_{ij} to be the indicator random variable whether elements i and j are compared or not during a quicksort. The total time of compares is then

$$\mathrm{E}\left[\sum_{1 \le i < j \le n} x_{ij}\right] = \sum_{1 \le i < j \le n} \mathrm{E}\left[x_{ij}\right].$$

Next we compute $\mathrm{E}[x_{ij}]$, which is the probability that i and j are ever compared by a quicksort. This happens if and only if either i or j is the first pivot selected by quicksort from the set $\{i, i+1, \ldots, j-1, j\}$ (assume $i < j$), and the probability is $2/(j-i-1)$.

Therefore, the average time of compares is

$$\sum_{1\le i<j\le n} \mathrm{E}[x_{ij}] = \sum_{1\le i<j\le n} \frac{2}{j-i+1} = 2(n+1)\sum_{k=1}^{n}\frac{1}{k} - 4n = O(n\log n).$$

In fact, the expected time argument is based on the uniformly choosing the pivots, although the worst-case time is also $O(n^2)$, when each time we are very unfortunate to pick the largest element as the pivot.

There are also other successful stories that randomized algorithms help. For example, the volume of a convex body can be estimated by a randomized algorithm to arbitrary precision in polynomial-time [31], while no deterministic algorithm can do the same [13].

In applying to NP-hard optimization problems, randomized algorithms are often associated with approximation algorithms to prove performance ratios, in terms of expectation, or with high probability. A simple example is a randomized approximation algorithm with approximation ratio 0.5 for the *max-cut* problem. Given an undirected graph $G = (V, E)$, and a nonnegative weight associated with each edge, find a partition (cut) of V into two disjoint sets A and B, so that the total weight of all the edges connecting one vertex in A and one vertex B is maximized. This is also one of the well known NP-hard problems [38]. The simple algorithm is as follows. For each vertex, independently toss a fair coin, and put it into A if head or B if tail. It is easy to see that the probability of an edge connecting A and B are exactly $1/2$. By the linearity of expectation, the expected total weight of this cut is exactly half of the total weight of all the edges, which is at least half of the maximum cut.

The current best approximation ratio of the max-cut problem is 0.878, another celebrated result of randomized algorithm due to Goemans and Williamson [40], which we shall elaborate in Section 2.5.

We conclude this section by defining polynomial-time randomized approximation algorithms, like Definition 2.3.1 and Definition 2.3.2. We are not going to elaborate all of their randomized versions, rather a typical one. The others can be similar described.

Definition 2.4.1 *A maximization problem* $\max_{\boldsymbol{x}\in X} p(\boldsymbol{x})$ *admits a polynomial-time randomized approximation algorithm with approximation ratio* $\tau \in (0, 1]$, *if* $v(P) \ge 0$ *and one of the following two facts holds:*

1. *A feasible solution* $\hat{\boldsymbol{x}} \in X$ *can be found in polynomial-time, such that* $\mathrm{E}[p(\hat{\boldsymbol{x}})] \ge \tau\, v(P)$;

2. *A feasible solution* $\hat{\boldsymbol{x}} \in X$ *can be found in polynomial-time, such that* $p(\hat{\boldsymbol{x}}) \geq \tau\, v(P)$ *with probability at least* $1-\epsilon$ *for all* $\epsilon \in (0,1)$.

2.5 Semidefinite Programming

Semidefinite programming (SDP) is a subfield of convex optimization concerned with the optimization of a linear objective function over the intersection of the cone of positive semidefinite matrices and an affine space. It can be viewed as an extension of the well known linear programming problem, where the vector of variables is replaced by a symmetric matrix, and the cone of nonnegative orthant is replaced by the cone of positive semidefinite matrices. It is a special case of the so-called conic programming problems (specialized to the cone of positive semidefinite matrices).

The standard formulation of an SDP problem is

$$\begin{array}{lll} (PSP) & \sup & \boldsymbol{C} \bullet \boldsymbol{X} \\ & \text{s.t.} & \boldsymbol{A}_i \bullet \boldsymbol{X} = b_i,\ i = 1, 2, \ldots, m, \\ & & \boldsymbol{X} \succeq 0, \end{array}$$

where the data $\boldsymbol{C}$ and $\boldsymbol{A}_i\ (i = 1, 2, \ldots, m)$ are symmetric matrices, $b_i\ (i = 1, 2, \ldots, m)$ are scalars, the dot product '$\bullet$' is the usual matrix inner product introduced in Section 2.2, and '$\boldsymbol{X} \succeq 0$' means matrix $\boldsymbol{X}$ is positive semidefinite.

The dual problem of (PSP) is

$$\begin{array}{lll} (DSP) & \inf & \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} \\ & \text{s.t.} & \sum_{i=1}^{m} y_i \boldsymbol{A}_i + \boldsymbol{Z} = \boldsymbol{C}, \\ & & \boldsymbol{Z} \succeq 0. \end{array}$$

A solution for an SDP problem is called *strictly feasible* if its feasible region has nonempty interior, which is also called *Slater condition*. We are now providing the strong duality theorem of SDP, for its proof one is refereed to Vandenberghe and Boyd [116] and Helmberg [51].

Theorem 2.5.1 *The followings hold for* (PSP) *and* (DSP)*:*

1. *If* (DSP) *is strictly feasible, then* $v(PSP) = v(DSP)$. *If in addition* (DSP) *is bounded above, then this optimal value is obtained by a feasible* $\boldsymbol{X}^*$ *of* (PSP);

2. *If* (PSP) *is strictly feasible, then* $v(PSP) = v(DSP)$. *If in addition* (PSP) *is bounded below, then this optimal value is obtained by a feasible* $(\boldsymbol{Z}^*, \boldsymbol{y}^*)$ *of* (DSP);

3. *Suppose one of* (PSP) *and* (DSP) *is strictly feasible and has bounded optimal value, then feasible* $\boldsymbol{X}$ *of* (PSP) *and feasible* $(\boldsymbol{Z}, \boldsymbol{y})$ *of* (DSP) *is a pair of optimal solutions to its respective problems, if and only if* $\boldsymbol{C} \bullet \boldsymbol{X} = \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y}$ *or* $\boldsymbol{X} \bullet \boldsymbol{Z} = 0$;

4. *If both* (PSP) *and* (DSP) *are strictly feasible, then* $v(PSP) = v(DSP)$ *and this optimal value is obtained by feasible* $\boldsymbol{X}^*$ *of* (PSP) *and* $(\boldsymbol{Z}^*, \boldsymbol{y}^*)$ *of* (DSP).

For convenience, an SDP problem may often be specified in a slightly different, but equivalent form. For example, linear expressions involving nonnegative scalar variables may be added to the program specification. This remains an SDP because each variable can be incorporated into the matrix $\boldsymbol{X}$ as a diagonal entry (X_{ii} for some i). To ensure that $X_{ii} \geq 0$, constraints $X_{ij} = 0$ can be added for all $i \neq j$. As another example, note that for any $n \times n$ positive semidefinite matrix $\boldsymbol{X}$, there exists a set of vectors $\{\boldsymbol{v}^1, \boldsymbol{v}^2, \cdots, \boldsymbol{v}^n\}$ such that $X_{ij} = (\boldsymbol{v}^i)^{\mathrm{T}}\boldsymbol{v}^j$ for all $1 \leq i, j \leq n$. Therefore, SDP problems are often formulated in terms of linear expressions on scalar products of vectors. Given the solution for the SDP in the standard form, the vectors $\{\boldsymbol{v}^1, \boldsymbol{v}^2, \cdots, \boldsymbol{v}^n\}$ can be recovered in $O(n^3)$ time, e.g., using the Cholesky decomposition of $\boldsymbol{X}$.

There are several types of algorithms for solving SDP problems. These algorithms output the solutions up to an additive error ϵ in a time that is polynomial in the problem dimensions and $\ln(1/\epsilon)$. Interior point methods are the most popular and widely use one. A lot of efficient SDP solvers based on interior point methods have been developed, including SeDuMi of Sturm [112], SDPT3 of Toh et al. [115], SDPA of Fujisawa et al. [37], CSDP of Borchers [19], DSDP of Benson and Ye [17], and so on.

SDP is of growing interest for several reasons. Many practical problems in operations research and combinatorial optimization can be modeled or approximated as SDP problems. In automatic control theory, SDP is used in the context of linear matrix inequalities. All linear programming problems can be expressed as SDP problems, and via hierarchies of SDP problems the solutions of polynomial optimization problems can be approximated. Besides, SDP has been used in the design of optimal experiments and it can aid in the design of quantum computing circuits.

SDP has a wide range of practical applications. One of its significant applications is in its role to design approximate solutions to combinatorial optimization problems, starting from the seminal work by Goemans and Williamson [40], who essentially proposed a polynomial-time randomized approximation algorithm with approximation

ratio 0.878 for the max-cut problem. The algorithm uses SDP relaxation and randomization techniques, whose ideas have been revised and generalized in solving various quadratic programming problems [88, 118, 119, 87, 120, 24, 5, 121, 75, 50] and even quartic polynomial optimizations [77, 73]. We now elaborates the max-cut algorithm of Goemans and Williamson.

As described in Section 2.4, the max-cut problem is to find a partition of an undirected graph $G = (V, E)$ with nonnegative weights on edges, into two disjoint sets, so that the total weight of all the edges connecting these two sets is maximized. Denote $\{1, 2, \ldots, n\}$ to be the set of vertices. Let $w_{ij} \geq 0$ be the weight of edge connecting vertices i and j for all $i \neq j$, and let it be 0 if there is no edge between i and j, or $i = j$. If we let $x_i \, (i = 1, 2, \ldots, n)$ be the binary variable denoting whether it is in the first set ($x_i = 1$) or the second set ($x_i = -1$), then max-cut is the following quadratic integer programming problem

$$\begin{array}{lll} (MC) & \max & \sum_{1 \leq i,j \leq n} w_{ij}(1 - x_i x_j)/4 \\ & \text{s.t.} & x_i \in \{1, -1\}, \, i = 1, 2, \ldots, n. \end{array}$$

The problem is NP-hard (see e.g., Garey and Johnson [38]). Now by introducing a matrix $\boldsymbol{X}$ with X_{ij} replacing $x_i x_j$, the constraint is then equivalent to $\mathrm{diag}\,(\boldsymbol{X}) = \boldsymbol{e}$, $\boldsymbol{X} \succeq 0$, $\mathrm{rank}\,(\boldsymbol{X}) = 1$. A straightforward SDP relaxation is dropping the rank-one constraint, which yields

$$\begin{array}{lll} (SMC) & \max & \sum_{1 \leq i,j \leq n} w_{ij}(1 - X_{ij})/4 \\ & \text{s.t.} & \mathrm{diag}\,(\boldsymbol{X}) = \boldsymbol{e}, \, \boldsymbol{X} \succeq 0. \end{array}$$

The algorithm first solves (SMC) to get an optimal solution $\boldsymbol{X}^*$, then randomly generates an n-dimensional vector following a zero-mean multivariate normal distribution

$$\boldsymbol{\xi} \sim \mathcal{N}(\boldsymbol{0}_n, \boldsymbol{X}^*),$$

and lets $\hat{x}_i = \mathrm{sign}\,(\xi_i)$ for $i = 1, 2, \ldots, n$. Note that generating a zero-mean normal random vector with co-variance matrix $\boldsymbol{X}^*$ can be done by multiplying $(\boldsymbol{X}^*)^{\frac{1}{2}}$ with a vector whose components are generating from n i.i.d. standard normal random variables. Besides, the sign function takes 1 for nonnegative numbers and -1 for negative numbers. Although the output cut (solution $\hat{\boldsymbol{x}}$) may not be optimal, and is random either. It can be shown that

$$\mathrm{E}\,[\hat{x}_i \hat{x}_j] = \frac{2}{\pi} \arcsin X^*_{ij} \quad \forall \, 1 \leq i, j \leq n,$$

which further leads to

$$\mathrm{E}\left[\sum_{1\le i,j\le n}\frac{w_{ij}(1-\hat{x}_i\hat{x}_j)}{4}\right] \ge 0.878\, v(SMC) \ge 0.878\, v(MC).$$

This yields a 0.878-approximation ratio for the max-cut problem. The ratio significantly improves the previous best known one, which is 0.5 introduced in Section 2.4.

We concludes this section as well as this chapter, by introducing another example of SDP relaxation and randomization technique for solving quadratic constrained quadratic programming (QCQP) in Nemirovski et al. [87]. The problem is

$$\begin{array}{lll} (QP) & \max & \boldsymbol{x}^{\mathrm{T}}\boldsymbol{F}\boldsymbol{x} \\ & \text{s.t.} & \boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x} \le 1,\ i=1,2,\ldots,m, \\ & & \boldsymbol{x}\in\mathbb{R}^n, \end{array}$$

where $\boldsymbol{Q}_i \succeq 0$ for $i = 1, 2, \ldots, m$ and $\sum_{i=1}^m \boldsymbol{Q}_i \succ 0$. Remark this is exact the model (H_Q) when $d = 2$, whose algorithm will be used in this thesis. By using the same method with X_{ij} to replace $x_i x_j$, and drop the rank-one constraint, we shall have the standard SDP relaxation for (QP)

$$\begin{array}{lll} (SQP) & \max & \boldsymbol{F}\bullet\boldsymbol{X} \\ & \text{s.t.} & \boldsymbol{Q}_i\bullet\boldsymbol{X} \le 1,\ i=1,2,\ldots,m, \\ & & \boldsymbol{X}\succeq 0. \end{array}$$

A polynomial-time randomized approximation algorithm runs in as follows:

1. Solve (SQP) to get an optimal solution $\boldsymbol{X}^*$;
2. Randomly generate a vector $\boldsymbol{\xi}\sim\mathcal{N}(\boldsymbol{0}_n, \boldsymbol{X}^*)$;
3. Compute $t = \max_{1\le i\le m}\sqrt{\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{\xi}}$ and output the solution $\hat{\boldsymbol{x}} = \boldsymbol{\xi}/t$.

A probability analysis can prove that

$$\hat{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{F}\hat{\boldsymbol{x}} \ge \Omega\left(1/\log m\right) v(SQP) \ge \Omega\left(1/\log m\right) v(QP)$$

holds with probability bigger than a constant. Thus running this algorithms $O\left(\log(1/\epsilon)\right)$ times and pick the best solution, which shall hit the approximation bound of $\Omega\left(1/\log m\right)$ with probability at least $1-\epsilon$. For details, one is referred to Nemirovski et al. [87] and He et al. [50].

Chapter 3

Multilinear Form Optimization with Quadratic Constraints

3.1 Introduction

The first subclass of polynomial optimization problems studied in this thesis are the following multilinear tensor function optimizations over quadratic constraints. Specifically, the models include maximizing a multilinear form under spherical constraints

$$\boxed{\begin{array}{lll} (T_S) & \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\ & \text{s.t.} & \boldsymbol{x}^k \in \mathbb{S}^{n_k},\ k = 1, 2, \ldots, d, \end{array}}$$

and maximizing a multilinear form over co-centered ellipsoidal constraints

$$\boxed{\begin{array}{lll} (T_Q) & \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\ & \text{s.t.} & (\boldsymbol{x}^k)^{\mathrm{T}} \boldsymbol{Q}_{i_k}^k \boldsymbol{x}^k \le 1,\ k = 1, 2, \ldots, d,\ i_k = 1, 2, \ldots, m_k, \\ & & \boldsymbol{x}^k \in \mathbb{R}^{n_k},\ k = 1, 2, \ldots, d, \end{array}}$$

where $\boldsymbol{Q}_{i_k}^k \succeq 0$ and $\sum_{i_k=1}^{m_k} \boldsymbol{Q}_{i_k}^k \succ 0$ for $k = 1, 2, \ldots, d$, $i_k = 1, 2, \ldots, m_k$.

It is easy to see that the optimal value of (T_S), denoted by $v(T_S)$, is positive by the assumption that $\boldsymbol{F}$ is not a zero tensor. Moreover, (T_S) is equivalent to

$$\begin{array}{lll} \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\ \text{s.t.} & \boldsymbol{x}^k \in \bar{\mathbb{S}}^{n_k},\ k = 1, 2, \ldots, d. \end{array}$$

This is because we can always scale the decision variables such that $\|\boldsymbol{x}^k\| = 1$ for all $1 \le k \le d$ without decreasing the objective. Thus (T_S) is a special case of (T_Q).

Homogeneous polynomial functions play an important role in approximation theory. In a certain well-defined sense, homogeneous polynomials are fairly dense among all the continuous functions (see e.g., [117, 69]). Multilinear form is a special class of homogeneous polynomials. In fact, one of the main reasons for us to study multilinear form optimizations is its strong connection to homogenous polynomial optimizations in deriving approximation bounds, whose details will be discussed in Chapter 4. This connection creates a new approach to handle polynomial optimization problems, and the fundamental issue is optimization of a multilinear tensor form. Chen et al. [25] establish the tightness result of multilinear form relaxation for maximizing a homogeneous polynomial over spherical constraint. The study of multilinear form optimizations becomes much important.

Low degree cases of (T_S) can be often encountered: When $d = 1$, its optimal solution is $\boldsymbol{F}/\|\boldsymbol{F}\|$ due to the Cauchy-Schwartz inequality; When $d = 2$, (T_S) is to compute the spectrum norm of the matrix $\boldsymbol{F}$ with efficient algorithms readily available. As we shall prove later that (T_S) is already NP-hard when $d = 3$, the focus of this chapter is to design polynomial-time approximation algorithms with worst-case performance ratios for any fixed degree d. The novel idea to handle high degree multilinear form is to reduce the its degree, which leads to a relaxed multilinear form optimization in a lower degree case. As any matrix can be treated as a long vector, any higher order tensor can also be rewritten as a tensor with its order deduced by one (see the tensor operation in Section 2.2), and thus rewritten its corresponding multilinear form with its degree deduced by one. After we solve the problem with a lower degree, we need decompose the solution to make it feasible for the higher degree case. Thus specific decomposition methods are required, which are the main contributions in this chapter.

For the model (T_Q): When $d = 1$, it can be formulated to a second order cone program (SOCP), which can be solved in polynomial-time (see e.g., [20, 86]); When $d = 2$, it can be formulated to a quadratically constrained quadratic programming problem discussed in Section 2.5, and known to be NP-hard in general. Nemirovski et al. [87] proposed a polynomial-time randomized approximation algorithm with approximation ratio $\Omega(1/\log m)$ based on SDP relaxation and randomization, and this algorithm serves as a basis in analyzing our algorithms and approximation ratios.

We discuss approximation algorithms of (T_S) in Section 3.2, followed by that of (T_Q) in Section 3.3. Some application examples of the models concerned are discussed

in Section 3.4. Finally, numerical performance of the proposed algorithms are reported in Section 3.5.

3.2 Multilinear Form with Spherical Constraints

Let us first consider the following optimization model

$$\boxed{\begin{array}{lll}(T_S) & \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\ & \text{s.t.} & \boldsymbol{x}^k \in \mathbb{S}^{n_k},\ k = 1, 2, \ldots, d,\end{array}}$$

where $n_1 \le n_2 \le \cdots \le n_d$. A special case of (T_S) is worth noting, which plays an important role in our algorithms.

Proposition 3.2.1 *If $d = 2$, then (T_S) can be solved in polynomial-time, with $v(T_S) \ge \|\boldsymbol{F}\|/\sqrt{n_1}$.*

Proof. The problem is essentially $\max_{\boldsymbol{x}\in\mathbb{S}^{n_1}, \boldsymbol{y}\in\mathbb{S}^{n_2}} \boldsymbol{x}^{\mathrm{T}}\boldsymbol{F}\boldsymbol{y}$. For any fixed $\boldsymbol{y}$, the corresponding optimal $\boldsymbol{x}$ must be $\boldsymbol{F}\boldsymbol{y}/\|\boldsymbol{F}\boldsymbol{y}\|$ due to the Cauchy-Schwartz inequality, and accordingly,

$$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{F}\boldsymbol{y} = \left(\frac{\boldsymbol{F}\boldsymbol{y}}{\|\boldsymbol{F}\boldsymbol{y}\|}\right)^{\mathrm{T}} \boldsymbol{F}\boldsymbol{y} = \|\boldsymbol{F}\boldsymbol{y}\| = \sqrt{\boldsymbol{y}^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}\boldsymbol{y}}.$$

Thus the problem is equivalent to $\max_{\boldsymbol{y}\in\mathbb{S}^{n_2}} \boldsymbol{y}^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}\boldsymbol{y}$, whose solution is the largest eigenvalue and a corresponding eigenvector of the positive semidefinite matrix $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}$. We then have

$$\lambda_{\max}(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}) \ge \mathrm{tr}\,(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F})/\mathrm{rank}\,(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}) \ge \|\boldsymbol{F}\|^2/n_1,$$

which implies $v(T_S) = \sqrt{\lambda_{\max}(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F})} \ge \|\boldsymbol{F}\|/\sqrt{n_1}$. □

However, for any degree $d \ge 3$, (T_S) becomes NP-hard.

Proposition 3.2.2 *If $d = 3$, then (T_S) is NP-hard.*

Proof. We first quote a result of Nesterov [90], which states that

$$\begin{array}{lll} & \max & \sum_{k=1}^m (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_k\boldsymbol{x})^2 \\ & \text{s.t.} & \boldsymbol{x} \in \mathbb{S}^n \end{array}$$

is NP-hard. Now in a special case $d = 3$, $n_1 = n_2 = n_3 = n$ and $\boldsymbol{F} \in \mathbb{R}^{n^3}$ satisfies $F_{ijk} = F_{jik}$ for all $1 \le i, j, k \le n$, the objective function of (T_S) can be written as

$$F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \sum_{i,j,k=1}^n F_{ijk}x_iy_jz_k = \sum_{k=1}^n z_k \left(\sum_{i,j=1}^n F_{ijk}x_iy_j\right) = \sum_{k=1}^n z_k(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_k\boldsymbol{y}),$$

where symmetric matrix $\boldsymbol{A}_k \in \mathbb{R}^{n\times n}$ with its (i,j)-th entry being F_{ijk} for all $1 \le i,j,k \le n$. By the Cauchy-Schwartz inequality, (T_S) is equivalent to

$$\begin{array}{ll} \max & \sum_{k=1}^n (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_k\boldsymbol{y})^2 \\ \text{s.t.} & \boldsymbol{x},\boldsymbol{y} \in \mathbb{S}^n. \end{array}$$

We need only to show that the optimal value of the above problem is always attainable at $\boldsymbol{x} = \boldsymbol{y}$. To see why, denote $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ to be any optimal solution pair, with optimal value $v^\star$. If $\hat{\boldsymbol{x}} = \pm\hat{\boldsymbol{y}}$, then the claim is true; otherwise, we may suppose that $\hat{\boldsymbol{x}} + \hat{\boldsymbol{y}} \neq \boldsymbol{0}$. Let us denote $\hat{\boldsymbol{w}} := (\hat{\boldsymbol{x}} + \hat{\boldsymbol{y}})/\|\hat{\boldsymbol{x}} + \hat{\boldsymbol{y}}\|$. Since $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ must be a KKT point, there exist (λ, μ) such that

$$\begin{cases} \sum_{k=1}^n \hat{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{A}_k\hat{\boldsymbol{y}}\,\boldsymbol{A}_k\hat{\boldsymbol{y}} = \lambda\hat{\boldsymbol{x}} \\ \sum_{k=1}^n \hat{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{A}_k\hat{\boldsymbol{y}}\,\boldsymbol{A}_k\hat{\boldsymbol{x}} = \mu\hat{\boldsymbol{y}}. \end{cases}$$

Pre-multiplying $\hat{\boldsymbol{x}}^{\mathrm{T}}$ to the first equation and $\hat{\boldsymbol{y}}^{\mathrm{T}}$ to the second equation yield $\lambda = \mu = v^\star$. Summing up the two equations, pre-multiplying $\hat{\boldsymbol{w}}^{\mathrm{T}}$, and then scaling, lead us to

$$\sum_{k=1}^n \hat{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{A}_k\hat{\boldsymbol{y}}\,\hat{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{A}_k\hat{\boldsymbol{w}} = v^\star.$$

By applying the Cauchy-Schwartz inequality to the above equality, we have

$$v^\star \le \left(\sum_{k=1}^n (\hat{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{A}_k\hat{\boldsymbol{y}})^2\right)^{\frac{1}{2}} \left(\sum_{k=1}^n (\hat{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{A}_k\hat{\boldsymbol{w}})^2\right)^{\frac{1}{2}} = \sqrt{v^\star}\left(\sum_{k=1}^n (\hat{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{A}_k\hat{\boldsymbol{w}})^2\right)^{\frac{1}{2}},$$

which implies that $(\hat{\boldsymbol{w}}, \hat{\boldsymbol{w}})$ is also an optimal solution. The problem is then reduced to Nesterov's quartic model, and its NP-hardness thus follows. □

In the remainder of this section, we focus on approximation algorithms for (T_S) for general degree d. To illustrate the main idea of the algorithms, let us first work with the case $d = 3$, i.e.,

$$\begin{array}{lll} (\hat{T}_S) & \max & F(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = \sum_{1\le i\le n_1, 1\le j\le n_2, 1\le k\le n_3} F_{ijk}x_iy_jz_k \\ & \text{s.t.} & \boldsymbol{x} \in \mathbb{S}^{n_1},\ \boldsymbol{y} \in \mathbb{S}^{n_2},\ \boldsymbol{z} \in \mathbb{S}^{n_3}. \end{array}$$

Denote $\boldsymbol{W} = \boldsymbol{x}\boldsymbol{y}^{\mathrm{T}}$, and we have

$$\|\boldsymbol{W}\|^2 = \mathrm{tr}\,(\boldsymbol{W}\boldsymbol{W}^{\mathrm{T}}) = \mathrm{tr}\,(\boldsymbol{x}\boldsymbol{y}^{\mathrm{T}}\boldsymbol{y}\boldsymbol{x}^{\mathrm{T}}) = \mathrm{tr}\,(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}\boldsymbol{y}^{\mathrm{T}}\boldsymbol{y}) = \|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2 = 1.$$

Model $(\hat{T}_S)$ can now be relaxed to

$$\begin{array}{ll} \max & F(\boldsymbol{W},\boldsymbol{z}) = \sum_{1\le i\le n_1, 1\le j\le n_2, 1\le k\le n_3} F_{ijk}W_{ij}z_k \\ \text{s.t.} & \boldsymbol{W} \in \mathbb{S}^{n_1\times n_2},\ \boldsymbol{z} \in \mathbb{S}^{n_3}. \end{array}$$

Notice that the above problem is exactly (T_S) with $d = 2$, which can be solved in polynomial-time by Proposition 3.2.1. Denote its optimal solution to be $(\hat{\boldsymbol{W}}, \hat{\boldsymbol{z}})$. Clearly $F(\hat{\boldsymbol{W}}, \hat{\boldsymbol{z}}) \geq v(\hat{T}_S)$. The key step is to recover solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ from the matrix $\hat{\boldsymbol{W}}$. Below we are going to introduce two basic decomposition routines: one is based on randomization and the other on eigen-decomposition. They play a fundamental role in our proposed algorithms; all solution methods to be developed later rely on these two routines as a basis.

Decomposition Routine 3.2.1

- *INPUT: matrices* $\boldsymbol{M} \in \mathbb{R}^{n_1 \times n_2}$, $\boldsymbol{W} \in \mathbb{S}^{n_1 \times n_2}$.

1 *Construct*

$$\tilde{\boldsymbol{W}} = \begin{bmatrix} \boldsymbol{I}_{n_1 \times n_1} & \boldsymbol{W} \\ \boldsymbol{W}^{\mathrm{T}} & \boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} \end{bmatrix} \succeq 0.$$

2 *Randomly generate*

$$\begin{pmatrix} \boldsymbol{\xi} \\ \boldsymbol{\eta} \end{pmatrix} \sim \mathcal{N}(\boldsymbol{0}_{n_1+n_2}, \tilde{\boldsymbol{W}})$$

and repeat if necessary, until $\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{\eta} \geq \boldsymbol{M} \bullet \boldsymbol{W}$ *and* $\|\boldsymbol{\xi}\|\|\boldsymbol{\eta}\| \leq O(\sqrt{n_1})$.

3 *Compute* $\boldsymbol{x} = \boldsymbol{\xi}/\|\boldsymbol{\xi}\|$ *and* $\boldsymbol{y} = \boldsymbol{\eta}/\|\boldsymbol{\eta}\|$.

- *OUTPUT: vectors* $\boldsymbol{x} \in \mathbb{S}^{n_1}$, $\boldsymbol{y} \in \mathbb{S}^{n_2}$.

Now, let $\boldsymbol{M} = F(\cdot, \cdot, \hat{\boldsymbol{z}})$ and $\boldsymbol{W} = \hat{\boldsymbol{W}}$ in applying the above decomposition routine. For the randomly generated $(\boldsymbol{\xi}, \boldsymbol{\eta})$, we have

$$\mathrm{E}[F(\boldsymbol{\xi}, \boldsymbol{\eta}, \hat{\boldsymbol{z}})] = \mathrm{E}[\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{\eta}] = \boldsymbol{M} \bullet \boldsymbol{W} = F(\hat{\boldsymbol{W}}, \hat{\boldsymbol{z}}).$$

He et al. [50] establish that if $f(\boldsymbol{x})$ is a homogeneous quadratic form and $\boldsymbol{x}$ is drawn from a zero-mean multivariate normal distribution, then there is a universal constant $\theta \geq 0.03$ such that

$$\mathrm{Prob}\{f(\boldsymbol{x}) \geq \mathrm{E}[f(\boldsymbol{x})]\} \geq \theta.$$

Since $\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{\eta}$ is a homogeneous quadratic form of the normal random vector $\binom{\boldsymbol{\xi}}{\boldsymbol{\eta}}$, we know

$$\mathrm{Prob}\{\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{\eta} \geq \boldsymbol{M} \bullet \boldsymbol{W}\} = \mathrm{Prob}\{F(\boldsymbol{\xi}, \boldsymbol{\eta}, \hat{\boldsymbol{z}}) \geq \mathrm{E}[F(\boldsymbol{\xi}, \boldsymbol{\eta}, \hat{\boldsymbol{z}})]\} \geq \theta.$$

Moreover, by using a property of normal random vectors (see Lemma 3.1 of [77]), we have

$$\mathrm{E}\left[\|\boldsymbol{\xi}\|^2\|\boldsymbol{\eta}\|^2\right] = \mathrm{E}\left[\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\xi_i{}^2\eta_j{}^2\right] = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\left(\mathrm{E}[\xi_i{}^2]\mathrm{E}[\eta_j{}^2] + 2(\mathrm{E}[\xi_i\eta_j])^2\right)$$

$$= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\left[(\hat{W}^{\mathrm{T}}\hat{W})_{jj} + 2\hat{W}_{ij}{}^2\right] = (n_1+2)\,\mathrm{tr}\,(\hat{\boldsymbol{W}}^{\mathrm{T}}\hat{\boldsymbol{W}}) = n_1 + 2.$$

By applying the Markov inequality, for any $t > 0$

$$\mathrm{Prob}\left\{\|\boldsymbol{\xi}\|^2\|\boldsymbol{\eta}\|^2 \geq t\right\} \leq \mathrm{E}\left[\|\boldsymbol{\xi}\|^2\|\boldsymbol{\eta}\|^2\right]/t = (n_1+2)/t.$$

Therefore, by the so-called union inequality for the probability of joint events, we have

$$\begin{aligned}
&\mathrm{Prob}\left\{F(\boldsymbol{\xi},\boldsymbol{\eta},\hat{\boldsymbol{z}}) \geq F(\hat{\boldsymbol{W}},\hat{\boldsymbol{z}}),\ \|\boldsymbol{\xi}\|^2\|\boldsymbol{\eta}\|^2 \leq t\right\} \\
\geq\ & 1 - \mathrm{Prob}\left\{F(\boldsymbol{\xi},\boldsymbol{\eta},\hat{\boldsymbol{z}}) < F(\hat{\boldsymbol{W}},\hat{\boldsymbol{z}})\right\} - \mathrm{Prob}\left\{\|\boldsymbol{\xi}\|^2\|\boldsymbol{\eta}\|^2 > t\right\} \\
\geq\ & 1 - (1-\theta) - (n_1+2)/t = \theta/2,
\end{aligned}$$

where we let $t = 2(n_1+2)/\theta$. Thus we have

$$F(\boldsymbol{x},\boldsymbol{y},\hat{\boldsymbol{z}}) \geq \frac{F(\hat{\boldsymbol{W}},\hat{\boldsymbol{z}})}{\sqrt{t}} \geq v(\hat{T}_S)\sqrt{\frac{\theta}{2(n_1+2)}},$$

obtaining an $\Omega(1/\sqrt{n_1})$-approximation ratio.

Below we present an alternative (and deterministic) decomposition routine.

Decomposition Routine 3.2.2

- *INPUT: a matrix* $\boldsymbol{M} \in \mathbb{R}^{n_1\times n_2}$.

1 *Find an eigenvector* $\hat{\boldsymbol{y}}$ *corresponding to the largest eigenvalue of* $\boldsymbol{M}^{\mathrm{T}}\boldsymbol{M}$.

2 *Compute* $\boldsymbol{x} = \boldsymbol{M}\hat{\boldsymbol{y}}/\|\boldsymbol{M}\hat{\boldsymbol{y}}\|$ *and* $\boldsymbol{y} = \hat{\boldsymbol{y}}/\|\hat{\boldsymbol{y}}\|$.

- *OUTPUT: vectors* $\boldsymbol{x} \in \mathbb{S}^{n_1}$, $\boldsymbol{y} \in \mathbb{S}^{n_2}$.

This decomposition routine literally follows the proof of Proposition 3.2.1, which tells us that $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{y} \geq \|\boldsymbol{M}\|/\sqrt{n_1}$. Thus we have

$$F(\boldsymbol{x},\boldsymbol{y},\hat{\boldsymbol{z}}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{y} \geq \frac{\|\boldsymbol{M}\|}{\sqrt{n_1}} = \max_{\boldsymbol{Z}\in\mathbb{S}^{n_1\times n_2}} \frac{\boldsymbol{M}\bullet\boldsymbol{Z}}{\sqrt{n_1}} \geq \frac{\boldsymbol{M}\bullet\hat{\boldsymbol{W}}}{\sqrt{n_1}} = \frac{F(\hat{\boldsymbol{W}},\hat{\boldsymbol{z}})}{\sqrt{n_1}} \geq \frac{v(\hat{T}_S)}{\sqrt{n_1}}.$$

The complexity for DR 3.2.1 is $O(n_1 n_2 \log(1/\epsilon))$ with probability $1-\epsilon$, and for DR 3.2.2 it is $O\left(\max\{{n_1}^3, n_1 n_2\}\right)$. However DR 3.2.2 is indeed very easy to implement, and is deterministic. Both DR 3.2.1 and DR 3.2.2 lead to the following approximation result in terms of the order of the approximation ratio.

Theorem 3.2.3 *If $d = 3$, then (T_S) admits a polynomial-time approximation algorithm with approximation ratio $1/\sqrt{n_1}$.*

Now we proceed to the case for general d. Let $\boldsymbol{X} = \boldsymbol{x}^1(\boldsymbol{x}^d)^{\mathrm{T}}$, and (T_S) can be relaxed to

$$\begin{array}{lll} (\tilde{T}_S) & \max & F(\boldsymbol{X}, \boldsymbol{x}^2, \boldsymbol{x}^3, \cdots, \boldsymbol{x}^{d-1}) \\ & \text{s.t.} & \boldsymbol{X} \in \mathbb{S}^{n_1 \times n_d},\ \boldsymbol{x}^k \in \mathbb{S}^{n_k},\ k = 2, 3, \ldots, d-1. \end{array}$$

Clearly it is a type of the model (T_S) with degree $d-1$. Suppose $(\tilde{T}_S)$ can be solved approximately in polynomial-time with approximation ratio τ, i.e., we find $(\hat{\boldsymbol{X}}, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1})$ with

$$F(\hat{\boldsymbol{X}}, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1}) \geq \tau v(\tilde{T}_S) \geq \tau v(T_S).$$

Observing that $F(\cdot, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1}, \cdot)$ is an $n_1 \times n_d$ matrix, using DR 3.2.2 we shall find $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^d)$ such that

$$F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) \geq F(\hat{\boldsymbol{X}}, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1})/\sqrt{n_1} \geq {n_1}^{-\frac{1}{2}} \tau v(T_S).$$

By induction this leads to the following:

Theorem 3.2.4 *(T_S) admits a polynomial-time approximation algorithm with approximation ratio $\tau(T_S)$, where*

$$\tau(T_S) := \left(\prod_{k=1}^{d-2} n_k\right)^{-\frac{1}{2}}.$$

Below we summarize the above recursive procedure to solve (T_S) as in Theorem 3.2.4. Remark that the approximation performance ratio of this algorithm is tight. In a special example $F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) = \sum_{i=1}^{n} x_i^1 x_i^2 \cdots x_i^d$, the algorithm can be made to return a solution with approximation ratio being exactly $\tau(T_S)$.

Algorithm 3.2.3

- *INPUT: a d-th order tensor $\boldsymbol{F} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ with $n_1 \leq n_2 \leq \cdots \leq n_d$.*

1 *Rewrite* $\boldsymbol{F}$ *as a* $(d-1)$*-th order tensor* $\boldsymbol{F}' \in \mathbb{R}^{n_2 \times n_3 \times \cdots \times n_{d-1} \times n_d n_1}$ *by combing its first and last modes into one, and placing it in the last mode of* $\boldsymbol{F}'$*, i.e.,*

$$F_{i_1,i_2,\cdots,i_d} = F'_{i_2,i_3,\cdots,i_{d-1},(i_1-1)n_d+i_d} \quad \forall\, 1 \le i_1 \le n_1, 1 \le i_2 \le n_2, \cdots, 1 \le i_d \le n_d.$$

2 *For* (T_S) *with the* $(d-1)$*-th order tensor* $\boldsymbol{F}'$*: if* $d-1=2$*, then apply DR 3.2.2, with input* $\boldsymbol{F}' = \boldsymbol{M}$ *and output* $(\hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^{1,d}) = (\boldsymbol{x}, \boldsymbol{y})$*; otherwise obtain a solution* $(\hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1}, \hat{\boldsymbol{x}}^{1,d})$ *by recursion.*

3 *Compute a matrix* $\boldsymbol{M}' = F(\cdot, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1}, \cdot)$ *and rewrite the vector* $\hat{\boldsymbol{x}}^{1,d}$ *as a matrix* $\boldsymbol{X} \in \mathbb{S}^{n_1 \times n_d}$*.*

4 *Apply either DR 3.2.1 or DR 3.2.2, with input* $(\boldsymbol{M}', \boldsymbol{X}) = (\boldsymbol{M}, \boldsymbol{W})$ *and output* $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^d) = (\boldsymbol{x}, \boldsymbol{y})$*.*

- *OUTPUT: a feasible solution* $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$*.*

3.3 Multilinear Form with Ellipsoidal Constraints

In this section, we consider a generalization of the optimization model discussed in Section 3.2, to include general ellipsoidal constraints. Specifically, the model is

$$\boxed{\begin{array}{lll} (T_Q) & \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\ & \text{s.t.} & (\boldsymbol{x}^k)^{\mathrm{T}} \boldsymbol{Q}^k_{i_k} \boldsymbol{x}^k \le 1,\ k = 1, 2, \ldots, d,\ i_k = 1, 2, \ldots, m_k, \\ & & \boldsymbol{x}^k \in \mathbb{R}^{n_k},\ k = 1, 2, \ldots, d, \end{array}}$$

where $\boldsymbol{Q}^k_{i_k} \succeq 0$ and $\sum_{i_k=1}^{m_k} \boldsymbol{Q}^k_{i_k} \succ 0$ for $k = 1, 2, \ldots, d$, $i_k = 1, 2, \ldots, m_k$.

Let us start with the case $d = 2$, and suppose $F(\boldsymbol{x}^1, \boldsymbol{x}^2) = (\boldsymbol{x}^1)^{\mathrm{T}} \boldsymbol{F} \boldsymbol{x}^2$. Denote $\boldsymbol{y} = \begin{pmatrix} \boldsymbol{x}^1 \\ \boldsymbol{x}^2 \end{pmatrix}$, $\boldsymbol{F}' = \begin{bmatrix} \boldsymbol{0}_{n_1 \times n_1} & \frac{\boldsymbol{F}}{2} \\ \frac{\boldsymbol{F}^{\mathrm{T}}}{2} & \boldsymbol{0}_{n_2 \times n_2} \end{bmatrix}$, $\boldsymbol{Q}_i = \begin{bmatrix} \boldsymbol{Q}^1_i & \boldsymbol{0}_{n_1 \times n_2} \\ \boldsymbol{0}_{n_2 \times n_1} & \boldsymbol{0}_{n_2 \times n_2} \end{bmatrix}$ for all $1 \le i \le m_1$, and $\boldsymbol{Q}_i = \begin{bmatrix} \boldsymbol{0}_{n_1 \times n_1} & \boldsymbol{0}_{n_1 \times n_2} \\ \boldsymbol{0}_{n_2 \times n_1} & \boldsymbol{Q}^2_{i-m_1} \end{bmatrix}$ for all $m_1 + 1 \le i \le m_1 + m_2$. Then (T_Q) is equivalent to

$$\begin{array}{lll} & \max & \boldsymbol{y}^{\mathrm{T}} \boldsymbol{F}' \boldsymbol{y} \\ & \text{s.t.} & \boldsymbol{y}^{\mathrm{T}} \boldsymbol{Q}_i \boldsymbol{y} \le 1,\ i = 1, 2, \ldots, m_1 + m_2, \\ & & \boldsymbol{y} \in \mathbb{R}^{n_1 + n_2}. \end{array}$$

This QCQP problem is discussed in Section 2.5, and is well known to be solved approximately by a polynomial-time randomized algorithm with approximation ratio $\Omega\left(\frac{1}{\log(m_1+m_2)}\right)$ (see e.g., Nemirovski et al. [87] and He et al. [50]).

We now proceed to the higher order cases. To illustrate the essential ideas, we shall focus on the case $d = 3$. The extension to any higher order can be done by induction. In case $d = 3$ we may explicitly write (T_Q) as:

$$\begin{aligned}(\hat{T}_Q) \quad \max \quad & F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \\ \text{s.t.} \quad & \boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x} \le 1,\ i = 1, 2, \ldots, m_1, \\ & \boldsymbol{y}^{\mathrm{T}}\boldsymbol{P}_j\boldsymbol{y} \le 1,\ j = 1, 2, \ldots, m_2, \\ & \boldsymbol{z}^{\mathrm{T}}\boldsymbol{R}_k\boldsymbol{z} \le 1,\ k = 1, 2, \ldots, m_3, \\ & \boldsymbol{x} \in \mathbb{R}^{n_1}, \boldsymbol{y} \in \mathbb{R}^{n_2}, \boldsymbol{z} \in \mathbb{R}^{n_3},\end{aligned}$$

where $\boldsymbol{Q}_i \succeq 0$ for all $1 \le i \le m_1$, $\boldsymbol{P}_j \succeq 0$ for all $1 \le j \le m_2$, $\boldsymbol{R}_k \succeq 0$ for all $1 \le k \le m_3$, and $\sum_{i=1}^{m_1} \boldsymbol{Q}_i \succ 0$, $\sum_{j=1}^{m_2} \boldsymbol{P}_j \succ 0$, $\sum_{k=1}^{m_3} \boldsymbol{R}_k \succ 0$.

Combining the constraints of $\boldsymbol{x}$ and $\boldsymbol{y}$, we have

$$\mathrm{tr}\,(\boldsymbol{Q}_i\boldsymbol{x}\boldsymbol{y}^{\mathrm{T}}\boldsymbol{P}_j\boldsymbol{y}\boldsymbol{x}^{\mathrm{T}}) = \mathrm{tr}\,(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x}\boldsymbol{y}^{\mathrm{T}}\boldsymbol{P}_j\boldsymbol{y}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x} \cdot \boldsymbol{y}^{\mathrm{T}}\boldsymbol{P}_j\boldsymbol{y} \le 1.$$

Denoting $\boldsymbol{W} = \boldsymbol{x}\boldsymbol{y}^{\mathrm{T}}$, $(\hat{T}_Q)$ can be relaxed to

$$\begin{aligned}(\check{T}_Q) \quad \max \quad & F(\boldsymbol{W}, \boldsymbol{z}) \\ \text{s.t.} \quad & \mathrm{tr}\,(\boldsymbol{Q}_i\boldsymbol{W}\boldsymbol{P}_j\boldsymbol{W}^{\mathrm{T}}) \le 1,\ i = 1, 2, \ldots, m_1,\ j = 1, 2, \ldots, m_2, \\ & \boldsymbol{z}^{\mathrm{T}}\boldsymbol{R}_k\boldsymbol{z} \le 1,\ k = 1, 2, \ldots, m_3, \\ & \boldsymbol{W} \in \mathbb{R}^{n_1 \times n_2}, \boldsymbol{z} \in \mathbb{R}^{n_3}.\end{aligned}$$

Observe that for any $\boldsymbol{W} \in \mathbb{R}^{n_1 \times n_2}$,

$$\mathrm{tr}\,(\boldsymbol{Q}_i\boldsymbol{W}\boldsymbol{P}_j\boldsymbol{W}^{\mathrm{T}}) = \mathrm{tr}\,(\boldsymbol{Q}_i^{\frac{1}{2}}\boldsymbol{W}\boldsymbol{P}_j^{\frac{1}{2}}\boldsymbol{P}_j^{\frac{1}{2}}\boldsymbol{W}^{\mathrm{T}}\boldsymbol{Q}_i^{\frac{1}{2}}) = \left\|\boldsymbol{Q}_i^{\frac{1}{2}}\boldsymbol{W}\boldsymbol{P}_j^{\frac{1}{2}}\right\|^2 \ge 0,$$

and that for any $\boldsymbol{W} \neq \boldsymbol{0}$,

$$\begin{aligned}\sum_{1 \le i \le m_1, 1 \le j \le m_2} \mathrm{tr}\,(\boldsymbol{Q}_i\boldsymbol{W}\boldsymbol{P}_j\boldsymbol{W}^{\mathrm{T}}) &= \mathrm{tr}\left(\left(\sum_{i=1}^{m_1}\boldsymbol{Q}_i\right)\boldsymbol{W}\left(\sum_{j=1}^{m_2}\boldsymbol{P}_j\right)\boldsymbol{W}^{\mathrm{T}}\right) \\ &= \left\|\left(\sum_{i=1}^{m_1}\boldsymbol{Q}_i\right)^{\frac{1}{2}}\boldsymbol{W}\left(\sum_{j=1}^{m_2}\boldsymbol{P}_j\right)^{\frac{1}{2}}\right\|^2 > 0.\end{aligned}$$

Indeed, it is easy to verify that $\mathrm{tr}\,(\boldsymbol{Q}_i\boldsymbol{W}\boldsymbol{P}_j\boldsymbol{W}^{\mathrm{T}}) = (\mathrm{vec}(\boldsymbol{W}))^{\mathrm{T}}(\boldsymbol{Q}_i \otimes \boldsymbol{P}_j)\mathrm{vec}(\boldsymbol{W})$, which implies that $\mathrm{tr}\,(\boldsymbol{Q}_i\boldsymbol{W}\boldsymbol{P}_j\boldsymbol{W}^{\mathrm{T}}) \le 1$ is actually a convex quadratic constraint for $\boldsymbol{W}$.

Thus, $(\check{T}_Q)$ is exactly in the form of (T_Q) with $d = 2$. Therefore we are able to find a feasible solution $(\hat{\boldsymbol{W}}, \hat{\boldsymbol{z}})$ of $(\check{T}_Q)$ in polynomial-time, such that

$$F(\hat{\boldsymbol{W}}, \hat{\boldsymbol{z}}) \geq \Omega\left(\frac{1}{\log(m_1 m_2 + m_3)}\right) v(\check{T}_Q) \geq \Omega\left(\frac{1}{\log m}\right) v(\check{T}_Q), \tag{3.1}$$

where $m = \max\{m_1, m_2, m_3\}$. Let us fix $\hat{\boldsymbol{z}}$, and then $F(\cdot, \cdot, \hat{\boldsymbol{z}})$ is a matrix. Our next step is to generate $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ from $\hat{\boldsymbol{W}}$. For this purpose, we first introduce the following lemma.

Lemma 3.3.1 *Suppose $\boldsymbol{Q}_i \in \mathbb{R}^{n\times n}$, $\boldsymbol{Q}_i \succeq 0$ for all $1 \leq i \leq m$, and $\sum_{i=1}^m \boldsymbol{Q}_i \succ 0$, the following SDP problem*

$$\begin{array}{lll} (PS) & \min & \sum_{i=1}^m t_i \\ & \text{s.t.} & \operatorname{tr}(\boldsymbol{U}\boldsymbol{Q}_i) \leq 1,\ i = 1, 2, \ldots, m, \\ & & t_i \geq 0,\ i = 1, 2, \ldots, m, \\ & & \begin{bmatrix} \boldsymbol{U} & \boldsymbol{I}_{n\times n} \\ \boldsymbol{I}_{n\times n} & \sum_{i=1}^m t_i \boldsymbol{Q}_i \end{bmatrix} \succeq 0 \end{array}$$

has an optimal solution with optimal value equal to n.

Proof. Straightforward computation shows that the dual of (PS) is

$$\begin{array}{lll} (DS) & \max & -\sum_{i=1}^m s_i - 2\operatorname{tr}(\boldsymbol{Z}) \\ & \text{s.t.} & \operatorname{tr}(\boldsymbol{X}\boldsymbol{Q}_i) \leq 1,\ i = 1, 2, \ldots, m, \\ & & s_i \geq 0,\ i = 1, 2, \ldots, m, \\ & & \begin{bmatrix} \boldsymbol{X} & \boldsymbol{Z} \\ \boldsymbol{Z}^{\mathrm{T}} & \sum_{i=1}^m s_i \boldsymbol{Q}_i \end{bmatrix} \succeq 0. \end{array}$$

Observe that (DS) indeed resembles (PS). Since $\sum_{i=1}^m \boldsymbol{Q}_i \succ 0$, both (PS) and (DS) satisfy the Slater condition, and thus both of them have attainable optimal solutions satisfying the strong duality relationship, i.e., $v(PS) = v(DS)$. Let $(\boldsymbol{U}^*, t^*)$ be an optimal solution of (PS). Clearly $\boldsymbol{U}^* \succ 0$, and by the Schur complement relationship we have $\sum_{i=1}^m t_i^* \boldsymbol{Q}_i \succeq (\boldsymbol{U}^*)^{-1}$. Therefore,

$$v(PS) = \sum_{i=1}^m t_i^* \geq \sum_{i=1}^m t_i^* \operatorname{tr}(\boldsymbol{U}^* \boldsymbol{Q}_i) \geq \operatorname{tr}(\boldsymbol{U}^* (\boldsymbol{U}^*)^{-1}) = n. \tag{3.2}$$

Observe that for any dual feasible solution $(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{s})$ we always have $-\sum_{i=1}^m s_i \leq -\operatorname{tr}(\boldsymbol{X}\sum_{i=1}^m s_i \boldsymbol{Q}_i)$. Hence the following problem is a relaxation of (DS)

$$\begin{array}{lll} (RS) & \max & -\operatorname{tr}(\boldsymbol{X}\boldsymbol{Y}) - 2\operatorname{tr}(\boldsymbol{Z}) \\ & \text{s.t.} & \begin{bmatrix} \boldsymbol{X} & \boldsymbol{Z} \\ \boldsymbol{Z}^{\mathrm{T}} & \boldsymbol{Y} \end{bmatrix} \succeq 0. \end{array}$$

Consider any feasible solution $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$ of (RS). Let $\boldsymbol{X} = \boldsymbol{P}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{P}$ be an orthonormal decomposition with $\boldsymbol{D} = \mathrm{Diag}\,(d_1, d_2, \cdots, d_n)$ and $\boldsymbol{P}^{-1} = \boldsymbol{P}^{\mathrm{T}}$. Notice that $(\boldsymbol{D}, \boldsymbol{Y}', \boldsymbol{Z}') := (\boldsymbol{P}\boldsymbol{X}\boldsymbol{P}^{\mathrm{T}}, \boldsymbol{P}\boldsymbol{Y}\boldsymbol{P}^{\mathrm{T}}, \boldsymbol{P}\boldsymbol{Z}\boldsymbol{P}^{\mathrm{T}})$ is also a feasible solution for (RS) with the same objective value. By the feasibility, it follows that $d_i Y'_{ii} - (Z'_{ii})^2 \geq 0$ for $i = 1, 2, \ldots, n$. Therefore,

$$-\mathrm{tr}\,(\boldsymbol{X}\boldsymbol{Y}) - 2\,\mathrm{tr}\,(\boldsymbol{Z}) = -\mathrm{tr}\,(\boldsymbol{D}\boldsymbol{Y}') - 2\,\mathrm{tr}\,(\boldsymbol{Z}') = -\sum_{i=1}^{n} d_i Y'_{ii} - 2\sum_{i=1}^{n} Z'_{ii}$$

$$\leq -\sum_{i=1}^{n}(Z'_{ii})^2 - 2\sum_{i=1}^{n} Z'_{ii} \leq -\sum_{i=1}^{n}(Z'_{ii}+1)^2 + n \leq n.$$

This implies that $v(DS) \leq v(RS) \leq n$. By combining this with (3.2), and noticing the strong duality relationship, it follows that $v(PS) = v(DS) = n$. □

We then have the following decomposition method, to be called DR 3.3.1, as a further extension of DR 3.2.1. It plays a similar role in Algorithm 3.3.2 as DR 3.2.1 or DR 3.2.2 does in Algorithm 3.2.3.

Decomposition Routine 3.3.1

- *INPUT: matrices* $\boldsymbol{Q}_i \in \mathbb{R}^{n_1 \times n_1}$, $\boldsymbol{Q}_i \succeq 0$ *for all* $1 \leq i \leq m_1$ *with* $\sum_{i=1}^{m_1} \boldsymbol{Q}_i \succ 0$, $\boldsymbol{P}_j \in \mathbb{R}^{n_2 \times n_2}$, $\boldsymbol{P}_j \succeq 0$ *for all* $1 \leq j \leq m_2$ *with* $\sum_{j=1}^{m_2} \boldsymbol{P}_j \succ 0$, $\boldsymbol{W} \in \mathbb{R}^{n_1 \times n_2}$ *with* $\mathrm{tr}\,(\boldsymbol{Q}_i \boldsymbol{W} \boldsymbol{P}_j \boldsymbol{W}^{\mathrm{T}}) \leq 1$ *for all* $1 \leq i \leq m_1$ *and* $1 \leq j \leq m_2$, *and* $\boldsymbol{M} \in \mathbb{R}^{n_1 \times n_2}$.

1 *Solve the SDP problem*

$$\begin{array}{ll} \min & \sum_{i=1}^{m_1} t_i \\ \text{s.t.} & \mathrm{tr}\,(\boldsymbol{U}\boldsymbol{Q}_i) \leq 1,\ i = 1, 2, \ldots, m_1, \\ & t_i \geq 0,\ i = 1, 2, \ldots, m_1, \\ & \begin{bmatrix} \boldsymbol{U} & \boldsymbol{I}_{n\times n} \\ \boldsymbol{I}_{n\times n} & \sum_{i=1}^{m_1} t_i \boldsymbol{Q}_i \end{bmatrix} \succeq 0 \end{array}$$

to get an optimal solution of a matrix $\boldsymbol{U}$ *and scalars* $t_1, t_2, \cdots, t_{m_1}$.

2 *Construct*

$$\tilde{\boldsymbol{W}} = \begin{bmatrix} \boldsymbol{U} & \boldsymbol{W} \\ \boldsymbol{W}^{\mathrm{T}} & \boldsymbol{W}^{\mathrm{T}}(\sum_{i=1}^{m_1} t_i \boldsymbol{Q}_i)\boldsymbol{W} \end{bmatrix} \succeq 0.$$

3 *Randomly generate*

$$\begin{pmatrix} \boldsymbol{\xi} \\ \boldsymbol{\eta} \end{pmatrix} \sim \mathcal{N}(\mathbf{0}_{n_1+n_2}, \tilde{\boldsymbol{W}})$$

and repeat if necessary, until $\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{\eta} \geq \boldsymbol{M} \bullet \boldsymbol{W}$, $\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{\xi} \leq O(\log m_1)$ *for all* $1 \leq i \leq m_1$, *and* $\boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{P}_j\boldsymbol{\eta} \leq O(n_1 \log m_2)$ *for all* $1 \leq j \leq m_2$.

4 *Compute* $\boldsymbol{x} = \boldsymbol{\xi} \Big/ \sqrt{\max_{1\leq i\leq m_1}\{\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{\xi}\}}$ *and* $\boldsymbol{y} = \boldsymbol{\eta} \Big/ \sqrt{\max_{1\leq j\leq m_2}\{\boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{P}_j\boldsymbol{\eta}\}}$.

- *OUTPUT: vectors* $\boldsymbol{x} \in \mathbb{R}^{n_1}$, $\boldsymbol{y} \in \mathbb{R}^{n_2}$.

The computational complexity of DR 3.3.1 depends on the algorithm for solving the SDP problem (PS), which has $O({n_1}^2)$ number of variables and $O(m_1)$ number of constraints. In addition it requires $O(n_2(n_1m_1 + n_2m_2)\log(1/\epsilon))$ other operations to get the quality assured solution with probability $1 - \epsilon$.

Lemma 3.3.2 *Under the input of DR 3.3.1, we can find* $\boldsymbol{x} \in \mathbb{R}^{n_1}$ *and* $\boldsymbol{y} \in \mathbb{R}^{n_2}$ *by a polynomial-time randomized algorithm, satisfying* $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x} \leq 1$ *for all* $1 \leq i \leq m_1$ *and* $\boldsymbol{y}^{\mathrm{T}}\boldsymbol{P}_j\boldsymbol{y} \leq 1$ *for all* $1 \leq j \leq m_2$, *such that*

$$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{y} \geq \frac{1}{\sqrt{n}}\Omega\left(\frac{1}{\sqrt{\log m_1 \log m_2}}\right)\boldsymbol{M} \bullet \boldsymbol{W}.$$

Proof. Following the randomization procedure in Step 3 of DR 3.3.1, by Lemma 3.3.1 we have for any $1 \leq i \leq m_1$ and $1 \leq j \leq m_2$,

$$\mathrm{E}[\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{\xi}] = \mathrm{tr}\,(\boldsymbol{Q}_i\boldsymbol{U}) \leq 1,$$
$$\mathrm{E}[\boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{P}_j\boldsymbol{\eta}] = \mathrm{tr}\left(\boldsymbol{P}_j\boldsymbol{W}^{\mathrm{T}}\left(\sum_{i=1}^{m_1} t_i\boldsymbol{Q}_i\right)\boldsymbol{W}\right) = \sum_{i=1}^{m_1} t_i\,\mathrm{tr}\,(\boldsymbol{P}_j\boldsymbol{W}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{W}) \leq \sum_{i=1}^{m_1} t_i = n_1.$$

So et al. [109] have established that if $\boldsymbol{\xi}$ is a normal random vector and $\boldsymbol{Q} \succeq 0$, then for any $\alpha > 0$,

$$\mathrm{Prob}\,\{\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{\xi} \geq \alpha\mathrm{E}[\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{\xi}]\} \leq 2e^{-\frac{\alpha}{2}}.$$

Applying this result we have

$$\mathrm{Prob}\,\{\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{\xi} \geq \alpha_1\} \leq \mathrm{Prob}\,\{\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{\xi} \geq \alpha_1\mathrm{E}[\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{\xi}]\} \leq 2e^{-\frac{\alpha_1}{2}},$$
$$\mathrm{Prob}\,\{\boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{P}_j\boldsymbol{\eta} \geq \alpha_2 n_1\} \leq \mathrm{Prob}\,\{\boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{P}_j\boldsymbol{\eta} \geq \alpha_2\mathrm{E}[\boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{P}_j\boldsymbol{\eta}]\} \leq 2e^{-\frac{\alpha_2}{2}}.$$

Moreover, $\mathrm{E}[\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{\eta}] = \boldsymbol{M} \bullet \boldsymbol{W}$. Now let $\hat{\boldsymbol{x}} = \boldsymbol{\xi}/\sqrt{\alpha_1}$ and $\hat{\boldsymbol{y}} = \boldsymbol{\eta}/\sqrt{\alpha_2 n_1}$, and we have

$$\begin{aligned}
&\mathrm{Prob}\left\{\hat{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{M}\hat{\boldsymbol{y}} \geq \frac{\boldsymbol{M} \bullet \boldsymbol{W}}{\sqrt{\alpha_1\alpha_2 n_1}},\ \hat{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{Q}_i\hat{\boldsymbol{x}} \leq 1\,\forall\, 1 \leq i \leq m_1,\ \hat{\boldsymbol{y}}^{\mathrm{T}}\boldsymbol{P}_j\hat{\boldsymbol{y}} \leq 1\,\forall\, 1 \leq j \leq m_2\right\} \\
&\geq 1 - \mathrm{Prob}\left\{\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{\eta} < \boldsymbol{M} \bullet W\right\} - \sum_{i=1}^{m_1}\mathrm{Prob}\left\{\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{\xi} > \alpha_1\right\} - \sum_{j=1}^{m_2}\mathrm{Prob}\left\{\boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{P}_j\boldsymbol{\eta} > \alpha_2 n_1\right\} \\
&\geq 1 - (1-\theta) - m_1 \cdot 2e^{-\frac{\alpha_1}{2}} - m_2 \cdot 2e^{-\frac{\alpha_2}{2}} = \theta/2,
\end{aligned}$$

where $\alpha_1 := 2\ln(8m_1/\theta)$ and $\alpha_2 := 2\ln(8m_2/\theta)$. Since $\alpha_1\alpha_2 = O(\log m_1 \log m_2)$, the desired $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ can be found with high probability in multiple trials. □

Let us now turn back to $(\hat{T}_Q)$. If we let $\boldsymbol{W} = \hat{\boldsymbol{W}}$ and $\boldsymbol{M} = F(\cdot,\cdot,\hat{\boldsymbol{z}})$ in applying Lemma 3.3.2, then in polynomial-time we can find $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$, satisfying the constraints of $(\hat{T}_Q)$, such that

$$F(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}) = \hat{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{M}\hat{\boldsymbol{y}} \geq \frac{1}{\sqrt{n_1}}\,\Omega\left(\frac{1}{\sqrt{\log m_1 \log m_2}}\right)\boldsymbol{M} \bullet \hat{\boldsymbol{W}} \geq \frac{1}{\sqrt{n_1}}\,\Omega\left(\frac{1}{\log m}\right)F(\hat{\boldsymbol{W}}, \hat{\boldsymbol{z}}).$$

Combined with (3.1), we thus prove the following result.

Theorem 3.3.3 *If $d = 3$, then (T_Q) admits a polynomial-time randomized approximation algorithm with approximation ratio $\frac{1}{\sqrt{n_1}}\,\Omega\left(\frac{1}{\log^2 m}\right)$, where $m = \max\{m_1, m_2, m_3\}$.*

This result can be generalized to the model (T_Q) of any fixed degree d.

Theorem 3.3.4 *(T_Q) admits a polynomial-time randomized approximation algorithm with approximation ratio $\tau(T_Q)$, where*

$$\tau(T_Q) := \left(\prod_{k=1}^{d-2} n_k\right)^{-\frac{1}{2}}\,\Omega\left(\log^{-(d-1)} m\right),$$

and $m = \max_{1\leq k\leq d}\{m_k\}$.

Proof. We again take recursive steps. Denoting $\boldsymbol{W} = \boldsymbol{x}^1(\boldsymbol{x}^d)^{\mathrm{T}}$ and (T_Q) is relaxed to

$$\begin{aligned}
(\tilde{T}_Q)\quad \max\quad & F(\boldsymbol{W}, \boldsymbol{x}^2, \boldsymbol{x}^3, \cdots, \boldsymbol{x}^{d-1}) \\
\text{s.t.}\quad & \mathrm{tr}\left(\boldsymbol{Q}^1_{i_1}\boldsymbol{W}\boldsymbol{Q}^d_{i_d}\boldsymbol{W}^{\mathrm{T}}\right) \leq 1,\ i_1 = 1, 2, \ldots, m_1,\ i_d = 1, 2, \ldots, m_d, \\
& (\boldsymbol{x}^k)^{\mathrm{T}}\boldsymbol{Q}^k_{i_k}\boldsymbol{x}^k \leq 1,\ k = 2, 3, \ldots, d-1,\ i_k = 1, 2, \ldots, m_k, \\
& \boldsymbol{W} \in \mathbb{R}^{n_1 \times n_d},\ \boldsymbol{x}^k \in \mathbb{R}^{n_k},\ k = 2, 3, \ldots, d-1.
\end{aligned}$$

Notice that $(\tilde{T}_Q)$ is exactly in the form of (T_Q) of degree $d-1$, by treating $\boldsymbol{W}$ as a vector of dimension $n_1 n_d$. By recursion, with high probability we can find a feasible

solution $(\hat{\boldsymbol{W}}, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1})$ of $(\tilde{T}_Q)$ in polynomial-time, such that

$$\begin{aligned} F(\hat{\boldsymbol{W}}, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1}) &\geq \left(\prod_{k=2}^{d-2} n_k\right)^{-\frac{1}{2}} \Omega\left(\log^{-(d-2)} \max\{m, m_1 m_d\}\right) v(\tilde{T}_Q) \\ &\geq \left(\prod_{k=2}^{d-2} n_k\right)^{-\frac{1}{2}} \Omega\left(\log^{-(d-2)} m\right) v(T_Q). \end{aligned}$$

As long as we fix $(\hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1})$, and let $\boldsymbol{M} = F(\cdot, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1}, \cdot)$ and $\boldsymbol{W} = \hat{\boldsymbol{W}}$ in applying Lemma 3.3.2, we are be able to find $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^d)$ satisfying the constraints of (T_Q), such that

$$F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) \geq \frac{1}{\sqrt{n_1}} \Omega\left(\frac{1}{\log m}\right) F(\hat{\boldsymbol{W}}, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1}) \geq \tau(T_Q)\, v(T_Q).$$

□

Summarizing, the recursive procedure for solving general (T_Q) (Theorem 3.3.4) is highlighted as follows:

Algorithm 3.3.2

- *INPUT: a d-th order tensor* $\boldsymbol{F} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ *with* $n_1 \leq n_2 \leq \cdots \leq n_d$, *matrices* $\boldsymbol{Q}_{i_k}^k \in \mathbb{R}^{n_k \times n_k}$, $\boldsymbol{Q}_{i_k}^k \succeq 0$ *and* $\sum_{i_k=1}^{m_k} \boldsymbol{Q}_{i_k}^k \succ 0$ *for all* $1 \leq k \leq d$ *and* $1 \leq i_k \leq m_k$.

1 *Rewrite* $\boldsymbol{F}$ *as a* $(d-1)$*-th order tensor* $\boldsymbol{F}' \in \mathbb{R}^{n_2 \times n_3 \times \cdots \times n_{d-1} \times n_d n_1}$ *by combing its first and last modes into one, and placing it in the last mode of* $\boldsymbol{F}'$, *i.e.,*

$$F_{i_1, i_2, \cdots, i_d} = F'_{i_2, i_3, \cdots, i_{d-1}, (i_1-1)n_d + i_d} \quad \forall\, 1 \leq i_1 \leq n_1, 1 \leq i_2 \leq n_2, \cdots, 1 \leq i_d \leq n_d.$$

2 *Compute matrices* $\boldsymbol{P}_{i_1, i_d} = \boldsymbol{Q}_{i_1}^1 \otimes \boldsymbol{Q}_{i_d}^d$ *for all* $1 \leq i_1 \leq m_1$ *and* $1 \leq i_d \leq m_d$.

3 *For* (T_Q) *with the* $(d-1)$*-th order tensor* $\boldsymbol{F}'$, *matrices* $\boldsymbol{Q}_{i_k}^k$ $(2 \leq k \leq d-1, 1 \leq i_k \leq m_k)$ *and* $\boldsymbol{P}_{i_1, i_d}$ $(1 \leq i_1 \leq m_1, 1 \leq i_d \leq m_d)$*: if* $d - 1 = 2$, *then apply SDP relaxation and randomization procedure (Nemirovski et al. [87]) to obtain an approximate solution* $(\hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^{1,d})$*; otherwise obtain a solution* $(\hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1}, \hat{\boldsymbol{x}}^{1,d})$ *by recursion.*

4 *Compute a matrix* $\boldsymbol{M}' = F(\cdot, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1}, \cdot)$ *and rewrite the vector* $\hat{\boldsymbol{x}}^{1,d}$ *as a matrix* $\boldsymbol{X} \in \mathbb{R}^{n_1 \times n_d}$.

5 *Apply DR 3.3.1 with input* $(\boldsymbol{Q}_i^1, \boldsymbol{Q}_j^d, \boldsymbol{X}, \boldsymbol{M}') = (\boldsymbol{Q}_i, \boldsymbol{P}_j, \boldsymbol{W}, \boldsymbol{M})$ *for all* $1 \le i \le m_1$ *and* $1 \le j \le m_2$ *and output* $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^d) = (\boldsymbol{x}, \boldsymbol{y})$.

- *OUTPUT: a feasible solution* $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$.

3.4 Applications

As we mentioned in the beginning of this chapter, one of the main reasons to study multilinear form optimizations is its strong connection to homogenous polynomial optimizations in deriving approximation bounds, which will be discussed in the next chapter. Apart from that, these models also have versatile applications. Here we present two problems in this section and show that they are readily formulated by the polynomial optimization models in this chapter.

3.4.1 Singular Values of Trilinear Forms

Trilinear forms play an increasingly important role in many parts of analysis, e.g., in Fourier analysis, where they appear in the guise of paracommutators and compensated quantities (see a survey by Peng and Wong [95]). The problem of singular values of trilinear forms is the following (see also [18]). Denote $\mathbb{H}_1$, $\mathbb{H}_2$ and $\mathbb{H}_3$ to be three separable Hilbert spaces over the field $\mathbb{K}$, where $\mathbb{K}$ stands either for the real or the complex numbers, and denote a trilinear form $F : \mathbb{H}_1 \times \mathbb{H}_2 \times \mathbb{H}_3 \mapsto \mathbb{K}$. The *spectrum norm of the trilinear form* F is then the following maximization problem:

$$\begin{array}{rll} \|\boldsymbol{F}\|_S := & \sup & |F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})| \\ & \text{s.t.} & \|\boldsymbol{x}\| \le 1,\ \|\boldsymbol{y}\| \le 1,\ \|\boldsymbol{z}\| \le 1, \\ & & \boldsymbol{x} \in \mathbb{H}_1,\ \boldsymbol{y} \in \mathbb{H}_2,\ \boldsymbol{z} \in \mathbb{H}_3. \end{array}$$

More generally, one can state the problem of the *stationary* values of the functional $|F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})|$ under the same conditions. These corresponding stationary values are called *singular values of the trilinear form* F. Bernhardsson and Peetre [18] showed in the binary case, that $\|\boldsymbol{F}\|_S^2$ are among the roots of a certain algebraic equation, called the *millenial equation*, thought of as a generalization of the time honored secular equation in the case of matrices. Another approach to singular values is given by De Lathauwer et al. [28].

When specializing the Hilbert spaces to finite dimensional Euclidean spaces, i.e., $\mathbb{H}_i = \mathbb{R}^{n_i}$ for $i = 1, 2, 3$, and reserving the field $\mathbb{K}$ to be the real, the problem of computing the largest singular value $\|\boldsymbol{F}\|_S$ is equivalent to (T_S) when $d = 3$. This is because one can always use $(-\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ to replace $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ if its objective value is negative, hence the absolute value sign in $|F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})|$ can be omitted. Moreover, we can also scale the decision variables such that $\|\boldsymbol{x}\| = \|\boldsymbol{y}\| = \|\boldsymbol{z}\| = 1$ without decreasing the objective. According to Proposition 3.2.2, the problem of computing $\|\boldsymbol{F}\|_S$ is NP-hard already in this real case. Together with Theorem 3.2.3, the spectrum norm of a trilinear form can be approximated in polynomial-time with a factor of $\frac{1}{\sqrt{\min\{n_1, n_2, n_3\}}}$.

3.4.2 Rank-One Approximation of Tensors

Decompositions of higher order tensors (i.e., the order of the tensor is bigger than or equal to 3) have versatile applications in psychometrics, chemometrics, signal processing, computer vision, numerical analysis, data mining, neuroscience, graph analysis, and elsewhere (see e.g., an excellent survey by Kolda and Bader [68]). The earliest story of tensor decomposition dates back to 1927, where Hitchcock [55, 56] proposed the idea of the polyadic form of a tensor. Today, tensor decomposition is most widely used in the form of canonical decomposition (CANDECOMP) by Carroll and Chang [23] and parallel factors (PARAFAC) by Harshman [45], or in short CP decomposition.

A CP decomposition decomposes a tensor as a summation of rank-one tensors, i.e., tensors who can be written as outer product of vectors (see e.g., [67]). Specifically, for a d-th order tensor $\boldsymbol{F} = (F_{i_1 i_2 \cdots i_d}) \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ and a given positive integer r, its CP decomposition is as follows:

$$\boldsymbol{F} \approx \sum_{i=1}^{r} \boldsymbol{x}_i^1 \otimes \boldsymbol{x}_i^2 \otimes \cdots \otimes \boldsymbol{x}_i^d,$$

where $\boldsymbol{x}_i^k \in \mathbb{R}^{n_k}$ for $i = 1, 2, \ldots, r$, $k = 1, 2, \ldots, d$. Exact recovery of rank-one decompositions is in general impossible, due to various reasons, e.g., data errors. Thus the following optimization problem for the CP decomposition is straightforward, i.e., to minimize the norm of the difference,

$$\begin{array}{ll} \min & \|\boldsymbol{F} - \sum_{i=1}^{r} \boldsymbol{x}_i^1 \otimes \boldsymbol{x}_i^2 \otimes \cdots \otimes \boldsymbol{x}_i^d\| \\ \text{s.t.} & \boldsymbol{x}_i^k \in \mathbb{R}^{n_k},\ i = 1, 2, \ldots, r, k = 1, 2, \ldots, d. \end{array}$$

In particular, the case of $r = 1$ corresponds to the *best rank-one approximation* of a

tensor, i.e.,

$$(TA) \quad \begin{array}{ll} \min & \|\boldsymbol{F} - \boldsymbol{x}^1 \otimes \boldsymbol{x}^2 \otimes \cdots \otimes \boldsymbol{x}^d\| \\ \text{s.t.} & \boldsymbol{x}^k \in \mathbb{R}^{n_k},\ k = 1, 2, \ldots, d. \end{array}$$

By scaling, we may require the norm of $\boldsymbol{x}^k$ to be one, then (TA) is equivalent to

$$\begin{array}{ll} \min & \|\boldsymbol{F} - \lambda\, \boldsymbol{x}^1 \otimes \boldsymbol{x}^2 \otimes \cdots \otimes \boldsymbol{x}^d\| \\ \text{s.t.} & \lambda \in \mathbb{R},\ \boldsymbol{x}^k \in \mathbb{S}^{n_k},\ k = 1, 2, \ldots, d. \end{array}$$

For any fixed $\boldsymbol{x}^k \in \mathbb{S}^{n_k}$ $(k = 1, 2, \ldots, d)$, if we optimize the objective function of (TA) with respect to λ, we shall have

$$\begin{aligned} & \min_{\lambda \in \mathbb{R}} \|\boldsymbol{F} - \lambda\, \boldsymbol{x}^1 \otimes \boldsymbol{x}^2 \otimes \cdots \otimes \boldsymbol{x}^d\| \\ = & \min_{\lambda \in \mathbb{R}} \sqrt{\|\boldsymbol{F}\|^2 - 2\lambda\, \boldsymbol{F} \bullet (\boldsymbol{x}^1 \otimes \boldsymbol{x}^2 \otimes \cdots \otimes \boldsymbol{x}^d) + \lambda^2 \|\boldsymbol{x}^1 \otimes \boldsymbol{x}^2 \otimes \cdots \otimes \boldsymbol{x}^d\|^2} \\ = & \min_{\lambda \in \mathbb{R}} \sqrt{\|\boldsymbol{F}\|^2 - 2\lambda\, F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) + \lambda^2} \\ = & \sqrt{\|\boldsymbol{F}\|^2 - (F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d))^2}. \end{aligned}$$

Thus (TA) is equivalent to

$$\begin{array}{ll} \max & |F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d)| \\ \text{s.t.} & \boldsymbol{x}^k \in \mathbb{S}^{n_k},\ k = 1, 2, \ldots, d, \end{array}$$

which is the same as (T_S) discussed in Section 3.2. Remark that similar deductions can also be found in [29, 122, 67].

3.5 Numerical Experiments

In this section we are going to test the numerical performance of the approximation algorithms proposed in this chapter. As mentioned in Section 2.1.3, all the numerical computations reported in this thesis are performed on an Intel Pentium 4 CPU 2.80GHz computer with 2GB of RAM, and the supporting software is Matlab 7.7.0 (R2008b). The experiments in this section focus on the model (T_S) with $d = 4$, or equivalently, to recover the best rank-one approximation of a fourth order tensor discussed in Section 3.4.2. Specifically, we are going to test the following problem

$$(ET_S) \quad \begin{array}{ll} \max & F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{w}) = \sum_{1 \le i,j,k,\ell \le n} F_{ijk\ell}\, x_i y_j z_k w_\ell \\ \text{s.t.} & \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{w} \in \mathbb{S}^n. \end{array}$$

3.5.1 Randomly Simulated Data

A fourth order tensor $\boldsymbol{F}$ is generated randomly, whose n^4 entries are i.i.d. standard normals. Basically we have a choice to make in the recursion in Algorithm 3.2.3, yielding two different methods, both of which call the deterministic routine DR 3.2.2.

Method 1 follows the standard recursion procedures in Algorithm 3.2.3, and its first relaxation problem is

$$\begin{array}{rll} \bar{v}_1 = & \max & F(\boldsymbol{Z}, \boldsymbol{w}) = \sum_{1 \le i,j,k,\ell \le n} F_{ijk\ell} Z_{ijk} w_\ell \\ & \text{s.t.} & \boldsymbol{Z} \in \mathbb{S}^{n \times n \times n}, \boldsymbol{w} \in \mathbb{S}^n. \end{array}$$

After we get its optimal solution $(\boldsymbol{Z}^*, \boldsymbol{w}^*)$, we fix $\boldsymbol{w}^*$ and the problem is then reduced to a trilinear case of (ET_S), where recursion goes on. The objective value of the approximate solution obtained is denoted by v_1, and a ratio $\tau_1 := v_1/\bar{v}_1$ is also computed.

On the other hand, Method 2 chooses the other relaxation as its first step, which is

$$\begin{array}{rll} \bar{v}_2 = & \max & F(\boldsymbol{X}, \boldsymbol{Z}) = \sum_{1 \le i,j,k,\ell \le n} F_{ijk\ell} X_{ij} Z_{k\ell} \\ & \text{s.t.} & \boldsymbol{X}, \boldsymbol{Z} \in \mathbb{S}^{n \times n}. \end{array}$$

After we get its optimal solution $(\boldsymbol{X}^*, \boldsymbol{Z}^*)$, we may first fix $\boldsymbol{Z}^*$ and apply DR 3.2.2 to decompose $\boldsymbol{X}^*$ into $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}} \in \mathbb{S}^n$, and then fix $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ and apply DR 3.2.2 again to decompose $\boldsymbol{Z}^*$ into $\hat{\boldsymbol{z}}, \hat{\boldsymbol{w}} \in \mathbb{S}^n$, resulting a feasible solution. We also compute its objective value v_2 and a ratio $\tau_2 := v_2/\bar{v}_2$.

According to Theorem 3.2.4, Method 1 enjoys a theoretic worst-case performance ratio of $1/n$. Method 2 follows a similar fashion of Algorithm 3.2.3 by choosing a different recursion. It also enjoys a worst-case ratio of $1/n$, which can be proven similarly as that of Theorem 3.2.4. From the simulation results in Table 3.1, the objective values of their feasible solutions are indeed very close. However, Method 2 computes a much better upper bound of $v(ET_S)$, and thus ends up with a better approximation ratio.

The numerical results in Table 3.1 seem to indicate that the performance ratio of Method 1 is about $2/n$, while that of Method 2 is about $1/\sqrt{n}$. The main reason for the difference of upper bounds of $v(ET_S)$ ($\bar{v}_1$ vs. $\bar{v}_2$) is the first relaxation methods. By Proposition 3.2.1 we may guess that $\bar{v}_1 = \Omega(\|\boldsymbol{F}\|/\sqrt{n})$, while $\bar{v}_2 = \Omega(\|\boldsymbol{F}\|/n)$, and this may contribute to the large gap between $\bar{v}_1$ and $\bar{v}_2$. Consequently, it is quite possible that the true value of $v(ET_S)$ is closer to the solution values (v_1 and v_2), rather than the optimal value of the relaxed problem ($\bar{v}_2$). The real quality of the solutions produced is possibly much better than what is shown by the upper bounds.

Table 3.1: Numerical results (average of 10 instances) of (ET_S)

n	2	5	10	20	30	40	50	60	70
v_1	2.61	5.64	8.29	9.58	12.55	13.58	15.57	17.65	18.93
v_2	2.69	6.57	7.56	10.87	11.74	13.89	14.56	17.10	17.76
$\bar{v}_1$	3.84	12.70	34.81	93.38	169.08	258.94	360.89	472.15	594.13
$\bar{v}_2$	2.91	9.46	20.46	39.40	59.55	79.53	99.61	119.77	140.03
τ_1 (%)	67.97	44.41	23.81	10.26	7.42	5.24	4.31	3.74	3.19
τ_2 (%)	92.44	69.45	36.95	27.59	19.71	17.47	14.62	14.28	12.68
$n \cdot \tau_1$	1.36	2.22	2.38	2.05	2.23	2.10	2.16	2.24	2.23
$n \cdot \tau_2$	1.85	3.47	3.69	5.52	5.91	6.99	7.31	8.57	8.88
$\sqrt{n} \cdot \tau_2$	1.31	1.55	1.17	1.23	1.08	1.10	1.03	1.11	1.06

Table 3.2: CPU seconds (average of 10 instances) for solving (ET_S)

n	5	10	20	30	40	50	60	70	80	90	100	150
Method 1	0.01	0.01	0.02	0.06	0.20	0.45	0.95	1.94	3.04	5.08	8.04	58.4
Method 2	0.01	0.02	1.13	12.6	253	517	2433	9860	∞	∞	∞	∞

Although Method 2 works clearly better than Method 1 in terms of upper bound of $v(ET_S)$, it requires much more computational time. The most expensive part of Method 2 is in its first relaxation, computing the eigenvalue and its corresponding eigenvector of an $n^2 \times n^2$ matrix. In comparison, for Method 1 the corresponding part involves only an $n \times n$ matrix. Evidence in Table 3.2 shows that Method 1 can find a good quality solution very fast even for large size problems. We remark here that for $n = 100$, the sizes of the input data are already in the magnitude of 10^8.

3.5.2 Data with Known Optimal Solutions

The upper bounds appear to be quite loose in general, as one may observe from the previous numerical results. To test how good the solutions are without referring to the computed upper bounds, in this subsection we report the test results where the problem instances are constructed in such a way that the optimal solutions are known. By this we hope to get some impression, from a different angle, on the quality of the approximate solutions produced by our algorithms. We first randomly generate a

Table 3.3: Numerical ratios of (ET_S) with known optima for $n = 50$

m	5	10	20	30	40	50	100	150	200
Minimal ratio (%)	50	66	43	37	37	100	100	100	100
Maximal ratio (%)	100	100	100	100	100	100	100	100	100
Average ratio (%)	97	86	76	87	97	100	100	100	100
Optimality instances (%)	7	10	35	71	94	100	100	100	100

vector $\boldsymbol{a} \in \mathbb{S}^n$, and generate m symmetric matrices $\boldsymbol{A}_i \in \mathbb{R}^{n\times n}$ $(1 \le i \le m)$ with their eigenvalues lying in the interval $[-1, 1]$ and $\boldsymbol{A}_i\boldsymbol{a} = \boldsymbol{a}$ for $i = 1, 2, \ldots, m$. Then, we randomly generate a vector $\boldsymbol{b} \in \mathbb{S}^n$, and m symmetric matrices $\boldsymbol{B}_i \in \mathbb{R}^{n\times n}$ $(1 \le i \le m)$ with their eigenvalues in the interval $[-1, 1]$ and $\boldsymbol{B}_i\boldsymbol{b} = \boldsymbol{b}$ for $i = 1, 2, \ldots, m$. Define

$$F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{w}) = \sum_{i=1}^{m} \left(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_i\boldsymbol{y} \cdot \boldsymbol{z}^{\mathrm{T}}\boldsymbol{B}_i\boldsymbol{w}\right).$$

For this particular multilinear form $F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{w})$, it is easy to see that (ET_S) has an optimal solution $(\boldsymbol{a}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{b})$ and optimal value is m.

We generate such random instances with $n = 50$ for various m, and subsequently apply Method 1 to solve them. Since the optimal values are known, it is possible to compute the *exact* performance ratios, i.e., v_1/m. For each m, 200 random instances are generated and tested. The results are shown in Table 3.3, which suggest that our algorithm works very well and the performance ratios are much better than the theoretical worst-case bounds. Indeed, whenever $m \ge 50$ our algorithm always finds optimal solutions.

Chapter 4

Homogeneous Form Optimization with Quadratic Constraints

4.1 Introduction

This section studies optimizations of an important class of polynomial functions, namely, homogeneous polynomials, or forms. The constraint set is defined by homogeneous quadratic polynomial equalities or inequalities. Specifically, the models include maximizing a homogenous form over the Euclidean sphere

$$\boxed{\begin{array}{lll}(H_S) & \max & f(\boldsymbol{x}) \\ & \text{s.t.} & \boldsymbol{x} \in \mathbb{S}^n,\end{array}}$$

and maximizing a homogenous form over the intersection of co-centered ellipsoids

$$\boxed{\begin{array}{lll}(H_Q) & \max & f(\boldsymbol{x}) \\ & \text{s.t.} & \boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x} \leq 1,\ i = 1, 2, \ldots, m, \\ & & \boldsymbol{x} \in \mathbb{R}^n,\end{array}}$$

where $\boldsymbol{Q}_i \succeq 0$ for $k = 1, 2, \ldots, d$, and $\sum_{i=1}^m \boldsymbol{Q}_i \succ 0$.

As a general extension, we also consider optimizations on mixed forms, i.e.,

$$\text{Function M} \quad f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s) = F(\underbrace{\boldsymbol{x}^1, \boldsymbol{x}^1, \cdots, \boldsymbol{x}^1}_{d_1}, \underbrace{\boldsymbol{x}^2, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^2}_{d_2}, \cdots, \underbrace{\boldsymbol{x}^s, \boldsymbol{x}^s, \cdots, \boldsymbol{x}^s}_{d_s}),$$

where $d = d_1 + d_2 + \cdots + d_s$ is deemed as a fixed constant, and d-th order tensor $\boldsymbol{F} \in \mathbb{R}^{{n_1}^{d_1} \times {n_2}^{d_2} \times \cdots \times {n_s}^{d_s}}$ has partial symmetric property. The mixed form optimization

models include

$$\boxed{\begin{array}{lll}(M_S) & \max & f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s) \\ & \text{s.t.} & \boldsymbol{x}^k \in \mathbb{S}^{n_k},\ k = 1, 2, \ldots, s;\end{array}}$$

$$\boxed{\begin{array}{lll}(M_Q) & \max & f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s) \\ & \text{s.t.} & (\boldsymbol{x}^k)^{\mathrm{T}} \boldsymbol{Q}_{i_k}^k \boldsymbol{x}^k \le 1,\ k = 1, 2, \ldots, s,\ i_k = 1, 2, \ldots, m_k, \\ & & \boldsymbol{x}^k \in \mathbb{R}^{n_k},\ k = 1, 2, \ldots, s,\end{array}}$$

where $\boldsymbol{Q}_{i_k}^k \succeq 0$ and $\sum_{i_k=1}^{m_k} \boldsymbol{Q}_{i_k}^k \succ 0$ for $k = 1, 2, \ldots, s$, $i_k = 1, 2, \ldots, m_k$.

The model (M_S) is a generalization of (H_S) and (T_S) in Section 3.2, and (M_Q) is a generalization of (H_Q) and (T_Q) in Section 3.3. When the degree of the polynomial objective is odd, (H_S) is equivalent to

$$\begin{array}{lll}\max & f(\boldsymbol{x}) \\ \text{s.t.} & \boldsymbol{x} \in \bar{\mathbb{S}}^n.\end{array}$$

This is because we can always use $-\boldsymbol{x}$ to replace $\boldsymbol{x}$ if its objective value is negative, and can also scale the vector $\boldsymbol{x}$ along its direction to make it in $\mathbb{S}^n$. Thus (H_S) is a special case of (H_Q) when d is odd. However if d is even, the optimal value of (H_S) may be negative, while that of (H_Q) is always nonnegative since $\mathbf{0}$ is always a feasible solution of (H_Q). In the former case, the tensor $\boldsymbol{F}$ is called negative definite, i.e., $f(\boldsymbol{x}) < 0$ for all $\boldsymbol{x} \neq \mathbf{0}$.

The model (H_S) is in general NP-hard. When $d = 1$, (H_S) has a close-form solution, due to the Cauchy-Schwartz inequality; And when $d = 2$, (H_S) is to compute the largest eigenvalue of the symmetric matrix $\boldsymbol{F}$; However (H_S) becomes NP-hard when $d = 3$, first proven by Nesterov [90]. Interestingly, when $d \ge 3$, the model (H_S) is also regarded as computing the largest eigenvalue of the super-symmetric tensor $\boldsymbol{F}$, like the case $d = 2$ (see e.g., Qi [98]). Luo and Zhang [77] proposed the first polynomial-time randomized approximation algorithm with relative approximation ratio $\Omega\left(1/n^2\right)$ when $d = 4$, based on its quadratic SDP relaxation and randomization techniques.

For the model (H_Q): When $d = 1$, it can be formulated as a standard SOCP problem, which is solvable in polynomial-time; When $d = 2$, it is the well known QCQP problem discussed in Section 2.5, and known to be NP-hard in general. Nemirovski et al. [87] proposed a polynomial-time randomized approximation algorithm with approximation ratio $\Omega\left(1/\log m\right)$ based on SDP relaxation and randomization, and this ratio is actually tight; When $d = 4$, Luo and Zhang [77] established the relationship between

(H_Q) and its quadratic SDP relaxation, and proposed polynomial-time approximation algorithm when the number of constraints is one. Meanwhile, Ling et al. [73] proposed bi-quadratic optimization model, which is exactly the model (M_S) when $d = 4$ and $d_1 = d_2 = 2$. In particular, they established the equivalence between (M_S) and its quadratic SDP relaxation, based on which they proposed a polynomial-time randomized approximation algorithm with relative approximation ratio $\Omega\left(1/{n_2}^2\right)$.

For the model (M_S), the computational complexity is similar to its special cases (T_S) and (H_S). It is solvable in polynomial-time when $d \leq 2$, and is NP-hard when $d \geq 3$, which is claimed in Section 4.4.1. Moreover, when $d \geq 4$ and all d_i $(1 \leq k \leq s)$ are even, there is no polynomial-time approximation algorithm with a positive approximation ratio unless $P = NP$. This is verified in its simple case of $d = 4$ and $d_1 = d_2 = 2$ by Ling et al. [73]. The complexity of (M_Q) is also same to that of (H_Q), i.e., being solvable in polynomial-time when $d = 1$ and NP-hard when $d \geq 2$. Meanwhile, a special case of (M_Q) when $d = 4$ and $d_1 = d_2 = 2$ is the biquadratic form optimization over quadratic constraints, studied by Zhang et al. [123] and Ling al. [74]. In their work, the relationship between biquadratic optimization and its bilinear SDP relaxation is established, as well as some data dependent approximation bounds are derived.

In this chapter, we are going to present polynomial-time approximation algorithms with guaranteed worse-case performance ratios for the models concerned. Our algorithms work for any fixed degree d, and the approximation ratios improve all the previous works specialized to their particular degrees. The major break though for our work is the multilinear tensor form relaxation, in stead of quadratic SDP relaxation methods in [77, 73]. The relaxed multilinear optimization problems admit polynomial-time approximation algorithms discussed in Chapter 3. After we solved the relaxed problems, we merge a bunch of relaxed variables into one feasible solution by a link identity, and argue the quality ratios being deteriorated only by some constants, which is the main contribution in this chapter.

The approximate algorithms of (H_S) are presented in Section 4.2, followed by that of (H_Q) in Section 4.3. Models (M_S) and (M_Q) will be studied in Section 4.4. In Section 4.5, we discuss some applications with the models presented in this chapter. Finally, numerical performance of the proposed algorithms will be reported in Section 4.6.

4.2 Homogeneous Form with Spherical Constraint

The first model in this chapter is to maximize a homogenous polynomial function of fixed degree d over a sphere, i.e.,

$$\boxed{\begin{array}{lll}(H_S) & \max & f(\boldsymbol{x}) \\ & \text{s.t.} & \boldsymbol{x} \in \mathbb{S}^n.\end{array}}$$

Let $\boldsymbol{F}$ be the super-symmetric tensor satisfying $F(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x}}_{d}) = f(\boldsymbol{x})$. Then (H_S) can be *relaxed* to multilinear form optimization model (T_S) discussed in Chapter 3, as follows:

$$\begin{array}{lll}(\hat{H}_S) & \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\ & \text{s.t.} & \boldsymbol{x}^k \in \mathbb{S}^n,\ k = 1, 2, \ldots, d.\end{array}$$

Theorem 3.2.4 asserts that $(\hat{H}_S)$ can be solved approximately in polynomial-time, with approximation ratio $n^{-\frac{d-2}{2}}$. The key step is to draw a feasible solution of (H_S) from the approximate solution of $(\hat{H}_S)$. For this purpose, we establish the following link between (H_S) and $(\hat{H}_S)$.

Lemma 4.2.1 *Suppose $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^d \in \mathbb{R}^n$, and $\xi_1, \xi_2, \cdots, \xi_d$ are i.i.d. random variables, each taking values 1 and -1 with equal probability $1/2$. For any super-symmetric d-th order tensor $\boldsymbol{F}$ and function $f(\boldsymbol{x}) = F(\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x})$, it holds that*

$$\mathrm{E}\left[\prod_{i=1}^{d} \xi_i f\left(\sum_{k=1}^{d} \xi_k \boldsymbol{x}^k\right)\right] = d! F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d).$$

Proof. First we observe that

$$\begin{aligned}\mathrm{E}\left[\prod_{i=1}^{d} \xi_i f\left(\sum_{k=1}^{d} \xi_k \boldsymbol{x}^k\right)\right] &= \mathrm{E}\left[\prod_{i=1}^{d} \xi_i \sum_{1 \le k_1, k_2, \cdots, k_d \le d} F\left(\xi_{k_1} \boldsymbol{x}^{k_1}, \xi_{k_2} \boldsymbol{x}^{k_2}, \cdots, \xi_{k_d} \boldsymbol{x}^{k_d}\right)\right] \\ &= \sum_{1 \le k_1, k_2, \cdots, k_d \le d} \mathrm{E}\left[\prod_{i=1}^{d} \xi_i \prod_{j=1}^{d} \xi_{k_j} F\left(\boldsymbol{x}^{k_1}, \boldsymbol{x}^{k_2}, \cdots, \boldsymbol{x}^{k_d}\right)\right].\end{aligned}$$

If $\{k_1, k_2, \cdots, k_d\}$ is a permutation of $\{1, 2, \ldots, d\}$, then

$$\mathrm{E}\left[\prod_{i=1}^{d} \xi_i \prod_{j=1}^{d} \xi_{k_j}\right] = \mathrm{E}\left[\prod_{i=1}^{d} \xi_i^2\right] = 1;$$

Otherwise, there must be an index k_0 with $1 \le k_0 \le d$ and $k_0 \neq k_j$ for all $1 \le j \le d$. In the latter case,

$$\mathrm{E}\left[\prod_{i=1}^{d} \xi_i \prod_{j=1}^{d} \xi_{k_j}\right] = \mathrm{E}\left[\xi_{k_0}\right] \mathrm{E}\left[\prod_{1 \le i \le d, i \neq k_0} \xi_i \prod_{j=1}^{d} \xi_{k_j}\right] = 0.$$

Since the number of different permutations of $\{1, 2, \ldots, d\}$ is $d!$, by taking into account of the super-symmetric property of the tensor $\boldsymbol{F}$, the claimed relation follows. □

When d is odd, the identity in Lemma 4.2.1 can be rewritten as

$$d!F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) = \mathrm{E}\left[\prod_{i=1}^{d} \xi_i f\left(\sum_{k=1}^{d} \xi_k \boldsymbol{x}^k\right)\right] = \mathrm{E}\left[f\left(\sum_{k=1}^{d}\left(\prod_{i\neq k} \xi_i\right)\boldsymbol{x}^k\right)\right].$$

Since $\xi_1, \xi_2, \cdots, \xi_d$ are i.i.d. random variables taking values 1 or -1, by randomization we may find a particular binary vector $\boldsymbol{\beta} \in \mathbb{B}^d$, such that

$$f\left(\sum_{k=1}^{d}\left(\prod_{i\neq k} \beta_i\right)\boldsymbol{x}^k\right) \geq d!F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d). \tag{4.1}$$

We remark that d is considered a constant parameter in this thesis. Therefore, searching over all the combinations can be done, in principle, in constant time.

Let $\tilde{\boldsymbol{x}} = \sum_{k=1}^{d}\left(\prod_{i\neq k} \beta_i\right)\boldsymbol{x}^k$, and $\hat{\boldsymbol{x}} = \tilde{\boldsymbol{x}}/\|\tilde{\boldsymbol{x}}\|$. By the triangle inequality, we have $\|\tilde{\boldsymbol{x}}\| \leq d$, and thus

$$f(\hat{\boldsymbol{x}}) \geq d!d^{-d}F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d).$$

Combining with Theorem 3.2.4, we have

Theorem 4.2.2 *When $d \geq 3$ is odd, (H_S) admits a polynomial-time approximation algorithm with approximation ratio $\tau(H_S)$, where*

$$\tau(H_S) := d!\, d^{-d} n^{-\frac{d-2}{2}} = \Omega\left(n^{-\frac{d-2}{2}}\right).$$

The algorithm for approximately solving (H_S) with odd d is highlighted below.

Algorithm 4.2.1

- *INPUT: a d-th order super-symmetric tensor $\boldsymbol{F} \in \mathbb{R}^{n^d}$*

1 *Apply Algorithm 3.2.3 to solve the problem*

$$\begin{aligned} \max \quad & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\ \text{s.t.} \quad & \boldsymbol{x}^k \in \mathbb{S}^n,\ k = 1, 2, \ldots, d. \end{aligned}$$

approximately, with input $\boldsymbol{F}$ and output $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$.

2 *Compute $\boldsymbol{\beta} = \arg\max_{\boldsymbol{\xi}\in\mathbb{B}^d}\left\{f\left(\sum_{k=1}^{d} \xi_k \hat{\boldsymbol{x}}^k\right)\right\}$, or randomly generate $\boldsymbol{\beta}$ uniformly on $\mathbb{B}^d$ and repeat if necessary, until $f\left(\sum_{k=1}^{d} \beta_k \hat{\boldsymbol{x}}^k\right) \geq d!F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$.*

3 *Compute* $\hat{\boldsymbol{x}} = \sum_{k=1}^{d} \beta_k \hat{\boldsymbol{x}}^k / \| \sum_{k=1}^{d} \beta_k \hat{\boldsymbol{x}}^k \|$.

- *OUTPUT: a feasible solution* $\hat{\boldsymbol{x}} \in \mathbb{S}^n$.

We remark that it is unnecessary to enumerate all possible 2^d combinations in Step 2 of Algorithm 4.2.1, as (4.1) suggests that a simple randomization process will serve the same purpose, especially when d is large. In the latter case, we will end up with a *polynomial-time randomized approximation algorithm*; otherwise, the computational complexity of Algorithm 4.2.1 is deterministic and is polynomial-time for fixed d.

When d is even, the only easy case of (H_S) appears when $d = 2$, and even worse, we have the following:

Proposition 4.2.3 *If* $d = 4$, *then there is no polynomial-time approximation algorithm with a positive approximation ratio for* (H_S) *unless* $P = NP$.

Proof. Let $f(\boldsymbol{x}) = F(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x})$ with $\boldsymbol{F}$ being super-symmetric. We say quartic form $F(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x})$ is positive semidefinite if $F(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x}) \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$. It is well known that checking the positive semidefiniteness of $F(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x})$ is co-NP-complete. If we were able to find a polynomial-time approximation algorithm to get a positive approximation ratio $\tau \in (0, 1]$ for $v^* = \max_{\boldsymbol{x} \in \mathbb{S}^n} -F(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x})$, then this algorithm can be used to check the positive semidefiniteness of $F(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x})$. To see why, suppose this algorithm returns a feasible solution $\hat{\boldsymbol{x}}$ with $-F(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}) > 0$, then $F(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x})$ is not positive semidefinite. Otherwise the algorithm must return a feasible solution $\hat{\boldsymbol{x}}$ with $0 \geq -F(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}) \geq \tau v^*$, which implies $v^* \leq 0$; hence, $F(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x})$ is positive semidefinite in this case. Therefore, such algorithm cannot exist unless $P = NP$. □

This negative result rules out any polynomial-time approximation algorithm with a positive *absolute* approximation ratio for (H_S) when $d \geq 4$ is even. Thus we can only speak of *relative* approximation ratio. The following algorithm applies for (H_S) when d is even.

Algorithm 4.2.2

- *INPUT: a d-th order super-symmetric tensor* $\boldsymbol{F} \in \mathbb{R}^{n^d}$.

1 *Choose any vector* $\boldsymbol{x}^0 \in \mathbb{S}^n$ *and define a* d*-th order super-symmetric tensor* $\boldsymbol{H} \in \mathbb{R}^{n^d}$ *with respect to the homogeneous polynomial* $h(\boldsymbol{x}) = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x})^{d/2}$.

2 *Apply Algorithm 3.2.3 to solve the problem*

$$\begin{array}{ll} \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) - f(\boldsymbol{x}^0) H(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\ \text{s.t.} & \boldsymbol{x}^k \in \mathbb{S}^n,\ k = 1, 2, \ldots, d \end{array}$$

approximately, with input $\boldsymbol{F} - f(\boldsymbol{x}^0)\boldsymbol{H}$ *and output* $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$.

3 *Compute* $\boldsymbol{\beta} = \arg\max_{\boldsymbol{\xi} \in \mathbb{B}^d, \prod_{k=1}^d \xi_k = 1} \left\{ f\left(\frac{\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k}{\| \sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k \|} \right) \right\}$.

4 *Compute* $\hat{\boldsymbol{x}} = \arg\max \left\{ f(\boldsymbol{x}^0), f\left(\frac{\sum_{k=1}^d \beta_k \hat{\boldsymbol{x}}^k}{\| \sum_{k=1}^d \beta_k \hat{\boldsymbol{x}}^k \|} \right) \right\}$.

- *OUTPUT: a feasible solution* $\hat{\boldsymbol{x}} \in \mathbb{S}^n$.

Theorem 4.2.4 *When* $d \geq 4$ *is even,* (H_S) *admits a polynomial-time approximation algorithm with relative approximation ratio* $\tau(H_S)$.

Proof. Denote $\boldsymbol{H}$ to be the super-symmetric tensor with respect to the homogeneous polynomial $h(\boldsymbol{x}) = \|\boldsymbol{x}\|^d = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x})^{d/2}$. Explicitly, if we denote Π to be the set of all permutations of $\{1, 2, \ldots, d\}$, then

$$H(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) = \frac{1}{|\Pi|} \sum_{(i_1, i_2, \cdots, i_d) \in \Pi} \left((\boldsymbol{x}^{i_1})^{\mathrm{T}} \boldsymbol{x}^{i_2} \right) \left((\boldsymbol{x}^{i_3})^{\mathrm{T}} \boldsymbol{x}^{i_4} \right) \cdots \left((\boldsymbol{x}^{i_{d-1}})^{\mathrm{T}} \boldsymbol{x}^{i_d} \right).$$

For any $\boldsymbol{x}^k \in \mathbb{S}^n$ $(k = 1, 2, \ldots, d)$, we have $|H(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d)| \leq 1$ by applying the Cauchy-Schwartz inequality termwise.

Pick any fixed $\boldsymbol{x}^0 \in \mathbb{S}^n$, and consider the following problem

$$\begin{array}{lll} (\tilde{H}_S) & \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) - f(\boldsymbol{x}^0) H(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\ & \text{s.t.} & \boldsymbol{x}^k \in \mathbb{S}^n,\ k = 1, 2, \ldots, d. \end{array}$$

Applying Theorem 3.2.4 we obtain a solution $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$ in polynomial-time, with

$$F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) - f(\boldsymbol{x}^0) H(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) \geq \tilde{\tau}(T_S) v(\tilde{H}_S),$$

where $\tilde{\tau}(T_S) := n^{-\frac{d-2}{2}}$.

Let us first work on the case that

$$f(\boldsymbol{x}^0) - \underline{v}(H_S) \leq (\tilde{\tau}(T_S)/4) \left(v(H_S) - \underline{v}(H_S) \right). \tag{4.2}$$

Since $|H(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)| \leq 1$, we have

$$
\begin{aligned}
& F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) - \underline{v}(H_S) H(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) \\
= & F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) - f(\boldsymbol{x}^0) H(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) + \left(f(\boldsymbol{x}^0) - \underline{v}(H_S)\right) H(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) \\
\geq & \tilde{\tau}(T_S) v(\tilde{H}_S) - \left(f(\boldsymbol{x}^0) - \underline{v}(H_S)\right) \\
\geq & \tilde{\tau}(T_S) \left(v(H_S) - f(\boldsymbol{x}^0)\right) - \left(\tilde{\tau}(T_S)/4\right) \left(v(H_S) - \underline{v}(H_S)\right) \\
\geq & \left(\tilde{\tau}(T_S)(1 - \tilde{\tau}(T_S)/4) - \tilde{\tau}(T_S)/4\right) \left(v(H_S) - \underline{v}(H_S)\right) \\
\geq & \left(\tilde{\tau}(T_S)/2\right) \left(v(H_S) - \underline{v}(H_S)\right),
\end{aligned}
$$

where the second inequality is due to the fact that the optimal solution of (H_S) is feasible for $(\tilde{H}_S)$.

On the other hand, let $\xi_1, \xi_2, \cdots, \xi_d$ be i.i.d. random variables, each taking values 1 and -1 with equal probability 1/2. By symmetricity, we have Prob $\left\{\prod_{i=1}^d \xi_i = 1\right\} =$ Prob $\left\{\prod_{i=1}^d \xi_i = -1\right\} = 1/2$. Applying Lemma 4.2.1 we know

$$
\begin{aligned}
& d! \left(F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) - \underline{v}(H_S) H(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)\right) \\
= & \mathrm{E}\left[\prod_{i=1}^d \xi_i \left(f\left(\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right) - \underline{v}(H_S) h\left(\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right)\right)\right] \\
= & \mathrm{E}\left[f\left(\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right) - \underline{v}(H_S) \left\|\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right\|^d \middle| \prod_{i=1}^d \xi_i = 1\right] \mathrm{Prob}\left\{\prod_{i=1}^d \xi_i = 1\right\} \\
& - \mathrm{E}\left[f\left(\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right) - \underline{v}(H_S) \left\|\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right\|^d \middle| \prod_{i=1}^d \xi_i = -1\right] \mathrm{Prob}\left\{\prod_{i=1}^d \xi_i = -1\right\} \\
\leq & \frac{1}{2} \mathrm{E}\left[f\left(\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right) - \underline{v}(H_S) \left\|\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right\|^d \middle| \prod_{i=1}^d \xi_i = 1\right],
\end{aligned}
$$

where the last inequality is due to the fact that

$$
f\left(\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right) - \underline{v}(H_S) \left\|\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right\|^d \geq 0,
$$

since $\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k \Big/ \left\|\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right\| \in \mathbb{S}^n$. Thus by randomization, we can find $\boldsymbol{\beta} \in \mathbb{B}^d$ with $\prod_{i=1}^d \beta_i = 1$, such that

$$
\frac{1}{2}\left(f\left(\sum_{k=1}^d \beta_k \hat{\boldsymbol{x}}^k\right) - \underline{v}(H_S) \left\|\sum_{k=1}^d \beta_k \hat{\boldsymbol{x}}^k\right\|^d\right) \geq d! \left(\tilde{\tau}(T_S)/2\right) \left(v(H_S) - \underline{v}(H_S)\right).
$$

By letting $\hat{\boldsymbol{x}} = \sum_{k=1}^d \beta_k \hat{\boldsymbol{x}}^k \Big/ \left\|\sum_{k=1}^d \beta_k \hat{\boldsymbol{x}}^k\right\|$, and noticing $\|\sum_{k=1}^d \beta_k \hat{\boldsymbol{x}}^k\| \leq d$, we have

$$
f(\hat{\boldsymbol{x}}) - \underline{v}(H_S) \geq \frac{d! \tilde{\tau}(T_S) \left(v(H_S) - \underline{v}(H_S)\right)}{\|\sum_{k=1}^d \beta_k \hat{\boldsymbol{x}}^k\|^d} \geq \tau(H_S) \left(v(H_S) - \underline{v}(H_S)\right).
$$

Recall that the above inequality is derived under the condition that (4.2) holds. In case (4.2) does not hold, then

$$f(\boldsymbol{x}^0) - \underline{v}(H_S) > (\tilde{\tau}(T_S)/4)\,(v(H_S) - \underline{v}(H_S)) \geq \tau(H_S)\,(v(H_S) - \underline{v}(H_S))\,. \qquad (4.3)$$

By picking $\tilde{\boldsymbol{x}} = \arg\max\{f(\hat{\boldsymbol{x}}), f(\boldsymbol{x}^0)\}$, regardless whether (4.2) or (4.3) holds, we shall uniformly have $f(\tilde{\boldsymbol{x}}) - \underline{v}(H_S) \geq \tau(H_S)\,(v(H_S) - \underline{v}(H_S))$. □

4.3 Homogeneous Form with Ellipsoidal Constraints

We proceed a further generalization of the optimization models to include general ellipsoidal constraints.

$$\begin{array}{lll} (H_Q) & \max & f(\boldsymbol{x}) \\ & \text{s.t.} & \boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x} \leq 1,\ i = 1, 2, \ldots, m, \\ & & \boldsymbol{x} \in \mathbb{R}^n, \end{array}$$

where $f(\boldsymbol{x})$ is a homogenous polynomial of degree d, $\boldsymbol{Q}_i \succeq 0$ for $i = 1, 2, \ldots, m$, and $\sum_{i=1}^m \boldsymbol{Q}_i \succ 0$.

If we relax (H_Q) to the multilinear form optimization problem like (T_Q), and we have

$$\begin{array}{lll} (\tilde{H}_Q) & \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\ & \text{s.t.} & (\boldsymbol{x}^k)^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x}^k \leq 1,\ k = 1, 2, \ldots, d,\ i = 1, 2, \ldots, m, \\ & & \boldsymbol{x}^k \in \mathbb{R}^n,\ k = 1, 2, \ldots, d. \end{array}$$

Theorem 3.3.4 asserts an approximate solution for $(\tilde{H}_Q)$, together with Lemma 4.2.1 we propose the following algorithm for approximately solving (H_Q), no matter d is odd or even.

Algorithm 4.3.1

- *INPUT: a d-th order super-symmetric tensor* $\boldsymbol{F} \in \mathbb{R}^{n^d}$, *matrices* $\boldsymbol{Q}_i \in \mathbb{R}^{n\times n}$, $\boldsymbol{Q}_i \succeq 0$ *for all* $1 \leq i \leq m$ *with* $\sum_{i=1}^m \boldsymbol{Q}_i \succ 0$.

1 *Apply Algorithm 3.3.2 to solve the problem*

$$\begin{array}{lll} \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\ \text{s.t.} & (\boldsymbol{x}^k)^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x}^k \leq 1,\ k = 1, 2, \ldots, d,\ i = 1, 2, \ldots, m, \\ & \boldsymbol{x}^k \in \mathbb{R}^n,\ k = 1, 2, \ldots, d \end{array}$$

approximately, and get a feasible solution $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$.

2 *Compute* $\hat{\boldsymbol{x}} = \arg\max \left\{ f\left(\frac{1}{d}\sum_{k=1}^{d} \xi_k \hat{\boldsymbol{x}}^k\right), \boldsymbol{\xi} \in \mathbb{B}^d \right\}$.

- *OUTPUT: a feasible solution* $\hat{\boldsymbol{x}} \in \mathbb{R}^n$.

Although Algorithm 4.3.1 applies for both odd and even d of the model (H_Q), the approximation results are different, as the following theorems claimed.

Theorem 4.3.1 *When $d \geq 3$ is odd, (H_Q) admits a polynomial-time randomized approximation algorithm with approximation ratio $\tau(H_Q)$, where*

$$\tau(H_Q) := d!\, d^{-d} n^{-\frac{d-2}{2}} \Omega\left(\log^{-(d-1)} m\right) = \Omega\left(n^{-\frac{d-2}{2}} \log^{-(d-1)} m\right).$$

Proof. According to Theorem 3.3.4 we can find a feasible solution $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$ of $(\tilde{H}_Q)$ in polynomial-time, such that

$$F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) \geq \tilde{\tau}(H_Q) v(\tilde{H}_Q) \geq \tilde{\tau}(H_Q) v(H_Q), \tag{4.4}$$

where $\tilde{\tau}(H_Q) := n^{-\frac{d-2}{2}} \Omega\left(\log^{-(d-1)} m\right)$. By (4.1), we can find a binary vector $\boldsymbol{\beta} \in \mathbb{B}^d$ in polynomial-time, such that

$$f\left(\sum_{i=1}^{d} \beta_i \hat{\boldsymbol{x}}^i\right) \geq d! F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d).$$

Notice that for any $1 \leq k \leq m$,

$$\begin{aligned}
&\left(\sum_{i=1}^{d} \beta_i \hat{\boldsymbol{x}}^i\right)^{\mathrm{T}} \boldsymbol{Q}_k \left(\sum_{j=1}^{d} \beta_j \hat{\boldsymbol{x}}^j\right) = \sum_{i,j=1}^{d} \beta_i (\hat{\boldsymbol{x}}^i)^{\mathrm{T}} \boldsymbol{Q}_k \beta_j \hat{\boldsymbol{x}}^j \\
&= \sum_{i,j=1}^{d} \left(\beta_i \boldsymbol{Q}_k^{\frac{1}{2}} \hat{\boldsymbol{x}}^i\right)^{\mathrm{T}} \left(\beta_j \boldsymbol{Q}_k^{\frac{1}{2}} \hat{\boldsymbol{x}}^j\right) \leq \sum_{i,j=1}^{d} \left\|\beta_i \boldsymbol{Q}_k^{\frac{1}{2}} \hat{\boldsymbol{x}}^i\right\| \left\|\beta_j \boldsymbol{Q}_k^{\frac{1}{2}} \hat{\boldsymbol{x}}^j\right\| \\
&= \sum_{i,j=1}^{d} \sqrt{(\hat{\boldsymbol{x}}^i)^{\mathrm{T}} \boldsymbol{Q}_k \hat{\boldsymbol{x}}^i} \sqrt{(\hat{\boldsymbol{x}}^j)^{\mathrm{T}} \boldsymbol{Q}_k \hat{\boldsymbol{x}}^j} \leq \sum_{i,j=1}^{d} 1 \cdot 1 = d^2.
\end{aligned} \tag{4.5}$$

If we denote $\hat{\boldsymbol{x}} = \frac{1}{d}\sum_{i=1}^{d} \beta_i \hat{\boldsymbol{x}}^i$, then $\hat{\boldsymbol{x}}$ is a feasible solution for (H_Q), satisfying

$$f(\hat{\boldsymbol{x}}) \geq d^{-d} d! F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) \geq d^{-d} d! \tilde{\tau}(H_Q)\, v(H_Q) = \tau(H_Q)\, v(H_Q).$$

□

Theorem 4.3.2 *When $d \geq 4$ is even, (H_Q) admits a polynomial-time randomized approximation algorithm with relative approximation ratio $\tau(H_Q)$.*

Proof. First, we observe that $v(H_Q) \leq v(\tilde{H}_Q)$ and $\underline{v}(H_Q) \geq \underline{v}(\tilde{H}_Q) = -v(\tilde{H}_Q)$. Therefore, $2\,v(\tilde{H}_Q) \geq v(H_Q) - \underline{v}(H_Q)$. Let $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$ be the feasible solution for $(\tilde{H}_Q)$ as in the proof of Theorem 4.3.1, satisfying (4.4). According to (4.5), it follows that $\hat{\boldsymbol{x}} = \arg\max\left\{f\left(\frac{1}{d}\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right), \boldsymbol{\xi} \in \mathbb{B}^d\right\}$ is feasible for (H_Q), where $\xi_1, \xi_2, \cdots, \xi_d$ are i.i.d. random variables, each taking values 1 and -1 with equal probability $1/2$. Therefore, by Lemma 4.2.1 we have

$$
\begin{aligned}
&2\,d!F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) = 2\,\mathrm{E}\left[\prod_{i=1}^d \xi_i f\left(\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right)\right] \\
&= \mathrm{E}\left[f\left(\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right) - d^d \underline{v}(H_Q) \,\middle|\, \prod_{i=1}^d \xi_i = 1\right] - \mathrm{E}\left[f\left(\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right) - d^d \underline{v}(H_Q) \,\middle|\, \prod_{i=1}^d \xi_i = -1\right] \\
&\leq \mathrm{E}\left[f\left(\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right) - d^d \underline{v}(H_Q) \,\middle|\, \prod_{i=1}^d \xi_i = 1\right] \leq d^d f(\hat{\boldsymbol{x}}) - d^d \underline{v}(H_Q).
\end{aligned}
$$

According to (4.4), this implies that

$$
f(\hat{\boldsymbol{x}}) - \underline{v}(H_Q) \geq 2d^{-d} d!F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) \geq 2\tau(H_Q) v(\tilde{H}_Q) \geq \tau(H_Q)\left(v(H_Q) - \underline{v}(H_Q)\right).
$$

□

4.4 Mixed Form with Quadratic Constraints

In this section, we further extend polynomial optimization models to the mixed forms. Specifically, we study the following two models:

$$
\begin{array}{lll}
(M_S) & \max & f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s) \\
& \text{s.t.} & \boldsymbol{x}^k \in \mathbb{S}^{n_k},\ k = 1, 2, \ldots, s;
\end{array}
$$

$$
\begin{array}{lll}
(M_Q) & \max & f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s) \\
& \text{s.t.} & (\boldsymbol{x}^k)^{\mathrm{T}} \boldsymbol{Q}_{i_k}^k \boldsymbol{x}^k \leq 1,\ k = 1, 2, \ldots, s,\ i_k = 1, 2, \ldots, m_k, \\
& & \boldsymbol{x}^k \in \mathbb{R}^{n_k},\ k = 1, 2, \ldots, s,
\end{array}
$$

where $\boldsymbol{Q}_{i_k}^k \succeq 0$ and $\sum_{i_k=1}^{m_k} \boldsymbol{Q}_{i_k}^k \succ 0$ for $k = 1, 2, \ldots, s$, $i_k = 1, 2, \ldots, m_k$. Here we assume that $n_1 \leq n_2 \leq \cdots \leq n_s$.

Both (T_S) in Section 3.2 and (H_S) in Section 4.2 are special cases of (M_S), and both (T_Q) in Section 3.3 and (H_Q) in Section 4.3 are special cases of (M_Q). In particular,

(M_S) is a generalization of the bi-quadratic optimization model discussed in Ling et al. [73], specialized to $d = 4$ and $d_1 = d_2 = 2$.

4.4.1 Mixed Form with Spherical Constraints

Let us study the optimization model (M_S). First, we have the following hardness result.

Proposition 4.4.1 *If $d = 3$, then (M_S) is NP-hard.*

Proof. We need verify the NP-hardness in three cases of $d = 3$, i.e., $d_1 = 3$, $d_1 = 2$ and $d_2 = 1$, and $d_1 = d_2 = d_3 = 1$. The case of $d_1 = 3$ is exactly (H_S) with $d = 3$, whose NP-hardness is proven by Nesterov [90], and the case of $d_1 = d_2 = d_3 = 1$ is exactly (T_S) with $d = 3$, whose NP-hardness is proven in Proposition 3.2.2.

When $d_1 = 2$ and $d_2 = 1$, in its special case $n_1 = n_2 = n$ and $\boldsymbol{F} \in \mathbb{R}^{n^3}$ satisfying $F_{ijk} = F_{jik}$ for all $1 \le i, j, k \le n$, we notice the following form of (T_S) is NP-hard in the proof of Proposition 3.2.2

$$\begin{array}{lll} (\check{T}_S) & \max & F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \\ & \text{s.t.} & \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{S}^n. \end{array}$$

We are going to show that the optimal value of $(\check{T}_S)$ is equal to the optimal value of this special case

$$\begin{array}{lll} (\check{M}_S) & \max & F(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{z}) \\ & \text{s.t.} & \boldsymbol{x}, \boldsymbol{z} \in \mathbb{S}^n. \end{array}$$

It is obvious $v(\check{T}_S) \ge v(\check{M}_S)$. Now choose any optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ of $(\check{T}_S)$ and compute the matrix $\boldsymbol{M} = F(\cdot, \cdot, \boldsymbol{z}^*)$. Since $\boldsymbol{M}$ is symmetric, we can compute an eigenvector $\hat{\boldsymbol{x}}$ corresponding to the largest absolute eigenvalue λ (which is also the largest singular value) in polynomial-time. Observe that

$$|F(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \boldsymbol{z}^*)| = |\hat{\boldsymbol{x}}^{\mathrm{T}} \boldsymbol{M} \hat{\boldsymbol{x}}| = \lambda = \max_{\boldsymbol{x}, \boldsymbol{y} \in \mathbb{S}^n} \boldsymbol{x}^{\mathrm{T}} \boldsymbol{M} \boldsymbol{y} = \max_{\boldsymbol{x}, \boldsymbol{y} \in \mathbb{S}^n} F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}^*) = F(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*) = v(\check{T}_S),$$

which implies either $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \boldsymbol{z}^*)$ or $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, -\boldsymbol{z}^*)$ is an optimal solution of $(\check{T}_S)$. Therefore $v(\check{T}_S) \le v(\check{M}_S)$, and this claims $v(\check{T}_S) = v(\check{M}_S)$. If $(\check{M}_S)$ can be solved in polynomial-time, then its optimal solution is also an optimal solution of $(\check{T}_S)$, which solves $(\check{T}_S)$ in polynomial-time, leading to a contradiction. □

Now, we focus on polynomial-time approximation algorithms as before. Similar to the relaxation in Section 4.2 in handling homogeneous polynomial optimizations, if we

relax (M_S) to the multilinear optimization $(\mathcal{T}_S)$, then by Theorem 3.2.4 we are able to find $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$ with $\|\boldsymbol{x}^k\| = 1$ for all $1 \le k \le d$ in polynomial-time, such that

$$F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) \ge \tilde{\tau}(M_S) v(M_S). \tag{4.6}$$

where

$$\tilde{\tau}(M_S) := \begin{cases} \left(\frac{\prod_{k=1}^{s-1} n_k{}^{d_k}}{n_{s-1}} \right)^{-\frac{1}{2}} & d_s = 1, \\ \left(\frac{\prod_{k=1}^{s} n_k{}^{d_k}}{n_s{}^2} \right)^{-\frac{1}{2}} & d_s \ge 2. \end{cases}$$

In order to draw a feasible solution of (M_S) from $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$, we need apply the link identity in Lemma 4.2.1 more carefully. Comparable approximation results as (H_S) can be similarly derived.

Theorem 4.4.2 *If $d \ge 3$ and one of $d_k\,(k = 1, 2, \ldots, s)$ is odd, then (M_S) admits a polynomial-time approximation algorithm with approximation ratio $\hat{\tau}(M_S)$, where*

$$\begin{aligned} \hat{\tau}(M_S) &:= \tilde{\tau}(M_S) \prod_{1 \le k \le s,\, 3 \le d_k} \frac{d_k!}{d_k{}^{d_k}} = \Omega\left(\tilde{\tau}(M_S)\right) \\ &= \begin{cases} \left(\prod_{1 \le k \le s,\, 3 \le d_k} \frac{d_k!}{d_k{}^{d_k}} \right) \left(\frac{\prod_{k=1}^{s-1} n_k{}^{d_k}}{n_{s-1}} \right)^{-\frac{1}{2}} & d_s = 1, \\ \left(\prod_{1 \le k \le s,\, 3 \le d_k} \frac{d_k!}{d_k{}^{d_k}} \right) \left(\frac{\prod_{k=1}^{s} n_k{}^{d_k}}{n_s{}^2} \right)^{-\frac{1}{2}} & d_s \ge 2. \end{cases} \end{aligned}$$

To avoid messy notations and better understand the main ideas of the algorithm and the proof, here we only consider a special case $(\hat{M}_S)$, which is easily extended to general (M_S).

$$\begin{aligned} (\hat{M}_S) \quad \max \quad & F(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{z}, \boldsymbol{z}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathbb{S}^{n_1}, \boldsymbol{y} \in \mathbb{S}^{n_2}, \boldsymbol{z} \in \mathbb{S}^{n_3}. \end{aligned}$$

By (4.6), we are able to find $\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{x}^3, \boldsymbol{x}^4 \in \mathbb{S}^{n_1}, \boldsymbol{y}^1, \boldsymbol{y}^2 \in \mathbb{S}^{n_2}$, and $\boldsymbol{z}^1, \boldsymbol{z}^2, \boldsymbol{z}^3 \in \mathbb{S}^{n_3}$ in polynomial-time, such that

$$F(\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{x}^3, \boldsymbol{x}^4, \boldsymbol{y}^1, \boldsymbol{y}^2, \boldsymbol{z}^1, \boldsymbol{z}^2, \boldsymbol{z}^3) \ge \tilde{\tau}(M_S) v(\hat{M}_S).$$

Let us first fix $(\boldsymbol{y}^1, \boldsymbol{y}^2, \boldsymbol{z}^1, \boldsymbol{z}^2, \boldsymbol{z}^3)$ and work for the problem

$$\begin{aligned} \max \quad & F(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x}, \boldsymbol{y}^1, \boldsymbol{y}^2, \boldsymbol{z}^1, \boldsymbol{z}^2, \boldsymbol{z}^3) \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathbb{S}^{n_1}. \end{aligned}$$

Using the same argument in proving Theorem 4.2.2, we are able to find $\hat{\boldsymbol{x}} \in \mathbb{S}^{n_1}$, such that either $F(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \boldsymbol{y}^1, \boldsymbol{y}^2, \boldsymbol{z}^1, \boldsymbol{z}^2, \boldsymbol{z}^3)$ or $F(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \boldsymbol{y}^1, \boldsymbol{y}^2, -\boldsymbol{z}^1, \boldsymbol{z}^2, \boldsymbol{z}^3)$ will be no less than $4!4^{-4}F(\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{x}^3, \boldsymbol{x}^4, \boldsymbol{y}^1, \boldsymbol{y}^2, \boldsymbol{z}^1, \boldsymbol{z}^2, \boldsymbol{z}^3)$, whereas in the latter case, use $-\boldsymbol{z}^1$ to update $\boldsymbol{z}^1$. Here even degree ($d_1 = 4$) for $\boldsymbol{x}$ makes no trouble, as we can always move the negative sign into $\boldsymbol{z}^1$. We call this process to be an adjustment of the variable $\boldsymbol{x}$. Up till now the approximation bound is $4!4^{-4}\tilde{\tau}(M_S)$.

Next we work on adjustment of variable $\boldsymbol{y}$ and consider the problem

$$\begin{array}{ll} \max & |F(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{y}, \boldsymbol{z}^1, \boldsymbol{z}^2, \boldsymbol{z}^3)| \\ \text{s.t.} & \boldsymbol{y} \in \mathbb{S}^{n_2}. \end{array}$$

This is the matrix largest absolute eigenvalue problem and can be solved in polynomial-time. Denote its optimal solution to be $\hat{\boldsymbol{y}}$, update $\boldsymbol{z}^1$ with $-\boldsymbol{z}^1$ if necessary, and we keep the approximation bound $4!4^{-4}\tilde{\tau}(M_S)$ for the solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{y}}, \boldsymbol{z}^1, \boldsymbol{z}^2, \boldsymbol{z}^3)$.

The last adjustment of the variable $\boldsymbol{z}$ is straightforward. Similar to the adjustment of the variable $\boldsymbol{x}$, by focusing on

$$\begin{array}{ll} \max & F(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{y}}, \boldsymbol{z}, \boldsymbol{z}, \boldsymbol{z}) \\ \text{s.t.} & \boldsymbol{z} \in \mathbb{S}^{n_3}, \end{array}$$

we can find $\hat{\boldsymbol{z}} \in \mathbb{S}^{n_3}$ in polynomial-time, such that the solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \hat{\boldsymbol{z}}, \hat{\boldsymbol{z}})$ admits an approximation bound $3!3^{-3}4!4^{-4}\tilde{\tau}(M_S)$.

We remark here that the variable $\boldsymbol{z}$ is the last variable for adjustment, since we cannot move the negative sign to other adjusted variables if the degree of $\boldsymbol{z}$ is even. That is why we require one of d_k to be odd as the condition of Theorem 4.4.2, where we can always adjust the last variable with an odd degree.

However, if all d_k ($k = 1, 2, \ldots, s$) are even, we can only hope for a *relative* approximation ratio. For its simplest case when $d = 4$ and $d_1 = d_2 = 2$, the bi-quadratic optimization model $\max_{\boldsymbol{x} \in \mathbb{S}^{n_1}, \boldsymbol{y} \in \mathbb{S}^{n_2}} F(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{y})$ does not admit any polynomial-time approximation algorithm with a positive approximation ratio by Ling et al. [73]. Before working on this even case, let us first introduce the following link extended from Lemma 4.2.1.

Lemma 4.4.3 *Suppose* $\boldsymbol{x}^k \in \mathbb{R}^{n_1}$ $(1 \le k \le d_1)$, $\boldsymbol{x}^k \in \mathbb{R}^{n_2}$ $(d_1 + 1 \le k \le d_1 + d_2)$, $\cdots$, $\boldsymbol{x}^k \in \mathbb{R}^{n_s}$ $(d_1 + d_2 + \cdots + d_{s-1} + 1 \le k \le d_1 + d_2 + \cdots + d_s = d)$, *and* $\xi_1, \xi_2, \cdots, \xi_d$ *are i.i.d. random variables, each taking values* 1 *and* -1 *with equal prob-*

ability 1/2. *Denote*

$$\boldsymbol{x}_\xi^1 = \sum_{k=1}^{d_1} \xi_k \boldsymbol{x}^k,\ \boldsymbol{x}_\xi^2 = \sum_{k=d_1+1}^{d_1+d_2} \xi_k \boldsymbol{x}^k,\ \cdots,\ \boldsymbol{x}_\xi^s = \sum_{k=d_1+d_2+\cdots+d_{s-1}+1}^{d} \xi_k \boldsymbol{x}^k. \tag{4.7}$$

For any partial symmetric d-th order tensor $\boldsymbol{F} \in \mathbb{R}^{n_1{}^{d_1} \times n_2{}^{d_2} \times \cdots \times n_s{}^{d_s}}$ *and function*

$$f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s) = F(\underbrace{\boldsymbol{x}^1, \boldsymbol{x}^1, \cdots, \boldsymbol{x}^1}_{d_1}, \underbrace{\boldsymbol{x}^2, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^2}_{d_2}, \cdots, \underbrace{\boldsymbol{x}^s, \boldsymbol{x}^s, \cdots, \boldsymbol{x}^s}_{d_s}),$$

it holds that

$$\mathrm{E}\left[\prod_{i=1}^{d} \xi_i f\left(\boldsymbol{x}_\xi^1, \boldsymbol{x}_\xi^2, \cdots, \boldsymbol{x}_\xi^s\right)\right] = \prod_{k=1}^{s} d_k! F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d).$$

This lemma is easy to prove by applying Lemma 4.2.1 s times.

Theorem 4.4.4 *If* $d \geq 4$ *and all* d_k $(k = 1, 2, \ldots, s)$ *are even, then* (M_S) *admits a polynomial-time approximation algorithm with relative approximation ratio* $\tau(M_S)$, *where*

$$\begin{aligned} \tau(M_S) &:= \tilde{\tau}(M_S) \prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}} = \Omega\left(\tilde{\tau}(M_S)\right) \\ &= \begin{cases} \left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}}\right) \left(\frac{\prod_{k=1}^{s-1} {n_k}^{d_k}}{n_{s-1}}\right)^{-\frac{1}{2}} & d_s = 1, \\ \left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}}\right) \left(\frac{\prod_{k=1}^{s} {n_k}^{d_k}}{{n_s}^2}\right)^{-\frac{1}{2}} & d_s \geq 2. \end{cases} \end{aligned}$$

Proof. Denote $\boldsymbol{H} \in \mathbb{R}^{n_1{}^{d_1} \times n_2{}^{d_2} \times \cdots \times n_s{}^{d_s}}$ to be the partial symmetric d-th order tensor with respect to the mixed form

$$h(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s) = \prod_{k=1}^{s} \|\boldsymbol{x}^k\|^{d_k} = \prod_{k=1}^{s} \left((\boldsymbol{x}^k)^{\mathrm{T}} \boldsymbol{x}^k\right)^{\frac{d_k}{2}}.$$

Choose any fixed $\hat{\boldsymbol{x}}^k \in \mathbb{S}^{n_k}$ for $k = 1, 2, \ldots, s$, and consider the following problem

$$\begin{aligned} (\tilde{M}_S) \quad \max \quad & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) - f(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^s) H(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\ \text{s.t.} \quad & \|\boldsymbol{x}^k\| = 1,\ k = 1, 2, \ldots, d. \end{aligned}$$

Applying Theorem 3.2.4 we obtain a solution $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$ with $\|\hat{\boldsymbol{x}}^k\| = 1$ for $k = 1, 2, \ldots, d$ in polynomial-time, such that

$$F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) - f(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^s) H(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) \geq \tilde{\tau}(M_S) v(\tilde{M}_S).$$

Let us start with the case that

$$f(\check{\boldsymbol{x}}^1, \check{\boldsymbol{x}}^2, \cdots, \check{\boldsymbol{x}}^s) - \underline{v}(M_S) \leq (\tilde{\tau}(M_S)/4)\,(v(M_S) - \underline{v}(M_S))\,. \tag{4.8}$$

Notice that $|H(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)| \leq 1$, we have

$$\begin{aligned}
& F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) - \underline{v}(M_S) H(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) \\
= \;& F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) - f(\check{\boldsymbol{x}}^1, \check{\boldsymbol{x}}^2, \cdots, \check{\boldsymbol{x}}^s) H(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) \\
& + \left(f(\check{\boldsymbol{x}}^1, \check{\boldsymbol{x}}^2, \cdots, \check{\boldsymbol{x}}^s) - \underline{v}(M_S)\right) H(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) \\
\geq \;& \tilde{\tau}(M_S) v(\tilde{M}_S) - \left(f(\check{\boldsymbol{x}}^1, \check{\boldsymbol{x}}^2, \cdots, \check{\boldsymbol{x}}^s) - \underline{v}(M_S)\right) \\
\geq \;& \tilde{\tau}(M_S) \left(v(M_S) - f(\check{\boldsymbol{x}}^1, \check{\boldsymbol{x}}^2, \cdots, \check{\boldsymbol{x}}^s)\right) - (\tilde{\tau}(M_S)/4)\,(v(M_S) - \underline{v}(M_S)) \\
\geq \;& (\tilde{\tau}(M_S)(1 - \tilde{\tau}(M_S)/4) - \tilde{\tau}(M_S)/4)\,(v(M_S) - \underline{v}(M_S)) \\
\geq \;& (\tilde{\tau}(M_S)/2)\,(v(M_S) - \underline{v}(M_S))\,,
\end{aligned}$$

where the second inequality is due to the fact that the optimal solution of (M_S) is feasible for $(\tilde{M}_S)$.

On the other hand, using the notation of (4.7) and applying Lemma 4.4.3, we have

$$\begin{aligned}
& \prod_{k=1}^{s} d_k! \left(F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) - \underline{v}(M_S) H(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)\right) \\
= \;& \mathrm{E}\left[\prod_{i=1}^{d} \xi_i \left(f\left(\hat{\boldsymbol{x}}_\xi^1, \hat{\boldsymbol{x}}_\xi^2, \cdots, \hat{\boldsymbol{x}}_\xi^s\right) - \underline{v}(M_S) h\left(\hat{\boldsymbol{x}}_\xi^1, \hat{\boldsymbol{x}}_\xi^2, \cdots, \hat{\boldsymbol{x}}_\xi^s\right)\right)\right] \\
= \;& \mathrm{E}\left[f\left(\hat{\boldsymbol{x}}_\xi^1, \hat{\boldsymbol{x}}_\xi^2, \cdots, \hat{\boldsymbol{x}}_\xi^s\right) - \underline{v}(M_S) \prod_{k=1}^{s} \|\hat{\boldsymbol{x}}_\xi^k\|^{d_k} \,\middle|\, \prod_{i=1}^{d} \xi_i = 1\right] \mathrm{Prob}\left\{\prod_{i=1}^{d} \xi_i = 1\right\} \\
& - \mathrm{E}\left[f\left(\hat{\boldsymbol{x}}_\xi^1, \hat{\boldsymbol{x}}_\xi^2, \cdots, \hat{\boldsymbol{x}}_\xi^s\right) - \underline{v}(M_S) \prod_{k=1}^{s} \|\hat{\boldsymbol{x}}_\xi^k\|^{d_k} \,\middle|\, \prod_{i=1}^{d} \xi_i = -1\right] \mathrm{Prob}\left\{\prod_{i=1}^{d} \xi_i = -1\right\} \\
\leq \;& \frac{1}{2} \mathrm{E}\left[f\left(\hat{\boldsymbol{x}}_\xi^1, \hat{\boldsymbol{x}}_\xi^2, \cdots, \hat{\boldsymbol{x}}_\xi^s\right) - \underline{v}(M_S) \prod_{k=1}^{s} \|\hat{\boldsymbol{x}}_\xi^k\|^{d_k} \,\middle|\, \prod_{i=1}^{d} \xi_i = 1\right],
\end{aligned}$$

where the last inequality is due to $f\left(\hat{\boldsymbol{x}}_\xi^1, \hat{\boldsymbol{x}}_\xi^2, \cdots, \hat{\boldsymbol{x}}_\xi^s\right) - \underline{v}(M_S) \prod_{k=1}^{s} \|\hat{\boldsymbol{x}}_\xi^k\|^{d_k} \geq 0$, since $\left(\hat{\boldsymbol{x}}_\xi^1/\|\hat{\boldsymbol{x}}_\xi^1\|, \hat{\boldsymbol{x}}_\xi^2/\|\hat{\boldsymbol{x}}_\xi^2\|, \cdots, \hat{\boldsymbol{x}}_\xi^s/\|\hat{\boldsymbol{x}}_\xi^s\|\right)$ is feasible for (M_S). Thus, there is a binary vector $\boldsymbol{\beta} \in \mathbb{B}^d$ with $\prod_{i=1}^{d} \beta_i = 1$, such that

$$\frac{1}{2}\left(f\left(\hat{\boldsymbol{x}}_\beta^1, \hat{\boldsymbol{x}}_\beta^2, \cdots, \hat{\boldsymbol{x}}_\beta^s\right) - \underline{v}(M_S) \prod_{k=1}^{s} \|\hat{\boldsymbol{x}}_\beta^k\|^{d_k}\right) \geq \prod_{k=1}^{s} d_k! \,(\tilde{\tau}(M_S)/2)\,(v(M_S) - \underline{v}(M_S))\,.$$

by letting $\check{\boldsymbol{x}}^k = \hat{\boldsymbol{x}}_\beta^k/\|\hat{\boldsymbol{x}}_\beta^k\|$ for $k = 1, 2, \ldots, s$, and noticing $\|\hat{\boldsymbol{x}}_\beta^k\| \leq d_k$, we have

$$\begin{aligned}
f(\check{\boldsymbol{x}}^1, \check{\boldsymbol{x}}^2, \cdots, \check{\boldsymbol{x}}^s) - \underline{v}(M_S) &\geq \tilde{\tau}(M_S) \prod_{k=1}^{s} d_k! \|\hat{\boldsymbol{x}}_\beta^k\|^{-d_k} \,(v(M_S) - \underline{v}(M_S)) \\
&\geq \tau(M_S)\,(v(M_S) - \underline{v}(M_S))\,.
\end{aligned}$$

Recall that the above inequality is derived under the condition that (4.8) holds. In case (4.8) does not hold, then we shall have

$$f(\check{\boldsymbol{x}}^1, \check{\boldsymbol{x}}^2, \cdots, \check{\boldsymbol{x}}^s) - \underline{v}(M_S) > (\tilde{\tau}(M_S)/4)\,(v(M_S) - \underline{v}(M_S)) \geq \tau(M_S)\,(v(M_S) - \underline{v}(M_S))\,.$$

By picking $(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s) = \arg\max\{f(\tilde{\boldsymbol{x}}^1, \tilde{\boldsymbol{x}}^2, \cdots, \tilde{\boldsymbol{x}}^s), f(\check{\boldsymbol{x}}^1, \check{\boldsymbol{x}}^2, \cdots, \check{\boldsymbol{x}}^s)\}$, we shall uniformly have $f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s) - \underline{v}(M_S) \geq \tau(M_S)\,(v(M_S) - \underline{v}(M_S))$. □

4.4.2 Mixed Form with Ellipsoidal Constraints

Finally, let us discuss the most general model (M_Q) for homogeneous polynomial optimization with quadratic constraints. We have similar results as of (M_S) in Section 4.4.1.

Theorem 4.4.5 *If $d \geq 3$ and one of $d_k\,(k = 1, 2, \ldots, s)$ is odd, then (M_Q) admits a polynomial-time randomized approximation algorithm with approximation ratio $\tau(M_Q)$, where*

$$\begin{aligned}
\tau(M_Q) \;&:=\; \tilde{\tau}(M_S)\,\Omega\left(\log^{-(d-1)} m\right) \prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}} = \Omega\left(\tilde{\tau}(M_S)\,\log^{-(d-1)} m\right) \\
&= \begin{cases} \left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}}\right) \left(\frac{\prod_{k=1}^{s-1} {n_k}^{d_k}}{n_{s-1}}\right)^{-\frac{1}{2}} \Omega\left(\log^{-(d-1)} m\right) & d_s = 1, \\ \left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}}\right) \left(\frac{\prod_{k=1}^{s} {n_k}^{d_k}}{{n_s}^2}\right)^{-\frac{1}{2}} \Omega\left(\log^{-(d-1)} m\right) & d_s \geq 2, \end{cases}
\end{aligned}$$

and $m = \max_{1 \leq k \leq s}\{m_k\}$.

The proof of Theorem 4.4.5 is very similar to that of Theorem 4.4.2, where a typical example is illustrated. Here we only highlight the main ideas. First we relax (M_Q) to the multilinear form optimization (T_Q) which finds a feasible solution for (T_Q) with an approximation ratio $\tilde{\tau}(M_S)\Omega\left(\log^{-(d-1)} m\right)$. Then, Lemma 4.4.3 servers as a bridge from that solution to a feasible solution for (M_Q). Specifically, we may adjust the solution of (T_Q) one by one. During each adjustment, we apply Lemma 4.2.1 once, with the approximation ratio deteriorating no worse than $d_k! {d_k}^{-d_k}$. After s times of adjustments, we are able to get a feasible solution for (M_Q) with performance ratio $\tau(M_Q)$. Besides, the feasibility of the solution so-obtained is guaranteed by (4.5).

Theorem 4.4.6 *If $d \geq 4$ and all $d_k\,(k = 1, 2, \ldots, s)$ are even, then (M_Q) admits a polynomial-time randomized approximation algorithm with relative approximation ratio $\tau(M_Q)$.*

Proof. The proof is analogous to that of Theorem 4.3.2. The main differences are: (i) we use Lemma 4.4.3 instead of invoking Lemma 4.2.1 directly; and (ii) we use $f\left(\frac{1}{d_1}\hat{\boldsymbol{x}}_\xi^1, \frac{1}{d_2}\hat{\boldsymbol{x}}_\xi^2, \cdots, \frac{1}{d_s}\hat{\boldsymbol{x}}_\xi^s\right)$ instead of $f\left(\frac{1}{d}\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right)$ during the randomization process. □

4.5 Applications

To better appreciate the homogeneous polynomial optimization models presented in this chapter, in this section we are going to present a few examples rising from various applications. In particular we shall discuss applications of the models (H_S) and (M_S).

4.5.1 Eigenvalues and Approximation of Tensors

Similar as the eigenvalues of matrices, this kind of concept has been extended to higher order tensors (see e.g., [98, 99, 100, 91]). In fact, the concept for eigenvalues of tensors become richer than that restricting to matrices. Qi [98] proposed several definitions of tensor eigenvalues, among which the most popular and straightforward one is named *Z-eigenvalues*. For a given d-th order super-symmetric tensor $\boldsymbol{F} \in \mathbb{R}^{n^d}$, its Z-eigenvalue $\lambda \in \mathbb{R}$ with its corresponding eigenvector $\boldsymbol{x} \in \mathbb{R}^n$ are defined to be the solutions of the following system:

$$\begin{cases} F(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \ldots, \boldsymbol{x}}_{d-1}, \cdot) = \lambda \boldsymbol{x}, \\ \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} = 1. \end{cases}$$

Notice that the Z-eigenvalues are the usual eigenvalues for a symmetric matrix when the order of the tensor is 2. It was proven in Qi [98] that Z-eigenvalues exist for an even order real super-symmetric tensor $\boldsymbol{F}$, and $\boldsymbol{F}$ is positive definite if and only if all of its Z-eigenvalues are positive, which is similar to the matrix case. Thus, the smallest Z-eigenvalue of an even order super-symmetric tensor $\boldsymbol{F}$ is an important indicator of positive definiteness for $\boldsymbol{F}$. Conversely, the largest Z-eigenvalue can be an indicator of the negative definiteness for $\boldsymbol{F}$, which is exactly the model (H_S). In general, the optimal value and any optimal solution of (H_S) is the largest Z-eigenvalue and its corresponding eigenvector for the tensor $\boldsymbol{F}$, no matter d is even or odd. By Theorem 4.2.2, the largest Z-eigenvalue of an odd order super-symmetric tensor $\boldsymbol{F}$ can be approximated with a factor of $d!d^{-d}n^{-\frac{d-2}{2}}$. For an even order tensor, this approximation ratio is in relative

sense. However if we know in advance that the given even order tensor is positive semidefinite, we can also have an approximation factor of $d!d^{-d}n^{-\frac{d-2}{2}}$ for its largest Z-eigenvalue.

Regarding to the tensor approximation, in Section 3.4.2 we have discussed the best rank-one decomposition of a tensor. In case that the give tensor $\boldsymbol{F} \in \mathbb{R}^{n^d}$ is super-symmetric, then the corresponding best rank-one approximation should be

$$\begin{array}{ll} \min & \left\| \boldsymbol{F} - \underbrace{\boldsymbol{x} \otimes \boldsymbol{x} \otimes \cdots \otimes \boldsymbol{x}}_{d} \right\| \\ \text{s.t.} & \boldsymbol{x} \in \mathbb{R}^n. \end{array}$$

Applying the same technique discussed in Section 3.4.2, we can equivalently reformulate the above problem as

$$\begin{array}{ll} \max & F(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x}}_{d}) \\ \text{s.t.} & \boldsymbol{x} \in \mathbb{S}^n, \end{array}$$

which is identical to the largest eigenvalue problem and (H_S). In fact, when d is odd, if we denote its optimal solution (largest Z-eigenvector) to be $\hat{\boldsymbol{x}}$ and optimal value (largest Z-eigenvalue) to be $\lambda = F(\underbrace{\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}, \cdots, \hat{\boldsymbol{x}}}_{d})$, then the best rank-one approximation of the super-symmetric tensor $\boldsymbol{F}$ is $\lambda \underbrace{\hat{\boldsymbol{x}} \otimes \hat{\boldsymbol{x}} \otimes \cdots \otimes \hat{\boldsymbol{x}}}_{d}$.

4.5.2 Density Approximation in Quantum Physics

An interesting problem in physics is to give a precise characterization of entanglement in a quantum system. This describes types of correlations between subsystems of the full quantum system that go beyond the statistical correlations that can be found in a classical composite system. Specifically it arises a matrix approximation problem. The following formulation is proposed in Dahl et al. [27].

Denote Δ_+^n to be the set of all $n \times n$ positive semedefinite matrices with trace being 1, i.e., $\Delta_+^n := \{\boldsymbol{A} \in \mathbb{R}^{n \times n} \mid \boldsymbol{A} \succeq 0, \operatorname{tr}(\boldsymbol{A}) = 1\}$ (sometimes it is also called the matrix simplex). Using the matrix decomposition method (see e.g., Sturm and Zhang [113]), it is not hard to verify that the extreme points of Δ_+^n are all rank-one matrices, or specifically, $\Delta_+^n = \operatorname{conv}\{\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}} \mid \boldsymbol{x} \in \mathbb{S}^n\}$. If $n = n_1 n_2$, where n_1 and n_2 are given two positive integers, then we call a matrix $\boldsymbol{A} \in \Delta_+^n$ *separable* if $\boldsymbol{A}$ can be written as a convex combination

$$\boldsymbol{A} = \sum_{i=1}^{m} \lambda_i \boldsymbol{B}_i \otimes \boldsymbol{C}_i$$

for some positive integer m, matrices $\boldsymbol{B}_i \in \Delta_+^{n_1}$ and $\boldsymbol{C}_i \in \Delta_+^{n_2}$ for $i = 1, 2, \ldots, m$, and nonnegative scalars $\lambda_i\,(i = 1, 2, \ldots, m)$ with $\sum_{i=1}^m \lambda_i = 1$. For given n_1 and n_2, denote $\Delta_+^{n,\otimes}$ to be the set of all separable matrices of order $n = n_1 n_2$. The *density approximation* problem is the following. Given a density matrix $\boldsymbol{A} \in \Delta_+^n$, find a separable density matrix $\boldsymbol{X} \in \Delta_+^{n,\otimes}$ which is closest to $\boldsymbol{A}$, or specifically, the minimization model

$$\begin{array}{ll} (DA) & \min \quad \|\boldsymbol{X} - \boldsymbol{A}\| \\ & \text{s.t.} \quad \boldsymbol{X} \in \Delta_+^{n,\otimes}. \end{array}$$

This projection problem is in general NP-hard, mainly relying on the understanding of $\Delta_+^{n,\otimes}$. An important property of $\Delta_+^{n,\otimes}$ is that all its extreme points are symmetric rank-one matrices $(\boldsymbol{x} \otimes \boldsymbol{y})(\boldsymbol{x} \otimes \boldsymbol{y})^{\mathrm{T}}$ with $\boldsymbol{x} \in \mathbb{S}^{n_1}$ and $\boldsymbol{y} \in \mathbb{S}^{n_2}$ (see the proof in Theorem 2.2 of [27]), i.e.,

$$\Delta_+^{n,\otimes} = \mathrm{conv}\,\{(\boldsymbol{x} \otimes \boldsymbol{y})(\boldsymbol{x} \otimes \boldsymbol{y})^{\mathrm{T}} \mid \boldsymbol{x} \in \mathbb{S}^{n_1},\ \boldsymbol{y} \in \mathbb{S}^{n_2}\}.$$

Then in stead, we may turn to the projection subproblem of (DA), to find the projection of $\boldsymbol{A}$ on the extreme points of $\Delta_+^{n,\otimes}$, which is

$$\begin{array}{ll} \min & \|(\boldsymbol{x} \otimes \boldsymbol{y})(\boldsymbol{x} \otimes \boldsymbol{y})^{\mathrm{T}} - \boldsymbol{A}\| \\ \text{s.t.} & \boldsymbol{x} \in \mathbb{S}^{n_1},\ \boldsymbol{y} \in \mathbb{S}^{n_2}. \end{array}$$

Straightforward computation shows that

$$\|(\boldsymbol{x} \otimes \boldsymbol{y})(\boldsymbol{x} \otimes \boldsymbol{y})^{\mathrm{T}} - \boldsymbol{A}\|^2 = 1 - 2\,\boldsymbol{A} \bullet (\boldsymbol{x} \otimes \boldsymbol{y})(\boldsymbol{x} \otimes \boldsymbol{y})^{\mathrm{T}} + \|\boldsymbol{A}\|^2.$$

Therefore the projection subproblem is equivalent to

$$\begin{array}{ll} \max & A \bullet (\boldsymbol{x} \otimes \boldsymbol{y})(\boldsymbol{x} \otimes \boldsymbol{y})^{\mathrm{T}} \\ \text{s.t.} & \boldsymbol{x} \in \mathbb{S}^{n_1},\ \boldsymbol{y} \in \mathbb{S}^{n_2}, \end{array}$$

which is the exact model (M_S) with $d = 4$ and $d_1 = d_2 = 2$.

4.6 Numerical Experiments

In this section we are going to present the numerical performance of the approximation algorithms proposed in this chapter. In particular, the model (H_Q) with $d = 4$ is being tested, i.e.,

$$\begin{array}{lll} (EH_Q) & \max & f(\boldsymbol{x}) = \sum_{1 \le i,j,k,\ell \le n} F_{ijk\ell}\, x_i x_j x_k x_\ell \\ & \text{s.t.} & \boldsymbol{x}^{\mathrm{T}} \boldsymbol{Q}_i \boldsymbol{x} \le 1,\ i = 1, 2, \ldots, m, \\ & & \boldsymbol{x} \in \mathbb{R}^n, \end{array}$$

Table 4.1: Numerical results of (EH_Q) for $n = 10$ and $m = 30$

Instance	1	2	3	4	5	6	7	8	9	10
$100 \cdot v$	0.65	0.77	0.32	0.27	0.73	0.42	0.52	0.64	0.98	1.04
$100 \cdot \bar{v}$	4.96	4.53	4.75	5.05	5.86	5.32	5.00	5.19	5.07	5.92
τ (%)	13.10	17.00	6.74	5.35	12.46	7.89	10.40	12.33	19.33	17.57
$n \ln^3 m \cdot \tau$	51.56	66.88	26.51	21.04	49.01	31.06	40.92	48.52	76.05	69.12

where fourth order tensor $\boldsymbol{F}$ is super-symmetric, and matrix $\boldsymbol{Q}_i$ is positive semidefinite for $i = 1, 2, \ldots, m$. During the testings, cvx v1.2 (Grant and Boyd [41]) is called for solving the SDP problems whenever applicable.

4.6.1 Randomly Simulated Data

For the data of (EH_Q), a fourth order tensor $\boldsymbol{F}'$ is randomly generated, whose n^4 entries follow i.i.d. standard normals. We then symmetrize $\boldsymbol{F}'$ to form a super-symmetric tensor $\boldsymbol{F}$ by averaging the related entries. As to the constraints, we generate m matrix $\boldsymbol{Q}'_i \in \mathbb{R}^{n\times n}$ $(i = 1, 2, \ldots, m)$ independently, whose entries also follow i.i.d. standard normals, and then let $\boldsymbol{Q}_i = (\boldsymbol{Q}'_i)^{\mathrm{T}}\boldsymbol{Q}'_i$ for $i = 1, 2, \ldots, m$.

For the particular nature of (EH_Q), rather than directly applying Algorithm 4.3.1 to solve it, we use a simplified method. First (EH_Q) is relaxed to

$$\begin{array}{ll} \max & F(\boldsymbol{X}, \boldsymbol{X}) = \sum_{1\le i,j,k,\ell\le n} F_{ijk\ell} X_{ij} X_{k\ell} \\ \text{s.t.} & \mathrm{tr}\,(\boldsymbol{Q}_i \boldsymbol{X} \boldsymbol{Q}_j \boldsymbol{X}^{\mathrm{T}}) \le 1,\ i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, m, \\ & \boldsymbol{X} \in \mathbb{R}^{n\times n}, \end{array}$$

which is a standard quadratic program, and can be solved approximately by SDP relaxation and randomization (see e.g., [75] or Section 2.5). The optimal value of the SDP relaxation problem is denoted by $\bar{v}$, which we shall use as an upper bound of $v(EH_Q)$. We then apply DR 3.3.1 to decompose this approximate solution into $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}} \in \mathbb{R}^n$. Finally we pick a vector with the best objective value of $f(\boldsymbol{x})$ from $\{\boldsymbol{0}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, (\hat{\boldsymbol{x}} + \hat{\boldsymbol{y}})/2, (\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}})/2\}$ as the output. This objective value is denoted by v, and a ratio $\tau := v/\bar{v}$ is also computed.

By following essentially the same proof, this simplified method also enjoys a worst-case relative performance ratio of $\Omega\left(\frac{1}{n \log^3 m}\right)$, similar as Theorem 4.3.2 asserted. For $n = 10$ and $m = 30$, we randomly generate 10 instances of (EH_Q). The solution results

Table 4.2: Numerical ratios (average of 10 instances) of (EH_Q)

n	2	5	8	10	12
τ (%) for $m = 1$	90.2	57.9	73.3	66.2	60.0
τ (%) for $m = 5$	65.6	28.3	22.5	29.1	17.1
τ (%) for $m = 10$	60.4	22.3	14.6	16.0	8.9
τ (%) for $m = 30$	59.4	17.8	10.2	12.2	9.2

are shown in Table 4.1. In Table 4.2, the absolute approximation ratios for various n and m are shown. Remark that the dimensions of the problems that can be efficiently solved using our algorithms are not large, due to the limitation of solving large size SDP relaxation problems.

4.6.2 Comparison with Sum of Squares Method

In this subsection, we compare our solution method with the so-called sum of squares (SOS) method [70, 71] for solving (EH_Q). Due to the limitations of the current SDP solvers, our method works only for small size problems. Since the SOS approach works quite efficiently for small size polynomial optimization problems, it is interesting to know how the SOS method would perform in solving these randomly generated instances of (EH_Q). In particular, we shall use GloptiPoly 3 of Henrion et al. [53].

We randomly generated 10 instances of (EH_Q). By using the first SDP relaxation (Lasserre's procedure [70]), GloptiPoly 3 found global optimal solutions for 4 instances, and got upper bounds of optimal values for the other 6 instances. In the latter case, however, no feasible solutions are generated, while our algorithm always finds feasible solutions with guaranteed approximation ratio, and so the two approaches are complementary to each other. Moreover, GloptiPoly 3 always yields a better upper bound than $\bar{v}$ for our test instances, which helps to yield better approximation ratios. The average ratio is 0.112 by the using upper bound $\bar{v}$, and is 0.262 by using the upper bound produced by GloptiPoly 3 (see Table 4.3).

To conclude this section as well as this chapter, we remark that the algorithms proposed are actually practical, and they produce high quality solutions. The worst-case performance analysis offers a theoretical 'safety net', which is usually far from the *typical* performance. Moreover, it is of course possible to improve the solution by

Table 4.3: Numerical results of (EH_Q) compared with SOS for $n = 12$ and $m = 30$

Instance	1	2	3	4	5	6	7	8	9	10
$100 \cdot v$	0.30	0.76	0.43	0.76	0.70	0.49	0.81	0.34	0.29	0.62
$100 \cdot \bar{v}$	4.75	4.47	5.21	5.20	4.59	4.81	5.23	5.12	5.89	4.78
$100 \cdot \bar{v}_{SOS}$	2.05	2.02	2.43	2.41	1.86	2.02	1.99	2.24	2.83	1.88
Optimality of $\bar{v}_{SOS}$	No	No	Yes	Yes	No	Yes	No	Yes	No	No
$v/\bar{v}\,(\%)$	6.32	17.00	8.25	14.62	15.25	10.19	15.49	6.64	4.92	12.97
$v/\bar{v}_{SOS}\,(\%)$	14.63	37.62	17.70	31.54	37.63	24.26	40.70	15.18	10.25	32.98

some local search procedure, e.g., the projection gradient methods [22], maximum block improvement method [25].

Chapter 5

Polynomial Optimization with Convex Constraints

5.1 Introduction

This chapter tackles an important and useful extension of the models studied in previous chapters: to allow the objective function to be a generic inhomogeneous polynomial function. As is evident, many important applications of polynomial optimizations involve an objective that is intrinsically inhomogeneous. Specifically, we consider the following problems:

$$\boxed{\begin{array}{lll}(P_{\bar{S}}) & \max & p(\boldsymbol{x}) \\ & \text{s.t.} & \boldsymbol{x} \in \bar{\mathbb{S}}^n;\end{array}}$$

$$\boxed{\begin{array}{lll}(P_Q) & \max & p(\boldsymbol{x}) \\ & \text{s.t.} & \boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x} \le 1,\ i = 1, 2, \ldots, m, \\ & & \boldsymbol{x} \in \mathbb{R}^n\end{array}}$$

where $\boldsymbol{Q}_i \succeq 0$ for $k = 1, 2, \ldots, d$, and $\sum_{i=1}^m \boldsymbol{Q}_i \succ 0$. It is obvious that (P_Q) is an extension of $(P_{\bar{S}})$. We also in the chapter consider a much more general frame of polynomial optimization over a general convex compact set, i.e., for a give convex compact set $G \subset \mathbb{R}^n$, the problem

$$\boxed{\begin{array}{lll}(P_G) & \max & p(\boldsymbol{x}) \\ & \text{s.t.} & \boldsymbol{x} \in G.\end{array}}$$

The model (P_S) can be solved in polynomial-time when $d \leq 2$, and becomes NP-hard when $d \geq 3$. Even worse for $d \geq 3$, there is no polynomial-time approximation algorithm with a positive approximation ratio unless $P = NP$, which we shall argue later. Therefore, the whole chapter is focus on *relative* approximation algorithms. The inapproximability of (P_S) differs greatly to that of the homogeneous model (H_S) discussed in Section 4.2, since when d is odd, (H_S) admits a polynomial-time approximation algorithm with a positive approximation ratio by Theorem 4.2.2. Consequently, the optimization of an inhomogeneous polynomial is much harder than a homogeneous one. The complexity of (P_Q) and (P_G) is similar, being solvable in polynomial-time only when $d = 1$ and NP-hard when $d \geq 2$. This is because (P_G) is generalized from (P_Q), and (P_Q) is generalized from (H_Q), an NP-hard problem when $d \geq 2$ (see the discussion in Section 4.1).

Extending the solution methods and the corresponding analysis from *homogeneous* polynomial optimizations to the general *inhomogeneous* polynomials is not straightforward. As a matter of fact, so far all the successful approximation algorithms with provable approximation ratios in the literature, e.g., the quadratic models considered in [88, 87, 120, 75, 50] and the quartic models considered in [73, 77], are dependent on the homogeneity in a crucial way. Technically, a homogenous polynomial allows one to *scale* the overall function value along a given direction, which is an essential operation in proving the quality bound of the approximation algorithms. The current chapter breaks its path from the preceding practices, by directly dealing with a *homogenizing* variable. Although homogenization is a natural way to deal with inhomogeneous polynomial functions, it is quite a different matter when it comes to the worst-case performance ratio analysis. In fact, the usual homogenization does not lead to any assured performance ratio. In this chapter we shall point out a specific route to get around this difficulty, in which we actually provide a general scheme to approximately solve such problems via homogenization.

In Section 5.2, we start by analyzing the model where the constraint set is the Euclidean ball, i.e., the model (P_S). We propose polynomial-time approximation algorithms with guaranteed relative approximation ratios, which serve as a basis for the subsequent analysis. In Section 5.3, the discussion is extended to cover the problem where the constraint set is the intersection of a finite number of co-centered ellipsoids, i.e., the model (P_Q), and relative approximation algorithms are proposed as well. In

Section 5.4, the approximation bounds are derived even for some very general optimization models (P_G), e.g., optimization of a polynomial over a polytope. It turns out that for such general problems, it is still possible to derive relative approximation ratios, which depend on the problem dimensions only. The tool we used is the *Löwner-John ellipsoids*. In Section 5.5, we discuss some applications with the models presented in this chapter. Finally, we report our numerical experiment results in Section 5.6. As this chapter is concerned with the relative approximation ratios, we may without loss of generality assume the polynomial function $p(\boldsymbol{x})$ to have no constant term, i.e., $p(\mathbf{0}) = 0$.

5.2 Polynomial with Ball Constraint

Our first model in this chapter is to maximize a generic multivariate polynomial function subject to the Euclidean ball constraint, i.e.,

$$\boxed{\begin{array}{lll}(P_S) & \max & p(\boldsymbol{x}) \\ & \text{s.t.} & \boldsymbol{x} \in \mathbb{S}^n.\end{array}}$$

Since we assume $p(\boldsymbol{x})$ to have no constant term, the optimal value of this problem is obviously nonnegative, i.e., $v(P_S) \geq 0$.

The complexity to solve (P_S) can be summarized by the following proposition.

Proposition 5.2.1 *If $d \leq 2$, then (P_S) can be solved in polynomial-time; Otherwise if $d \geq 3$, then (P_S) is NP-hard, and there is no polynomial-time approximation algorithm with a positive approximation ratio unless $P = NP$.*

Proof. For $d \leq 2$, (P_S) is a standard trust region subproblem. As such it is well known to be solvable in polynomial-time (see e.g., [113, 114] and the references therein). For $d \geq 3$, in a special case where $p(\boldsymbol{x})$ is a homogeneous cubic form, it is easy to see that (P_S) is equivalent to $\max_{\boldsymbol{x} \in \mathbb{S}^n} p(\boldsymbol{x})$, which is shown to be NP-hard by Nesterov [90].

Let us now consider a special class of (P_S) when $d = 3$:

$$\begin{array}{rcll} v(\alpha) & = & \max & f(\boldsymbol{x}) - \alpha \|\boldsymbol{x}\|^2 \\ & & \text{s.t.} & \boldsymbol{x} \in \mathbb{S}^n, \end{array}$$

where $\alpha \geq 0$, and $f(\boldsymbol{x})$ is a homogeneous cubic form associated with a nonzero supersymmetric tensor $\boldsymbol{\mathcal{F}} \in \mathbb{R}^{n \times n \times n}$. If $v(\alpha) > 0$, then its optimal solution $\boldsymbol{x}^*$ satisfies

$$f(\boldsymbol{x}^*) - \alpha \|\boldsymbol{x}^*\|^2 = \|\boldsymbol{x}^*\|^3 f\left(\frac{\boldsymbol{x}^*}{\|\boldsymbol{x}^*\|}\right) - \alpha \|\boldsymbol{x}^*\|^2 = \|\boldsymbol{x}^*\|^2 \left(\|\boldsymbol{x}^*\| f\left(\frac{\boldsymbol{x}^*}{\|\boldsymbol{x}^*\|}\right) - \alpha\right) > 0.$$

Thus by the optimality of $\boldsymbol{x}^*$, we have $\|\boldsymbol{x}^*\| = 1$. If we choose $\alpha = \|\boldsymbol{F}\| \geq \max_{\boldsymbol{x}\in\mathbb{S}^n} f(\boldsymbol{x})$, then $v(\alpha) = 0$. Since otherwise we must have $v(\alpha) > 0$ and $\|\boldsymbol{x}^*\| = 1$, with

$$v(\alpha) = f(\boldsymbol{x}^*) - \alpha\|\boldsymbol{x}^*\|^2 \leq \max_{\boldsymbol{x}\in\mathbb{S}^n} f(\boldsymbol{x}) - \alpha \leq 0,$$

which is a contradiction. Moreover, $v(0) > 0$ simply because $\boldsymbol{F}$ is a nonzero tensor, and it is also easy to see that $v(\alpha)$ is non-increasing as $\alpha \geq 0$ increases. Hence, there is a threshold $\alpha_0 \in [0, \|\boldsymbol{F}\|]$, such that $v(\alpha) > 0$ if $0 \leq \alpha < \alpha_0$, and $v(\alpha) = 0$ if $\alpha \geq \alpha_0$.

Suppose there exists a polynomial-time approximation algorithm with a positive approximation ratio τ for (P_S) when $d \geq 3$. Then for every $\alpha \geq 0$, we can find $\boldsymbol{z} \in \mathbb{S}^n$ in polynomial-time, such that $g(\alpha) := f(\boldsymbol{z}) - \alpha\|\boldsymbol{z}\|^2 \geq \tau v(\alpha)$. It is obvious that $g(\alpha) \geq 0$ since $v(\alpha) \geq 0$. Together with the fact that $g(\alpha) \leq v(\alpha)$ we have that $g(\alpha) > 0$ if and only if $v(\alpha) > 0$, and $g(\alpha) = 0$ if and only if $v(\alpha) = 0$. Therefore, the threshold α_0 also satisfies $g(\alpha) > 0$ if $0 \leq \alpha < \alpha_0$, and $g(\alpha) = 0$ if $\alpha \geq \alpha_0$. By applying the bisection search over the interval $[0, \|\boldsymbol{F}\|]$ with this polynomial-time approximation algorithm, we can find α_0 and $\boldsymbol{z} \in \mathbb{S}^n$ in polynomial-time, such that $f(\boldsymbol{z}) - \alpha_0\|\boldsymbol{z}\|^2 = 0$. This implies that $\boldsymbol{z} \in \mathbb{S}^n$ is the optimal solution for the problem $\max_{\boldsymbol{x}\in\mathbb{S}^n} f(\boldsymbol{x})$ with the optimal value α_0, which is an NP-hard problem mentioned in the beginning of the proof. Therefore, such approximation algorithm cannot exist unless $P = NP$. □

The negative result in Proposition 5.2.1 rules out any polynomial-time approximation algorithm with a positive approximation ratio for (P_S). However a positive *relative* approximation ratio is still possible, which is the main subject of this section. Below we shall first present a polynomial-time algorithm for approximately solving (P_S), which admits a (relative) worst-case performance ratio. In fact, here we present a general scheme aiming at solving the polynomial optimization (P_S). This scheme breaks down to the following four major steps:

1. Introduce an equivalent model with the objective being a homogenous form;
2. Solve a relaxed model with the objective being a multilinear form;
3. Adjust to get a solution based on the solution of the relaxed model;
4. Assemble a solution for the original inhomogeneous model.

Some of these steps can be designed separately. The algorithm below is one realization of the general scheme for solving (P_S), with each step being carried out by a specific

procedure. We first present the specialized algorithm, and then in the remainder of the section, we elaborate on these four general steps, and prove that in combination they lead to a polynomial-time approximation algorithm with a quality-assured solution.

Algorithm 5.2.1

- *INPUT: an n-dimensional d-th degree polynomial function $p(\boldsymbol{x})$.*

1 *Rewrite $p(\boldsymbol{x}) - p(\mathbf{0}) = F(\underbrace{\bar{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \cdots, \bar{\boldsymbol{x}}}_{d})$ when $x_h = 1$ as in (5.2), with $\boldsymbol{F}$ being an $(n+1)$-dimensional d-th order super-symmetric tensor.*

2 *Apply Algorithm 3.2.3 to solve the problem*

$$\begin{array}{ll} \max & F(\bar{\boldsymbol{x}}^1, \bar{\boldsymbol{x}}^2, \cdots, \bar{\boldsymbol{x}}^d) \\ \text{s.t.} & \bar{\boldsymbol{x}}^k \in \mathbb{S}^{n+1},\ k = 1, 2, \ldots, d \end{array}$$

approximately, with input $\boldsymbol{F}$ and output $(\bar{\boldsymbol{y}}^1, \bar{\boldsymbol{y}}^2, \cdots, \bar{\boldsymbol{y}}^d)$.

3 *Compute $(\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \cdots, \bar{\boldsymbol{z}}^d) = \arg\max \left\{ F\left(\left(\bar{\boldsymbol{y}}_1^{1/d}\right), \left(\bar{\boldsymbol{y}}_1^{2/d}\right), \cdots, \left(\bar{\boldsymbol{y}}_1^{d/d}\right) \right), \boldsymbol{\xi} \in \mathbb{B}^d \right\}$.*

4 *Compute $\boldsymbol{z} = \arg\max \left\{ p(\mathbf{0}) : p\left(\boldsymbol{z}(\beta)/z_h(\beta)\right), \boldsymbol{\beta} \in \mathbb{B}^d \text{ and } \beta_1 = \prod_{k=2}^{d} \beta_k = 1 \right\}$, with $\bar{\boldsymbol{z}}(\beta) = \beta_1(d+1)\bar{\boldsymbol{z}}^1 + \sum_{k=2}^{d} \beta_k \bar{\boldsymbol{z}}^k$.*

- *OUTPUT: a feasible solution $\boldsymbol{z} \in \mathbb{S}^n$.*

In Step 2 of Algorithm 5.2.1, Algorithm 3.2.3 is called to approximately solve the spherically constrained multilinear form optimization problem, which is a deterministic polynomial-time algorithm. Notice the degree of the polynomial $p(\boldsymbol{x})$ is deemed a fixed parameter in this thesis, and thus Algorithm 5.2.1 runs in polynomial-time, and is deterministic too. Our main result in this section is the following:

Theorem 5.2.2 *(P_S) admits a polynomial-time approximation algorithm with relative approximation ratio $\tau(P_S)$, where*

$$\tau(P_S) := 2^{-\frac{5d}{2}}(d+1)!\, d^{-2d}(n+1)^{-\frac{d-2}{2}} = \Omega\left(n^{-\frac{d-2}{2}}\right).$$

Although homogenization is a natural way to deal with inhomogeneous polynomials, the worst-case performance ratio does not follow straightforwardly. What is lacking is that an inhomogeneous polynomial does not allow one to scale the overall function value along a given direction, which is however an essential operation to prove the quality bound of the approximation algorithms (see e.g., [87, 75, 50, 77]). Below we study in detail how a particular implementation of these four steps of the scheme (which becomes Algorithm 5.2.1) leads to the promised worst-case relative performance ratio in Theorem 5.2.2. As we shall see later, our solution scheme can be applied to solve a very general polynomial optimization model (P_G).

5.2.1 Homogenization

The method of homogenization depends on the form of the polynomial $p(\boldsymbol{x})$. Here in discussion we assume $p(\boldsymbol{x})$ to have no constant term, although Algorithm 5.2.1 applies for any polynomial. If $p(\boldsymbol{x})$ is given as a summation of homogeneous polynomial functions of different degrees, i.e., $f_k(\boldsymbol{x})\,(1 \le k \le d)$ is a homogeneous polynomial function of degree k, then we may first write

$$f_k(\boldsymbol{x}) = F_k(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x}}_{k}) \tag{5.1}$$

with $\boldsymbol{F}_k$ being a k-th order super-symmetric tensor. Then by introducing a homogenizing variable x_h, which is always equal to 1, we may rewrite $p(\boldsymbol{x})$ as

$$\begin{aligned} p(\boldsymbol{x}) &= \sum_{k=1}^{d} f_k(\boldsymbol{x}) = \sum_{k=1}^{d} f_k(\boldsymbol{x}) {x_h}^{d-k} = \sum_{k=1}^{d} F_k(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x}}_{k}) {x_h}^{d-k} \\ &= F\Big(\underbrace{\begin{pmatrix}\boldsymbol{x}\\ x_h\end{pmatrix}, \begin{pmatrix}\boldsymbol{x}\\ x_h\end{pmatrix}, \cdots, \begin{pmatrix}\boldsymbol{x}\\ x_h\end{pmatrix}}_{d}\Big) = F(\underbrace{\bar{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \cdots, \bar{\boldsymbol{x}}}_{d}) = f(\bar{\boldsymbol{x}}). \end{aligned} \tag{5.2}$$

where $\boldsymbol{F}$ is an $(n+1)$-dimensional d-th order super-symmetric tensor, whose last component is 0 (since $p(\boldsymbol{x})$ has no constant term).

If the polynomial $p(\boldsymbol{x})$ is given in terms of summation of monomials, we should first group them according to their degrees, and then rewrite the summation of monomials in each group as homogeneous polynomial function. After that, we then proceed according to (5.1) and (5.2) to obtain the tensor form $\boldsymbol{F}$, as required.

Finally in this step, we may equivalently reformulate (P_S) as

$$\begin{array}{lll}(\bar{P}_S) & \max & f(\bar{\boldsymbol{x}}) \\ & \text{s.t.} & \bar{\boldsymbol{x}} = \begin{pmatrix} \boldsymbol{x} \\ x_h \end{pmatrix}, \\ & & \boldsymbol{x} \in \bar{\mathbb{S}}^n,\ x_h = 1.\end{array}$$

Obviously, we have $v(P_S) = v(\bar{P}_S)$ and $\underline{v}(P_S) = \underline{v}(\bar{P}_S)$.

5.2.2 Multilinear Form Relaxation

Multilinear form relaxation has proven to be effective, as discussed in Chapter 4. Specifically, Lemma 4.2.1 and Lemma 4.4.3 are the key link formulae. Now we relax $(\bar{P}_S)$ to an inhomogeneous multilinear form optimization problem as:

$$\begin{array}{lll}(TP_S) & \max & F(\bar{\boldsymbol{x}}^1, \bar{\boldsymbol{x}}^2, \cdots, \bar{\boldsymbol{x}}^d) \\ & \text{s.t.} & \bar{\boldsymbol{x}}^k = \begin{pmatrix} \boldsymbol{x}^k \\ x_h^k \end{pmatrix},\ k = 1, 2, \ldots, d, \\ & & \boldsymbol{x}^k \in \bar{\mathbb{S}}^n,\ x_h^k = 1,\ k = 1, 2, \ldots, d.\end{array}$$

Obviously, we have $v(TP_S) \geq v(\bar{P}_S) = v(P_S)$. Before proceeding, let us first settle the computational complexity issue for solving (TP_S).

Proposition 5.2.3 *(TP_S) is NP-hard whenever $d \geq 3$.*

Proof. Notice that in Proposition 3.2.2, we proved the following problem is NP-hard:

$$\begin{array}{ll}\max & F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \\ \text{s.t.} & \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{S}^n.\end{array}$$

If $d = 3$ and a special case where $\boldsymbol{F}$ has the form $F_{n+1,j,k} = F_{i,n+1,k} = F_{i,j,n+1} = 0$ for all $1 \leq i, j, k \leq n+1$, (TP_S) is equivalent to the above model, and thus is NP-hard. □

(TP_S) is still difficult to solve, and moreover it remains inhomogeneous, since x_h^k is required to be 1. To our best knowledge, no polynomial-time approximation algorithm is available in the literature to solve this problem. Furthermore, we shall relax the constraint $x_h^k = 1$, and introduce the following parameterized and homogenized problem:

$$\begin{array}{lll}(TP_S(t)) & \max & F(\bar{\boldsymbol{x}}^1, \bar{\boldsymbol{x}}^2, \cdots, \bar{\boldsymbol{x}}^d) \\ & \text{s.t.} & \|\bar{\boldsymbol{x}}^k\| \leq t,\ \bar{\boldsymbol{x}}^k \in \mathbb{R}^{n+1},\ k = 1, 2, \ldots, d.\end{array}$$

Obviously, (TP_S) can be relaxed to $(TP_S(\sqrt{2}))$, since if $\bar{\boldsymbol{x}}$ is feasible for (TP_S) then $\|\bar{\boldsymbol{x}}\|^2 = \|\boldsymbol{x}\|^2 + {x_h}^2 \leq 1 + 1 = 2$. Consequently, $v(TP_S(\sqrt{2})) \geq v(TP_S)$.

Both the objective and the constraints are now homogeneous, and it is easy to see for all $t > 0$, $(TP_S(t))$ is equivalent to each other by a simple scaling method. Moreover, $(TP_S(1))$ is equivalent to

$$\begin{array}{ll} \max & F(\bar{\boldsymbol{x}}^1, \bar{\boldsymbol{x}}^2, \cdots, \bar{\boldsymbol{x}}^d) \\ \text{s.t.} & \bar{\boldsymbol{x}}^k \in \mathbb{S}^{n+1},\ k = 1, 2, \ldots, d, \end{array}$$

which is in the form of (T_S) discussed in Section 3.2. By using Algorithm 3.2.3 and applying Theorem 3.2.4, $(TP_S(1))$ admits a polynomial-time approximation algorithm with approximation ratio $(n+1)^{-\frac{d-2}{2}}$. Therefore, for all $t > 0$, $(TP_S(t))$ also admits a polynomial-time approximation algorithm with approximation ratio $(n+1)^{-\frac{d-2}{2}}$, and $v(TP_S(t)) = t^d\, v(TP_S(1))$. After this relaxation step (Step 2 in Algorithm 5.2.1), we are able to find a feasible solution $(\bar{\boldsymbol{y}}^1, \bar{\boldsymbol{y}}^2, \cdots, \bar{\boldsymbol{y}}^d)$ of $(TP_S(1))$ in polynomial-time, such that

$$\begin{array}{rcl} F(\bar{\boldsymbol{y}}^1, \bar{\boldsymbol{y}}^2, \cdots, \bar{\boldsymbol{y}}^d) & \geq & (n+1)^{-\frac{d-2}{2}}\, v(TP_S(1)) \\ & = & 2^{-\frac{d}{2}}(n+1)^{-\frac{d-2}{2}}\, v(TP_S(\sqrt{2})) \\ & \geq & 2^{-\frac{d}{2}}(n+1)^{-\frac{d-2}{2}}\, v(TP_S). \end{array} \tag{5.3}$$

Algorithm 3.2.3 is the engine which enables the second step of our scheme. In fact, any polynomial-time approximation algorithm of $(TP_S(1))$ can be used as an engine to yield a realization (algorithm) of our scheme. As will become evident later, any improvement of the approximation ratio of $(TP_S(1))$ leads to the improvement of relative approximation ratio in Theorem 5.2.2. For example, recently So [108] improved the approximation bound of $(TP_S(1))$ to $\Omega\left(\left(\frac{\log n}{n}\right)^{-\frac{d-2}{2}}\right)$ (though the algorithm is mainly of theoretical interest), and consequently the relative approximation ratio under our scheme is improved to $\Omega\left(\left(\frac{\log n}{n}\right)^{\frac{d-2}{2}}\right)$ too. Of course, one may apply any other favorite algorithm to solve the relaxation $(TP_S(1))$. For instance, the alternating least square (ALS) algorithm (see e.g., [68] and the references therein), and the maximum block improvement (MBI) method of Chen et al. [25], can be other alternatives for the second step.

5.2.3 Homogenizing Components Adjustment

The approximate solution $(\bar{\boldsymbol{y}}^1, \bar{\boldsymbol{y}}^2, \cdots, \bar{\boldsymbol{y}}^d)$ of $(TP_S(1))$ satisfies $\|\bar{\boldsymbol{y}}^k\| \leq 1$ for all $1 \leq k \leq d$, which implies $\|\boldsymbol{y}^k\| \leq 1$, but in general we do not have any control on the size of

y_h^k, and thus $(\bar{\boldsymbol{y}}^1, \bar{\boldsymbol{y}}^2, \cdots, \bar{\boldsymbol{y}}^d)$ may not be a feasible solution for (TP_S). The following lemma plays a link role in our analysis, to ensure that the construction of a feasible solution for the inhomogeneous model (TP_S) is possible.

Lemma 5.2.4 *Suppose $\bar{\boldsymbol{x}}^k \in \mathbb{R}^{n+1}$ with $|x_h^k| \leq 1$ for all $1 \leq k \leq d$. Let $\eta_1, \eta_2, \cdots, \eta_d$ be independent random variables, each taking values 1 and -1 with $\mathrm{E}[\eta_k] = x_h^k$ for all $1 \leq k \leq d$, and let $\xi_1, \xi_2, \cdots, \xi_d$ be i.i.d. random variables, each taking values 1 and -1 with equal probability $1/2$. If the last component of the tensor $\boldsymbol{F}$ is 0, then*

$$\mathrm{E}\left[\prod_{k=1}^{d} \eta_k F\left(\begin{pmatrix}\eta_1 \boldsymbol{x}^1 \\ 1\end{pmatrix}, \begin{pmatrix}\eta_2 \boldsymbol{x}^2 \\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\eta_d \boldsymbol{x}^d \\ 1\end{pmatrix}\right)\right] = F(\bar{\boldsymbol{x}}^1, \bar{\boldsymbol{x}}^2, \cdots, \bar{\boldsymbol{x}}^d), \tag{5.4}$$

and

$$\mathrm{E}\left[F\left(\begin{pmatrix}\xi_1 \boldsymbol{x}^1 \\ 1\end{pmatrix}, \begin{pmatrix}\xi_2 \boldsymbol{x}^2 \\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\xi_d \boldsymbol{x}^d \\ 1\end{pmatrix}\right)\right] = 0. \tag{5.5}$$

Proof. The claimed equations readily result from the following observations:

$$\begin{aligned}
& \mathrm{E}\left[\prod_{k=1}^{d} \eta_k F\left(\begin{pmatrix}\eta_1 \boldsymbol{x}^1 \\ 1\end{pmatrix}, \begin{pmatrix}\eta_2 \boldsymbol{x}^2 \\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\eta_d \boldsymbol{x}^d \\ 1\end{pmatrix}\right)\right] \\
= \; & \mathrm{E}\left[F\left(\begin{pmatrix}{\eta_1}^2 \boldsymbol{x}^1 \\ \eta_1\end{pmatrix}, \begin{pmatrix}{\eta_2}^2 \boldsymbol{x}^2 \\ \eta_2\end{pmatrix}, \cdots, \begin{pmatrix}{\eta_d}^2 \boldsymbol{x}^d \\ \eta_d\end{pmatrix}\right)\right] \quad \text{(multilinearity of } F\text{)} \\
= \; & F\left(\mathrm{E}\left[\begin{pmatrix}\boldsymbol{x}^1 \\ \eta_1\end{pmatrix}\right], \mathrm{E}\left[\begin{pmatrix}\boldsymbol{x}^2 \\ \eta_2\end{pmatrix}\right], \cdots, \mathrm{E}\left[\begin{pmatrix}\boldsymbol{x}^d \\ \eta_d\end{pmatrix}\right]\right) \quad \text{(independence of } \eta_k\text{'s)} \\
= \; & F(\bar{\boldsymbol{x}}^1, \bar{\boldsymbol{x}}^2, \cdots, \bar{\boldsymbol{x}}^d),
\end{aligned}$$

and

$$\begin{aligned}
& \mathrm{E}\left[F\left(\begin{pmatrix}\xi_1 \boldsymbol{x}^1 \\ 1\end{pmatrix}, \begin{pmatrix}\xi_2 \boldsymbol{x}^2 \\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\xi_d \boldsymbol{x}^d \\ 1\end{pmatrix}\right)\right] \\
= \; & F\left(\mathrm{E}\left[\begin{pmatrix}\xi_1 \boldsymbol{x}^1 \\ 1\end{pmatrix}\right], \mathrm{E}\left[\begin{pmatrix}\xi_2 \boldsymbol{x}^2 \\ 1\end{pmatrix}\right], \cdots, \mathrm{E}\left[\begin{pmatrix}\xi_d \boldsymbol{x}^d \\ 1\end{pmatrix}\right]\right) \quad \text{(independence of } \xi_k\text{'s)} \\
= \; & F\left(\begin{pmatrix}\boldsymbol{0} \\ 1\end{pmatrix}, \begin{pmatrix}\boldsymbol{0} \\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\boldsymbol{0} \\ 1\end{pmatrix}\right) \quad \text{(zero-mean of } \xi_k\text{'s)} \\
= \; & 0,
\end{aligned}$$

where the last equality is due to the fact that the last component of $\boldsymbol{F}$ is 0. □

Lemma 5.2.4 suggests that one may enumerate the 2^d possible combinations of $\left(\begin{pmatrix}\xi_1 \boldsymbol{y}^1 \\ 1\end{pmatrix}, \begin{pmatrix}\xi_2 \boldsymbol{y}^2 \\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\xi_d \boldsymbol{y}^d \\ 1\end{pmatrix}\right)$ and pick the one with the largest value of function F (or via a simple randomization procedure), to generate a feasible solution for the inhomogeneous multilinear form optimization (TP_S) from a feasible solution for the homogeneous multilinear form optimization $(TP_S(1))$, with a controlled quality deterioration.

It plays a key role in proving the approximation ratio for (TP_S), which is a byproduct in this section.

Theorem 5.2.5 *(TP_S) admits a polynomial-time approximation algorithm with approximation ratio $2^{-\frac{3d}{2}}(n+1)^{-\frac{d-2}{2}}$.*

Proof. Let $(\bar{\boldsymbol{y}}^1, \bar{\boldsymbol{y}}^2, \cdots, \bar{\boldsymbol{y}}^d)$ be the feasible solution found in Step 2 of Algorithm 5.2.1 satisfying (5.3), and let $\boldsymbol{\eta} = (\eta_1, \eta_2, \cdots, \eta_d)^{\mathrm{T}}$ with all η_k's being independent and taking values 1 and -1 such that $\mathrm{E}[\eta_k] = y_h^k$. By applying Lemma 5.2.4, (5.4) explicitly implies

$$\begin{aligned} & F(\bar{\boldsymbol{y}}^1, \bar{\boldsymbol{y}}^2, \cdots, \bar{\boldsymbol{y}}^d) \\ = & - \sum_{\boldsymbol{\beta}\in\mathbb{B}^d, \prod_{k=1}^d \beta_k = -1} \mathrm{Prob}\{\boldsymbol{\eta} = \boldsymbol{\beta}\} F\left(\begin{pmatrix}\beta_1 \boldsymbol{y}^1 \\ 1\end{pmatrix}, \begin{pmatrix}\beta_2 \boldsymbol{y}^2 \\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\beta_d \boldsymbol{y}^d \\ 1\end{pmatrix}\right) \\ & + \sum_{\boldsymbol{\beta}\in\mathbb{B}^d, \prod_{k=1}^d \beta_k = 1} \mathrm{Prob}\{\boldsymbol{\eta} = \boldsymbol{\beta}\} F\left(\begin{pmatrix}\beta_1 \boldsymbol{y}^1 \\ 1\end{pmatrix}, \begin{pmatrix}\beta_2 \boldsymbol{y}^2 \\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\beta_d \boldsymbol{y}^d \\ 1\end{pmatrix}\right), \end{aligned}$$

and (5.5) explicitly implies

$$\sum_{\boldsymbol{\beta}\in\mathbb{B}^d} F\left(\begin{pmatrix}\beta_1 \boldsymbol{y}^1 \\ 1\end{pmatrix}, \begin{pmatrix}\beta_2 \boldsymbol{y}^2 \\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\beta_d \boldsymbol{y}^d \\ 1\end{pmatrix}\right) = 0.$$

Combing the above two equalities, for any constant c, we have

$$\begin{aligned} & F(\bar{\boldsymbol{y}}^1, \bar{\boldsymbol{y}}^2, \cdots, \bar{\boldsymbol{y}}^d) \\ = & \sum_{\boldsymbol{\beta}\in\mathbb{B}^d, \prod_{k=1}^d \beta_k = -1} (c - \mathrm{Prob}\{\boldsymbol{\eta} = \boldsymbol{\beta}\}) F\left(\begin{pmatrix}\beta_1 \boldsymbol{y}^1 \\ 1\end{pmatrix}, \begin{pmatrix}\beta_2 \boldsymbol{y}^2 \\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\beta_d \boldsymbol{y}^d \\ 1\end{pmatrix}\right) \\ & + \sum_{\boldsymbol{\beta}\in\mathbb{B}^d, \prod_{k=1}^d \beta_k = 1} (c + \mathrm{Prob}\{\boldsymbol{\eta} = \boldsymbol{\beta}\}) F\left(\begin{pmatrix}\beta_1 \boldsymbol{y}^1 \\ 1\end{pmatrix}, \begin{pmatrix}\beta_2 \boldsymbol{y}^2 \\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\beta_d \boldsymbol{y}^d \\ 1\end{pmatrix}\right). \end{aligned} \tag{5.6}$$

If we let

$$c = \max_{\boldsymbol{\beta}\in\mathbb{B}^d, \prod_{k=1}^d \beta_k = -1} \mathrm{Prob}\{\boldsymbol{\eta} = \boldsymbol{\beta}\},$$

then the coefficients of each term in (5.6) will be nonnegative. Therefore we are able to find $\boldsymbol{\beta}' \in \mathbb{B}^d$, such that

$$F\left(\begin{pmatrix}\beta_1' \boldsymbol{y}^1 \\ 1\end{pmatrix}, \begin{pmatrix}\beta_2' \boldsymbol{y}^2 \\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\beta_d' \boldsymbol{y}^d \\ 1\end{pmatrix}\right) \geq \tau_0 F(\bar{\boldsymbol{y}}^1, \bar{\boldsymbol{y}}^2, \cdots, \bar{\boldsymbol{y}}^d), \tag{5.7}$$

where

$$\begin{aligned}\tau_0 &= \left(\sum_{\boldsymbol{\beta}\in\mathbb{B}^d, \prod_{k=1}^d \beta_k = 1} (c + \text{Prob}\{\boldsymbol{\eta} = \boldsymbol{\beta}\}) + \sum_{\boldsymbol{\beta}\in\mathbb{B}^d, \prod_{k=1}^d \beta_k = -1} (c - \text{Prob}\{\boldsymbol{\eta} = \boldsymbol{\beta}\}) \right)^{-1} \\ &\geq \left(2^{d-1}c + \sum_{\boldsymbol{\beta}\in\mathbb{B}^d, \prod_{k=1}^d \beta_k = 1} \text{Prob}\{\boldsymbol{\eta} = \boldsymbol{\beta}\} + (2^{d-1} - 1)c \right)^{-1} \\ &\geq \left(2^{d-1} + 1 + 2^{d-1} - 1\right)^{-1} = 2^{-d}.\end{aligned}$$

Let us denote $\bar{\boldsymbol{z}}^k := \begin{pmatrix} \beta'_k \boldsymbol{y}^k \\ 1 \end{pmatrix}$ for $k = 1, 2, \ldots, d$. Since $\|\boldsymbol{z}^k\| = \|\beta'_k \boldsymbol{y}^k\| \leq 1$, we know that $(\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \cdots, \bar{\boldsymbol{z}}^k)$ is a feasible solution for (TP_S). By combing with (5.3), we have

$$\begin{aligned} F(\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \cdots, \bar{\boldsymbol{z}}^d) &\geq \tau_0 F(\bar{\boldsymbol{y}}^1, \bar{\boldsymbol{y}}^2, \cdots, \bar{\boldsymbol{y}}^d) \\ &\geq 2^{-d} 2^{-\frac{d}{2}} (n+1)^{-\frac{d-2}{2}} v(TP_S) \\ &= 2^{-\frac{3d}{2}} (n+1)^{-\frac{d-2}{2}} v(TP_S). \end{aligned}$$

□

One may notice that our proposed algorithm for solving (TP_S) is very similar to Steps 2 and 3 of Algorithm 5.2.1, with only minor modification at Step 3, namely we choose a solution in $\arg\max \left\{ F\left(\begin{pmatrix} \beta_1 \boldsymbol{y}^1 \\ 1 \end{pmatrix}, \begin{pmatrix} \beta_2 \boldsymbol{y}^2 \\ 1 \end{pmatrix}, \cdots, \begin{pmatrix} \beta_d \boldsymbol{y}^d \\ 1 \end{pmatrix} \right), \boldsymbol{\beta} \in \mathbb{B}^d \right\}$, instead of choosing a solution in $\arg\max \left\{ F\left(\begin{pmatrix} \beta_1 \boldsymbol{y}^1/d \\ 1 \end{pmatrix}, \begin{pmatrix} \beta_2 \boldsymbol{y}^2/d \\ 1 \end{pmatrix}, \cdots, \begin{pmatrix} \beta_d \boldsymbol{y}^d/d \\ 1 \end{pmatrix} \right), \boldsymbol{\beta} \in \mathbb{B}^d \right\}$. The reason to divide d at Step 3 in Algorithm 5.2.1 (to solve (P_S)) will become clear later. Finally, we remark again that it is unnecessary to enumerate all possible 2^d combinations in this step, as (5.6) suggests that a simple randomization process will serve the same purpose, especially when d is large. In the latter case, we will end up with a *polynomial-time randomized approximation algorithm*; otherwise, the computational complexity of the procedure is deterministic and is polynomial-time.

5.2.4 Feasible Solution Assembling

Finally we come to the last step of the scheme. In Step 4 of Algorithm 5.2.1, a polarization formula $\bar{\boldsymbol{z}}(\beta) = \beta_1(d+1)\bar{\boldsymbol{z}}^1 + \sum_{k=2}^d \beta_k \bar{\boldsymbol{z}}^k$ with $\boldsymbol{\beta} \in \mathbb{B}^d$ and $\beta_1 = \prod_{k=2}^d \beta_k = 1$ is proposed. In fact, searching over all $\boldsymbol{\beta} \in \mathbb{B}^d$ will possibly improve the solution, although the worst-case performance ratio will remain the same. Moreover, one may choose $\bar{\boldsymbol{z}}^1$ or any other $\bar{\boldsymbol{z}}^k$ to play the same role here; alternatively one may enumerate $\beta_\ell(d+1)\bar{\boldsymbol{z}}^\ell + \sum_{1\leq k\leq d, k\neq \ell} \beta_k \bar{\boldsymbol{z}}^k$ over all $\boldsymbol{\beta} \in \mathbb{B}^d$ and $1 \leq \ell \leq d$, and take the best

possible solution; again, this will not change the theoretical performance ratio. The polarization formula at Step 4 of Algorithm 5.2.1 works for any fixed degree d, and we shall complete the final stage of the proof of Theorem 5.2.2. Specifically, we shall prove that by letting

$$\boldsymbol{z} = \arg\max \left\{ p(\mathbf{0});\, p\left(\frac{\boldsymbol{z}(\beta)}{z_h(\beta)}\right), \boldsymbol{\beta} \in \mathbb{B}^d \text{ and } \beta_1 = \prod_{k=2}^{d} \beta_k = 1 \right\}$$

with $\bar{\boldsymbol{z}}(\beta) = \beta_1(d+1)\bar{\boldsymbol{z}}^1 + \sum_{k=2}^{d} \beta_k \bar{\boldsymbol{z}}^k$, we have

$$p(\boldsymbol{z}) - \underline{v}(P_S) \geq \tau(P_S)\,(v(P_S) - \underline{v}(P_S))\,. \tag{5.8}$$

First, the solution $(\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \cdots, \bar{\boldsymbol{z}}^d)$ as established at Step 3 of Algorithm 5.2.1 satisfies $\|\boldsymbol{z}^k\| \leq 1/d$ (notice we divided d in each term at Step 3) and $z_h^k = 1$ for $k = 1, 2, \ldots, d$. A same proof of Theorem 5.2.5 can show that

$$F(\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \cdots, \bar{\boldsymbol{z}}^d) \geq d^{-d} 2^{-\frac{3d}{2}} (n+1)^{-\frac{d-2}{2}}\, v(TP_S) \geq 2^{-\frac{3d}{2}} d^{-d} (n+1)^{-\frac{d-2}{2}}\, v(P_S). \tag{5.9}$$

It is easy to see that

$$2 \leq |z_h(\beta)| \leq 2d \text{ and } \|\boldsymbol{z}(\beta)\| \leq (d+1)/d + (d-1)/d = 2. \tag{5.10}$$

Thus $\bar{\boldsymbol{z}}(\beta)/z_h(\beta)$ is a feasible solution for $(\bar{P}_S)$, and so $f(\bar{\boldsymbol{z}}(\beta)/z_h(\beta)) \geq \underline{v}(\bar{P}_S) = \underline{v}(P_S)$. Moreover, we shall argue below that

$$\beta_1 = 1 \Longrightarrow f(\bar{\boldsymbol{z}}(\beta)) \geq (2d)^d\, \underline{v}(P_S). \tag{5.11}$$

If this were not the case, then $f\left(\bar{\boldsymbol{z}}(\beta)/(2d)\right) < \underline{v}(P_S) \leq 0$. Notice that $\beta_1 = 1$ implies $z_h(\beta) > 0$, and thus we have

$$f\left(\frac{\bar{\boldsymbol{z}}(\beta)}{z_h(\beta)}\right) = \left(\frac{2d}{z_h(\beta)}\right)^d f\left(\frac{\bar{\boldsymbol{z}}(\beta)}{2d}\right) \leq f\left(\frac{\bar{\boldsymbol{z}}(\beta)}{2d}\right) < \underline{v}(P_S),$$

which contradicts the feasibility of $\bar{\boldsymbol{z}}(\beta)/z_h(\beta)$.

Suppose $\xi_1, \xi_2, \cdots, \xi_d$ are i.i.d. random variables, each taking values 1 and -1 with equal probability 1/2. By the link Lemma 4.2.1, noticing that $f\left(\bar{\boldsymbol{z}}(-\xi)\right) = f\left(-\bar{\boldsymbol{z}}(\xi)\right) =$

$(-1)^d f(\bar{\boldsymbol{z}}(\xi))$, we have

$$
\begin{aligned}
&d!F\left((d+1)\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \cdots, \bar{\boldsymbol{z}}^d\right) = \mathrm{E}\left[\prod_{k=1}^{d} \xi_k f(\bar{\boldsymbol{z}}(\xi))\right] \\
&= \frac{1}{4}\mathrm{E}\left[f(\bar{\boldsymbol{z}}(\xi)) \middle| \xi_1 = 1, \prod_{k=2}^{d} \xi_k = 1\right] - \frac{1}{4}\mathrm{E}\left[f(\bar{\boldsymbol{z}}(\xi)) \middle| \xi_1 = 1, \prod_{k=2}^{d} \xi_k = -1\right] \\
&\quad - \frac{1}{4}\mathrm{E}\left[f(\bar{\boldsymbol{z}}(\xi)) \middle| \xi_1 = -1, \prod_{k=2}^{d} \xi_k = 1\right] + \frac{1}{4}\mathrm{E}\left[f(\bar{\boldsymbol{z}}(\xi)) \middle| \xi_1 = -1, \prod_{k=2}^{d} \xi_k = -1\right] \\
&= \frac{1}{4}\mathrm{E}\left[f(\bar{\boldsymbol{z}}(\xi)) \middle| \xi_1 = 1, \prod_{k=2}^{d} \xi_k = 1\right] - \frac{1}{4}\mathrm{E}\left[f(\bar{\boldsymbol{z}}(\xi)) \middle| \xi_1 = 1, \prod_{k=2}^{d} \xi_k = -1\right] \\
&\quad - \frac{1}{4}\mathrm{E}\left[f(\bar{\boldsymbol{z}}(-\xi)) \middle| \xi_1 = 1, \prod_{k=2}^{d} \xi_k = (-1)^{d-1}\right] + \frac{1}{4}\mathrm{E}\left[f(\bar{\boldsymbol{z}}(-\xi)) \middle| \xi_1 = 1, \prod_{k=2}^{d} \xi_k = (-1)^{d}\right].
\end{aligned}
$$

By inserting and canceling a constant term, the above expression further leads to

$$
\begin{aligned}
&d!F\left((d+1)\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \cdots, \bar{\boldsymbol{z}}^d\right) = \mathrm{E}\left[\prod_{k=1}^{d} \xi_k f(\bar{\boldsymbol{z}}(\xi))\right] \\
&= \frac{1}{4}\mathrm{E}\left[\left(f(\bar{\boldsymbol{z}}(\xi)) - (2d)^d \underline{v}(P_S)\right) \middle| \xi_1 = 1, \prod_{k=2}^{d} \xi_k = 1\right] \\
&\quad - \frac{1}{4}\mathrm{E}\left[\left(f(\bar{\boldsymbol{z}}(\xi)) - (2d)^d \underline{v}(P_S)\right) \middle| \xi_1 = 1, \prod_{k=2}^{d} \xi_k = -1\right] \\
&\quad + \frac{(-1)^{d-1}}{4}\mathrm{E}\left[\left(f(\bar{\boldsymbol{z}}(\xi)) - (2d)^d \underline{v}(P_S)\right) \middle| \xi_1 = 1, \prod_{k=2}^{d} \xi_k = (-1)^{d-1}\right] \\
&\quad + \frac{(-1)^{d}}{4}\mathrm{E}\left[\left(f(\bar{\boldsymbol{z}}(\xi)) - (2d)^d \underline{v}(P_S)\right) \middle| \xi_1 = 1, \prod_{k=2}^{d} \xi_k = (-1)^{d}\right] \\
&\leq \frac{1}{2}\mathrm{E}\left[\left(f(\bar{\boldsymbol{z}}(\xi)) - (2d)^d \underline{v}(P_S)\right) \middle| \xi_1 = 1, \prod_{k=2}^{d} \xi_k = 1\right], \qquad (5.12)
\end{aligned}
$$

where the last inequality is due to (5.11). Therefore, there is a binary vector $\boldsymbol{\beta}' \in \mathbb{B}^d$ with $\beta'_1 = \prod_{k=2}^{d} \beta'_k = 1$, such that

$$
f(\bar{\boldsymbol{z}}(\beta')) - (2d)^d \underline{v}(P_S) \geq 2d!F((d+1)\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \cdots, \bar{\boldsymbol{z}}^d) \geq 2^{-\frac{3d}{2}+1}(d+1)!d^{-d}(n+1)^{-\frac{d-2}{2}} v(P_S),
$$

where the last step is due to (5.9).

Below we argue $\boldsymbol{z} = \arg\max\left\{p(\mathbf{0}); p(\boldsymbol{z}(\beta)/z_h(\beta)), \boldsymbol{\beta} \in \mathbb{B}^d \text{ and } \beta_1 = \prod_{k=2}^{d} \beta_k = 1\right\}$ satisfies (5.8). In fact, if $-\underline{v}(P_S) \geq \tau(P_S)(v(P_S) - \underline{v}(P_S))$, then $\mathbf{0}$ trivially satisfies (5.8), and so does $\boldsymbol{z}$ in this case. Otherwise, if $-\underline{v}(P_S) < \tau(P_S)(v(P_S) - \underline{v}(P_S))$, then we have

$$
v(P_S) > (1 - \tau(P_S))(v(P_S) - \underline{v}(P_S)) \geq \frac{v(P_S) - \underline{v}(P_S)}{2},
$$

which implies

$$f\left(\frac{\bar{\boldsymbol{z}}(\beta')}{2d}\right)-\underline{v}(P_S)\geq(2d)^{-d}2^{-\frac{3d}{2}+1}(d+1)!d^{-d}(n+1)^{-\frac{d-2}{2}}v(P_S)\geq\tau(P_S)\left(v(P_S)-\underline{v}(P_S)\right).$$

The above inequality also implies that $f\left(\bar{\boldsymbol{z}}(\beta')/(2d)\right)>0$. Recall that $\beta'_1=1$ implies $z_h(\beta')>0$, and thus $2d/z_h(\beta')\geq 1$ by (5.10). Therefore, we have

$$p(\boldsymbol{z})\geq p\left(\frac{\boldsymbol{z}(\beta')}{z_h(\beta')}\right)=f\left(\frac{\bar{\boldsymbol{z}}(\beta')}{z_h(\beta')}\right)=\left(\frac{2d}{z_h(\beta')}\right)^d f\left(\frac{\bar{\boldsymbol{z}}(\beta')}{2d}\right)\geq f\left(\frac{\bar{\boldsymbol{z}}(\beta')}{2d}\right).$$

This shows that $\boldsymbol{z}$ satisfies (5.8) in both cases, which concludes the whole proof.

5.3 Polynomial with Ellipsoidal Constraints

In this section, we consider an extension of (P_S), namely

$$\begin{array}{lll}(P_Q) & \max & p(\boldsymbol{x})\\ & \text{s.t.} & \boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x}\leq 1,\ i=1,2,\ldots,m,\\ & & x\in\mathbb{R}^n,\end{array}$$

where $\boldsymbol{Q}_i\succeq 0$ for $i=1,2,\ldots,m$, and $\sum_{i=1}^m\boldsymbol{Q}_i\succ 0$. Since $p(\boldsymbol{x})$ is assumed to have no constant term, we know that $\underline{v}(P_Q)\leq 0\leq v(P_Q)$.

Here, like in Section 5.2, we propose a polynomial-time randomized algorithm for approximately solving (P_Q), with a worst-case relative performance ratio. The main algorithm and the approximation result of this section is the following.

Algorithm 5.3.1

- *INPUT: an n-dimensional d-th degree polynomial function $p(\boldsymbol{x})$, matrices $\boldsymbol{Q}_i\in\mathbb{R}^{n\times n}$, $\boldsymbol{Q}_i\succeq 0$ for all $1\leq i\leq m$ with $\sum_{i=1}^m\boldsymbol{Q}_i\succ 0$.*

1 *Rewrite $p(\boldsymbol{x})-p(\boldsymbol{0})=F(\underbrace{\bar{\boldsymbol{x}},\bar{\boldsymbol{x}},\cdots,\bar{\boldsymbol{x}}}_{d})$ when $x_h=1$ as in (5.2), with $\boldsymbol{F}$ being an $(n+1)$-dimensional d-th order super-symmetric tensor.*

2 *Apply Algorithm 3.3.2 to solve the problem*

$$\begin{array}{ll}\max & F(\bar{\boldsymbol{x}}^1,\bar{\boldsymbol{x}}^2,\cdots,\bar{\boldsymbol{x}}^d)\\ \text{s.t.} & (\bar{\boldsymbol{x}}^k)^{\mathrm{T}}\begin{bmatrix}\boldsymbol{Q}_i & \boldsymbol{0}\\ \boldsymbol{0}^{\mathrm{T}} & 1\end{bmatrix}\bar{\boldsymbol{x}}^k\leq 1,\ k=1,2,\ldots,d,\ i=1,2,\ldots,m\end{array}$$

approximately, and get a feasible solution $(\bar{\boldsymbol{y}}^1,\bar{\boldsymbol{y}}^2,\cdots,\bar{\boldsymbol{y}}^d)$.

3 *Compute* $(\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \cdots, \bar{\boldsymbol{z}}^d) = \arg\max\left\{F\left(\binom{\xi_1 \boldsymbol{y}^1/d}{1}, \binom{\xi_2 \boldsymbol{y}^2/d}{1}, \cdots, \binom{\xi_d \boldsymbol{y}^d/d}{1}\right), \boldsymbol{\xi} \in \mathbb{B}^d\right\}$.

4 *Compute* $\boldsymbol{z} = \arg\max\left\{p(\mathbf{0}); p(\boldsymbol{z}(\beta)/z_h(\beta)), \boldsymbol{\beta} \in \mathbb{B}^d \text{ and } \beta_1 = \prod_{k=2}^d \beta_k = 1\right\}$, *with* $\bar{\boldsymbol{z}}(\beta) = \beta_1(d+1)\bar{\boldsymbol{z}}^1 + \sum_{k=2}^d \beta_k \bar{\boldsymbol{z}}^k$.

- *OUTPUT: a feasible solution* $\boldsymbol{z} \in \mathbb{R}^n$.

Theorem 5.3.1 (P_Q) *admits a polynomial-time randomized approximation algorithm with relative approximation ratio* $\tau(P_Q)$, *where*

$$\tau(P_Q) := 2^{-\frac{5d}{2}}(d+1)!\, d^{-2d}(n+1)^{-\frac{d-2}{2}}\Omega\left(\log^{-(d-1)} m\right) = \Omega\left(n^{-\frac{d-2}{2}}\log^{-(d-1)} m\right).$$

Our scheme for solving general polynomial optimization model (P_Q) is similar to that for solving (P_S) in Section 5.2. The main difference lies in Step 2, where a different relaxation model requires a different solution method to cope with. The method in question is Algorithm 3.3.2.

The proof of Theorem 5.3.1 is similar to that of Theorem 5.2.2. Here we only illustrate the main ideas and skip the details.

By homogenizing $p(\boldsymbol{x})$ who has no constant term, we may rewrite (P_Q) as

$$\begin{array}{lll} (\bar{P}_Q) & \max & f(\bar{\boldsymbol{x}}) \\ & \text{s.t.} & \bar{\boldsymbol{x}} = \begin{pmatrix} \boldsymbol{x} \\ x_h \end{pmatrix}, \\ & & \boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x} \le 1,\ \boldsymbol{x} \in \mathbb{R}^n,\ i = 1, 2, \ldots, m, \\ & & x_h = 1, \end{array}$$

which can be relaxed to the inhomogeneous multilinear function problem

$$\begin{array}{lll} (TP_Q) & \max & F(\bar{\boldsymbol{x}}^1, \bar{\boldsymbol{x}}^2, \cdots, \bar{\boldsymbol{x}}^d) \\ & \text{s.t.} & \bar{\boldsymbol{x}}^k = \begin{pmatrix} \boldsymbol{x}^k \\ x_h^k \end{pmatrix},\ k = 1, 2, \ldots, d, \\ & & (\boldsymbol{x}^k)^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x}^k \le 1,\ \boldsymbol{x}^k \in \mathbb{R}^n,\ k = 1, 2, \ldots, d,\ i = 1, 2, \ldots, m, \\ & & x_h^k = 1,\ k = 1, 2, \ldots, d, \end{array}$$

where $F(\underbrace{\bar{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \cdots, \bar{\boldsymbol{x}}}_{d}) = f(\bar{\boldsymbol{x}})$ with $\boldsymbol{F}$ being super-symmetric. We then further relax (TP_Q) to the multilinear form optimization model $(TP_Q(\sqrt{2}))$, with

$$\begin{array}{lll} (TP_Q(t)) & \max & F(\bar{\boldsymbol{x}}^1, \bar{\boldsymbol{x}}^2, \cdots, \bar{\boldsymbol{x}}^d) \\ & & (\bar{\boldsymbol{x}}^k)^{\mathrm{T}}\bar{\boldsymbol{Q}}_i\bar{\boldsymbol{x}}^k \le t^2,\ k = 1, 2, \ldots, d,\ i = 1, 2, \ldots, m, \\ & & \bar{\boldsymbol{x}}^k \in \mathbb{R}^{n+1},\ k = 1, 2, \ldots, d, \end{array}$$

where $\bar{\boldsymbol{Q}}_i = \begin{bmatrix} \boldsymbol{Q}_i & \boldsymbol{0} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix}$ for $i = 1, 2, \ldots, m$.

By Theorem 3.3.4, for any $t > 0$, $(TP_Q(t))$ admits a polynomial-time randomized approximation algorithm with approximation ratio $(n+1)^{-\frac{d-2}{2}}\Omega\left(\log^{-(d-1)} m\right)$, and $v(TP_Q(t)) = t^d\, v(TP_Q(1))$. Thus the approximate solution $(\bar{\boldsymbol{y}}^1, \bar{\boldsymbol{y}}^2, \cdots, \bar{\boldsymbol{y}}^d)$ found by Step 2 of Algorithm 5.3.1 satisfies

$$\begin{aligned} F(\bar{\boldsymbol{y}}^1, \bar{\boldsymbol{y}}^2, \cdots, \bar{\boldsymbol{y}}^d) &\geq (n+1)^{-\frac{d-2}{2}}\Omega\left(\log^{-(d-1)} m\right) v(TP_Q(1)) \\ &= (\sqrt{2})^{-d}(n+1)^{-\frac{d-2}{2}}\Omega\left(\log^{-(d-1)} m\right) v(TP_Q(\sqrt{2})) \\ &\geq 2^{-\frac{d}{2}}(n+1)^{-\frac{d-2}{2}}\Omega\left(\log^{-(d-1)} m\right) v(TP_Q). \end{aligned}$$

Noticing that $(y_h^k)^2 \leq (\bar{\boldsymbol{y}}^k)^{\mathrm{T}}\bar{\boldsymbol{Q}}_1\bar{\boldsymbol{y}}^k \leq 1$ for $k = 1, 2, \ldots, d$, we again apply Lemma 5.2.4 to $(\bar{\boldsymbol{y}}^1, \bar{\boldsymbol{y}}^2, \cdots, \bar{\boldsymbol{y}}^d)$, and use the same argument as in the proof of Theorem 5.2.5.

Let $c = \max_{\boldsymbol{\beta}\in\mathbb{B}^d, \prod_{k=1}^d \beta_k = -1} \mathrm{Prob}\{\boldsymbol{\eta} = \boldsymbol{\beta}\}$, where $\boldsymbol{\eta} = (\eta_1, \eta_2, \cdots, \eta_d)^{\mathrm{T}}$ and its components are independent random variables, each taking values 1 and -1 with $\mathrm{E}[\eta_k] = y_h^k$ for $k = 1, 2, \ldots, d$. Then we are able to find a binary vector $\boldsymbol{\beta}' \in \mathbb{B}^d$, such that

$$\begin{aligned} F\left(\begin{pmatrix}\beta_1'\boldsymbol{y}^1 \\ 1\end{pmatrix}, \begin{pmatrix}\beta_2'\boldsymbol{y}^2 \\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\beta_d'\boldsymbol{y}^d \\ 1\end{pmatrix}\right) &\geq \tau_0 F(\bar{\boldsymbol{y}}^1, \bar{\boldsymbol{y}}^2, \cdots, \bar{\boldsymbol{y}}^d) \\ &\geq 2^{-d} F(\bar{\boldsymbol{y}}^1, \bar{\boldsymbol{y}}^2, \cdots, \boldsymbol{y}^d) \\ &\geq 2^{-\frac{3d}{2}}(n+1)^{-\frac{d-2}{2}}\Omega\left(\log^{-(d-1)} m\right) v(TP_Q). \end{aligned}$$

This proves the following theorem as a byproduct.

Theorem 5.3.2 *(TP_Q) admits a polynomial-time randomized approximation algorithm with approximation ratio $2^{-\frac{3d}{2}}(n+1)^{-\frac{d-2}{2}}\Omega\left(\log^{-(d-1)} m\right)$.*

To prove the main theorem in this section (Theorem 5.3.1), we only need to check the feasibility of $\boldsymbol{z}$ generated by Algorithm 5.3.1, while the worst-case performance ratio can be proven by the similar argument in Section 5.2.4. Indeed, $(\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \cdots, \bar{\boldsymbol{z}}^d)$ at Step 3 of Algorithm 5.3.1 satisfies

$$(\boldsymbol{z}^k)^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{z}^k \leq 1/d^2 \quad \forall\, 1 \leq i \leq m, 1 \leq k \leq d.$$

For any binary vector $\boldsymbol{\beta} \in \mathbb{B}^d$, as $\bar{\boldsymbol{z}}(\beta) = \beta_1(d+1)\bar{\boldsymbol{z}}^1 + \sum_{k=2}^d \beta_k\bar{\boldsymbol{z}}^k$, we have $2 \leq |z_h(\beta)| \leq 2d$. Noticing by the Cauchy-Schwarz inequality,

$$|(\boldsymbol{z}^j)^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{z}^k| \leq \|\boldsymbol{Q}_i^{\frac{1}{2}}\boldsymbol{z}^j\| \cdot \|\boldsymbol{Q}_i^{\frac{1}{2}}\boldsymbol{z}^k\| \leq 1/d^2 \quad \forall\, 1 \leq i \leq m,\ 1 \leq j, k \leq d,$$

it follows that

$$(\boldsymbol{z}(\beta))^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{z}(\beta) \le 2d \cdot 2d \cdot 1/d^2 = 4 \quad \forall\, 1 \le i \le m.$$

Thus $\boldsymbol{z}(\beta)/z_h(\beta)$ is a feasible solution for (P_Q), which implies $\boldsymbol{z}$ is also feasible.

To conclude this section, we remark here that (P_Q) includes as a special case the optimization of a general polynomial function over a central-symmetric polytope:

$$\begin{array}{ll} \max & p(\boldsymbol{x}) \\ \text{s.t.} & -1 \le (\boldsymbol{a}^i)^{\mathrm{T}}\boldsymbol{x} \le 1,\ i = 1, 2, \ldots, m, \\ & \boldsymbol{x} \in \mathbb{R}^n, \end{array}$$

with $\mathrm{rank}\,(\boldsymbol{a}^1, \boldsymbol{a}^2, \cdots, \boldsymbol{a}^m) = n$.

5.4 Polynomial with General Convex Constraints

In this section we study polynomial optimization model in a generic constraint format:

$$\boxed{\begin{array}{lll} (P_G) & \max & p(\boldsymbol{x}) \\ & \text{s.t.} & \boldsymbol{x} \in G, \end{array}}$$

where $G \subset \mathbb{R}^n$ is a given convex compact set. As before, we derive polynomial-time approximation algorithms for solving (P_G). Our approaches make use of the well known *Löwner-John ellipsoids* (see e.g., [20, 86]), which is the following:

Theorem 5.4.1 *Given a convex compact set $G \subset \mathbb{R}^n$ with non-empty interior.*

1. *There exists a unique largest volume ellipsoid $\{\boldsymbol{A}\boldsymbol{x} + \boldsymbol{a} \,|\, \boldsymbol{x} \in \bar{\mathbb{S}}^n\} \subset G$, whose n times linear-size larger ellipsoid $\{n\boldsymbol{A}\boldsymbol{x} + \boldsymbol{a} \,|\, \boldsymbol{x} \in \bar{\mathbb{S}}^n\} \supset G$, and if in addition G is central-symmetric, then $\{\sqrt{n}\boldsymbol{A}\boldsymbol{x} + \boldsymbol{a} \,|\, \boldsymbol{x} \in \bar{\mathbb{S}}^n\} \supset G$;*

2. *There exists a unique smallest volume ellipsoid $\{\boldsymbol{B}\boldsymbol{x} + \boldsymbol{b} \,|\, \boldsymbol{x} \in \bar{\mathbb{S}}^n\} \supset G$, whose n times linear-size smaller ellipsoid $\{\boldsymbol{B}\boldsymbol{x}/n + \boldsymbol{b} \,|\, \boldsymbol{x} \in \bar{\mathbb{S}}^n\} \subset G$, and if in addition G is central-symmetric, then $\{\boldsymbol{B}\boldsymbol{x}/\sqrt{n} + \boldsymbol{b} \,|\, \boldsymbol{x} \in \bar{\mathbb{S}}^n\} \subset G$.*

Armed with the above theorem, if we are able to find the Löwner-John ellipsoid (either the inner or the outer) of the feasible region G in polynomial-time, then the following algorithm approximately solves (P_G) with a worst-case performance ratio.

Algorithm 5.4.1

- *INPUT: an n-dimensional d-th degree polynomial function $p(\boldsymbol{x})$ and a set $G \subset \mathbb{R}^n$.*

1 *Find a scalar $t \in \mathbb{R}$, a vector $\boldsymbol{b} \in \mathbb{R}^n$, and a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ with $\mathrm{rank}\,(\boldsymbol{A}) = m \leq n$, such that two co-centered ellipsoids $E_1 = \{\boldsymbol{A}\boldsymbol{u} + \boldsymbol{b} \,|\, \boldsymbol{u} \in \bar{\mathbb{S}}^m\}$ and $E_2 = \{t\boldsymbol{A}\boldsymbol{u} + \boldsymbol{b} \,|\, \boldsymbol{u} \in \bar{\mathbb{S}}^m\}$ satisfy $E_1 \subset G \subset E_2$.*

2 *Compute a polynomial function $p_0(\boldsymbol{u}) = p(\boldsymbol{A}\boldsymbol{u} + \boldsymbol{b})$ of variable $\boldsymbol{u} \in \mathbb{R}^m$.*

3 *Apply Algorithm 5.2.1 with input $p_0(\boldsymbol{x})$ and output $\boldsymbol{y} \in \bar{\mathbb{S}}^m$.*

4 *Compute $\boldsymbol{z} = \boldsymbol{A}\boldsymbol{y} + \boldsymbol{b}$.*

- *OUTPUT: a feasible solution $\boldsymbol{z} \in G$.*

The key result in this section is the following theorem.

Theorem 5.4.2 *If $\bar{\mathbb{S}}^n \subset G \subset t\bar{\mathbb{S}}^n := \{\boldsymbol{x} \in \mathbb{R}^n \,|\, \|\boldsymbol{x}\| \leq t\}$ for some $t \geq 1$, then (P_G) admits a polynomial-time approximation algorithm with relative approximation ratio $\tau(P_G)(t)$, where*

$$\tau(P_G)(t) := 2^{-2d}(d+1)!\, d^{-2d}(n+1)^{-\frac{d-2}{2}}(t^2+1)^{-\frac{d}{2}} = \Omega\left(n^{-\frac{d-2}{2}}t^{-d}\right).$$

Proof. By homogenizing the object function of (P_G), we get the equivalent problem

$$\begin{array}{lll} (\bar{P}_G) & \max & f(\bar{\boldsymbol{x}}) \\ & \text{s.t.} & \bar{\boldsymbol{x}} = \begin{pmatrix} \boldsymbol{x} \\ x_h \end{pmatrix}, \\ & & \boldsymbol{x} \in G,\ x_h = 1, \end{array}$$

where $f(\bar{\boldsymbol{x}}) = p(\boldsymbol{x})$ if $x_h = 1$, and $f(\bar{\boldsymbol{x}})$ is an $(n+1)$-dimensional homogeneous polynomial function of degree d. If we write $f(\bar{\boldsymbol{x}}) = F(\underbrace{\bar{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \cdots, \bar{\boldsymbol{x}}}_{d})$ with $\boldsymbol{F}$ being super-symmetric, then $(\bar{P}_G)$ can be relaxed to the inhomogeneous multilinear form problem

$$\begin{array}{lll} (TP_G) & \max & F(\bar{\boldsymbol{x}}^1, \bar{\boldsymbol{x}}^2, \cdots, \bar{\boldsymbol{x}}^d) \\ & \text{s.t.} & \bar{\boldsymbol{x}}^k = \begin{pmatrix} \boldsymbol{x}^k \\ x_h^k \end{pmatrix},\ k = 1, 2, \ldots, d, \\ & & \boldsymbol{x}^k \in G,\ x_h^k = 1,\ k = 1, 2, \ldots, d. \end{array}$$

Recall that in Section 5.2.2, we have defined

$$(TP_S(t)) \quad \max \quad F(\bar{\boldsymbol{x}}^1, \bar{\boldsymbol{x}}^2, \cdots, \bar{\boldsymbol{x}}^d)$$
$$\text{s.t.} \quad \|\bar{\boldsymbol{x}}^k\| \le t,\ \bar{\boldsymbol{x}}^k \in \mathbb{R}^{n+1},\ k = 1, 2, \ldots, d.$$

As $\boldsymbol{x}^k \in G \subset t\bar{\mathbb{S}}^n$, it follows that $\|\bar{\boldsymbol{x}}^k\| \le \sqrt{t^2+1}$ in (TP_G). Therefore, $(TP_S(\sqrt{t^2+1}))$ is a relaxation of (TP_G), and $v(TP_S(\sqrt{t^2+1})) \ge v(TP_G) \ge v(\bar{P}_G) = v(P_G)$. The rest of the proof follows similarly as that in Section 5.2.4. Specifically, we are able to construct a feasible solution $\boldsymbol{x} \in \bar{\mathbb{S}}^n \subset G$ in polynomial-time with a relative performance ratio $\tau(P_G)(t)$. □

Observe that any ellipsoid can be linearly transformed to the Euclidean ball. By a variable transformation if necessary, we are led to the main result in this section.

Corollary 5.4.3 *Given a bounded set $G \subset \mathbb{R}^n$, if two co-centered ellipsoids $E_1 = \{\boldsymbol{A}\boldsymbol{u} + \boldsymbol{b} \,|\, \boldsymbol{u} \in \bar{\mathbb{S}}^n\}$ and $E_2 = \{t\boldsymbol{A}\boldsymbol{u} + \boldsymbol{b} \,|\, \boldsymbol{u} \in \bar{\mathbb{S}}^n\}$ can be found in polynomial-time, satisfying $E_1 \subset G \subset E_2$, then (P_G) admits a polynomial-time approximation algorithm with relative approximation ratio $\tau(P_G)(t)$.*

We remark that in fact the set G in Theorem 5.4.2 and Corollary 5.4.3 does not need to be convex, as long as the two required ellipsoids are in place. However, the famous Löwner-John theorem guarantees the existence of such inner and outer ellipsoids required in Corollary 5.4.3 for any convex compact set, with $t = n$ for G being non-central-symmetric, and $t = \sqrt{n}$ for G being central-symmetric. Thus, if we are able to find a pair of ellipsoids (E_1, E_2) in polynomial-time for G, then (P_G) can be solved by a polynomial-time approximation algorithm with relative approximation ratio $\tau(P_G)(t)$. Indeed, it is possible to compute in polynomial-time the Löwner-John ellipsoids in several interesting cases. Below is a list of such cases (assuming G is bounded); for the details one is referred to [20, 86]:

- $G = \{\boldsymbol{x} \in \mathbb{R}^n \,|\, (\boldsymbol{a}^i)^{\mathrm{T}}\boldsymbol{x} \le b_i,\ i = 1, 2, \ldots, m\}$;
- $G = \mathrm{conv}\,\{\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^m\}$, where $\boldsymbol{x}^i \in \mathbb{R}^n$ for $i = 1, 2, \ldots, m$;
- $G = \bigcap_{i=1}^m E_i$, where E_i is an ellipsoid in $\mathbb{R}^n$ for $i = 1, 2, \ldots, m$;
- $G = \mathrm{conv}\,\{\bigcup_{i=1}^m E_i\}$, where E_i is an ellipsoid in $\mathbb{R}^n$ for $i = 1, 2, \ldots, m$;
- $G = \sum_{i=1}^m E_i := \{\sum_{i=1}^m \boldsymbol{x}^i \,|\, \boldsymbol{x}^i \in E_i,\ i = 1, 2, \ldots, m\}$, where E_i is an ellipsoid in $\mathbb{R}^n$ for $i = 1, 2, \ldots, m$.

By Corollary 5.4.3, and the computability of the Löwner-John ellipsoids discussed above, we conclude that for (P_G) with the constraint set G being any of the above cases, then there is a polynomial-time approximation algorithm with a relative approximation quality assurance. In particular, the ratio is $\tau(P_G)(\sqrt{m}) = \Omega\left(n^{-\frac{d-2}{2}} m^{-\frac{d}{2}}\right)$ for the last case, and is $\tau(P_G)(n) = \Omega\left(n^{-\frac{3d-2}{2}}\right)$ for the other cases.

We also remark that (P_Q) : $\max_{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x}\le 1, i=1,2,\ldots,m} p(\boldsymbol{x})$ discussed in Section 5.3, may in principle be solved by directly applying Corollary 5.4.3 as well. If we adopt that approach (Algorithm 5.4.1), then the relative approximation ratio is $\tau(P_G)(\sqrt{n}) = \Omega\left(n^{-\frac{2d-2}{2}}\right)$, which prevails if m is exceedingly large. Taking the better one, the quality ratio in Theorem 5.3.1 can be improved to $\Omega\left(\max\left\{n^{-\frac{d-2}{2}} \log^{-(d-1)} m,\ n^{-\frac{2d-2}{2}}\right\}\right)$.

Our investigation quite naturally leads to a question which is of a general geometric interest itself. Consider the intersection of m co-centered ellipsoids in $\mathbb{R}^n$ as a geometric structure. Denote $\mathcal{E}_{m,n}$ to be the collection of all such structures, or more specifically

$$\mathcal{E}_{m,n} := \left\{ \bigcap_{i=1}^{m} \{\boldsymbol{x} \in \mathbb{R}^n \,|\, \boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x} \le 1\} \;\middle|\; \boldsymbol{Q}_i \succeq 0 \text{ for } i = 1, 2, \ldots, m \text{ and } \sum_{i=1}^{m} \boldsymbol{Q}_i \succ 0 \right\}.$$

For any central-symmetric and convex compact set $G \subset \mathbb{R}^n$ centered at $\boldsymbol{b}$, there exists $E_{m,n} \in \mathcal{E}_{m,n}$ and $t \ge 1$, such that $\boldsymbol{b} + E_{m,n} \subset G \subset \boldsymbol{b} + tE_{m,n}$. Obviously, one can naturally define

$$t(G; m, n) := \inf\{t \,|\, E_{m,n} \in \mathcal{E}_{m,n} \text{ such that } \boldsymbol{b} + E_{m,n} \subset G \subset \boldsymbol{b} + tE_{m,n}\},$$

$$\theta(m, n) := \sup\{t(G; m, n) \,|\, G \subset \mathbb{R}^n \text{ is convex compact and central-symmetric}\}.$$

The famous Löwner-John theorem states that $\theta(1, n) = \sqrt{n}$. Naturally, $\theta(\infty, n) = 1$, because any central-symmetric convex set can be expressed by the intersection of an infinite number of co-centered ellipsoids. It is interesting to compute $\theta(m, n)$ for general m and n. Of course, it is trivial to observe that $\theta(m, n)$ is monotonically decreasing in m for any fixed n. Anyway, if we are able to compute $\theta(m, n)$ and find the corresponding $E_{m,n}$ in polynomial-time, then Theorem 5.3.1 suggests a polynomial-time randomized approximation algorithm of (P_G) with relative approximation ratio $(\theta(m, n))^{-d}\tau(P_Q) = \Omega\left((\theta(m, n))^{-d} n^{-\frac{d-2}{2}} \log^{-(d-1)} m\right)$.

5.5 Applications

The generality of the polynomial optimization models studied in this chapter have versatile applications. In order to better appreciate these models as well as the approx-

imation algorithms presented, in this section we shall discuss a few detailed examples rising from real applications, and show that they are readily formulated by the inhomogeneous polynomial optimization models in this chapter.

5.5.1 Portfolio Selection with Higher Moments

The portfolio selection problem dates back to early 1950', when the seminal work of mean-variance model was proposed by Markowitz [81]. Essentially, in Markowitz's model, the mean of the portfolio return is treated as the 'gain' factor, while the variance of the portfolio return is treated as the 'risk' factor. By minimizing the risk subject to certain target of reward, the mean-variance model is as follows:

$$\begin{array}{lll} (MV) & \min & \boldsymbol{x}^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{x} \\ & \text{s.t.} & \boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{x} = \mu_0, \\ & & \boldsymbol{e}^{\mathrm{T}}\boldsymbol{x} = 1,\ \boldsymbol{x} \geq \boldsymbol{0},\ \boldsymbol{x} \in \mathbb{R}^n, \end{array}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean vector and co-variance matrix of n given assets respectively, and $\boldsymbol{e}$ is the all one vector. This model and its variations have been studies extensively along the history of portfolio management. Despite its popularity and originality, the mean-variance model certainly has drawbacks. An important one is that it neglects the *higher moments* information of the portfolio. Mandelbrot and Hudson [78] made a strong case against a 'normal view' of the investment returns. The use of higher moments in portfolio selection becomes quite necessary, i.e., involving more than the first two moments (e.g., the skewness and the kurtosis of the investment returns) if they are also available. That problem has been receiving much attention in the literature (see e.g., de Athayde and Flôre [10], Prakash et al. [96], Jondeau and Rockinger [60], Kleniati et al. [64], and the references therein). In particular, a very general model in [64] is

$$\begin{array}{lll} (PM) & \max & \alpha\,\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{x} - \beta\,\boldsymbol{x}^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{x} + \gamma\sum_{i,j,k=1}^{n}\varsigma_{ijk}x_ix_jx_k - \delta\sum_{i,j,k,\ell=1}^{n}\kappa_{ijk\ell}x_ix_jx_kx_\ell \\ & \text{s.t.} & \boldsymbol{e}^{\mathrm{T}}\boldsymbol{x} = 1,\ \boldsymbol{x} \geq \boldsymbol{0},\ \boldsymbol{x} \in \mathbb{R}^n, \end{array}$$

where $\boldsymbol{\mu}, \boldsymbol{\Sigma}, (\varsigma_{ijk}), (\kappa_{ijk\ell})$ are the first four central moments of the n given assets. The nonnegative parameters $\alpha, \beta, \gamma, \delta$ measure the investor's preference to the four moments, and they sum up to one, i.e., $\alpha + \beta + \gamma + \delta = 1$.

In fact, the mean-variance model (MV) can be taken as a special case of (PM) with $\gamma = \delta = 0$. The model (PM) is essentially in the frame work of our model (P_G), as

the constraint set is convex and compact. By directly applying Corollary 5.4.3 and the discussion on its applicability in a polytope, it admits a polynomial-time approximation algorithm with relative approximation ratio $\Omega\left(n^{-5}\right)$.

5.5.2 Sensor Network Localization

Suppose in a certain specified region $G \subset \mathbb{R}^3$, there are a set of anchor nodes, denoted by A, and a set of sensor nodes, denoted by S. What we have known are the positions of the anchor nodes $\boldsymbol{a}^j \in G\,(j \in A)$, and the (possibly noisy) distance measurements between anchor nodes and sensor nodes, and between two different sensor nodes, denoted by $d_{ij}\,(i \in S,\, j \in S \cup A)$. The task is to estimate the positions of the unknown sensor nodes $\boldsymbol{x}^i \in G\,(i \in S)$. Luo and Zhang [77] proposed a least square formulation to this sensor network localization problem. Specifically, the problem takes the form of

$$\begin{array}{lll} (SNL) & \min & \sum_{i,j\in S}\left(\|\boldsymbol{x}^i-\boldsymbol{x}^j\|^2-d_{ij}{}^2\right)^2+\sum_{i\in S,j\in A}\left(\|\boldsymbol{x}^i-\boldsymbol{a}^j\|^2-d_{ij}{}^2\right)^2 \\ & \text{s.t.} & \boldsymbol{x}^i \in G,\, i \in S. \end{array}$$

Notice that the objective function of (SNL) is an inhomogeneous quartic polynomial function. If the specified region G is well formed, say the Euclidean ball, an ellipsoid, a polytope, or any other convex compact set that can be sandwiched by two co-centered ellipsoids, then (SNL) can be fit into the model (P_G) in the following way. Suppose $E_1 \subset G \subset E_2$ with E_1 and E_2 being two co-centered ellipsoids, we know by the Löwner-John theorem that E_2 is bounded by three times larger of E_1 in linear size (for the Euclidean ball or an ellipsoid it is 1, for a central-symmetric G it is less than $\sqrt{3}$, and for a general convex compact G it is less than 3). Denote the number of sensor nodes to be $n = |S|$, and denote $\boldsymbol{x} = \left((\boldsymbol{x}^1)^{\mathrm{T}}, (\boldsymbol{x}^2)^{\mathrm{T}}, \cdots, (\boldsymbol{x}^n)^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{3n}$. Then $\boldsymbol{x} \in \underbrace{G \times G \times \cdots \times G}_{n}$, and this feasible region can be sandwiched by two co-centered sets $\underbrace{E_1 \times E_1 \times \cdots \times E_1}_{n}$ and $\underbrace{E_2 \times E_2 \times \cdots \times E_2}_{n}$, which are both intersections of n co-centered ellipsoids, i.e., belonging to $\mathcal{E}_{n,3n}$. According to the discussion at the end of Section 5.4, and noticing in this case $t(G; n, 3n) \leq 3$ is a constant, (SNL) admits a polynomial-time randomized approximation algorithm with relative approximation ratio $\Omega\left(\frac{1}{n \log^3 n}\right)$.

5.6 Numerical Experiments

In this section, we present some preliminary test results for the approximation algorithms proposed in this chapter, to give the readers an impression about how our algorithms work in practice. We shall focus on (P_S) with $d = 4$, specifically, the model being tested is

$$(EP_S) \quad \begin{array}{ll} \max & p(\boldsymbol{x}) = F_4(\boldsymbol{x},\boldsymbol{x},\boldsymbol{x},\boldsymbol{x}) + F_3(\boldsymbol{x},\boldsymbol{x},\boldsymbol{x}) + F_2(\boldsymbol{x},\boldsymbol{x}) + F_1(\boldsymbol{x}) \\ \text{s.t.} & \boldsymbol{x} \in \bar{\mathbb{S}}^n, \end{array}$$

where $\boldsymbol{F}_4 \in \mathbb{R}^{n^4}$, $\boldsymbol{F}_3 \in \mathbb{R}^{n^3}$, $\boldsymbol{F}_2 \in \mathbb{R}^{n^2}$ and $\boldsymbol{F}_1 \in \mathbb{R}^n$, are super-symmetric tensors of orders 4, 3, 2 and 1, respectively.

5.6.1 Randomly Simulated Data

A fourth order tensor $\boldsymbol{F}'_4$ is generated randomly, whose n^4 entries follow i.i.d. standard normals. We then symmetrize $\boldsymbol{F}'_4$ to form the super-symmetric tensor $\boldsymbol{F}_4$ by averaging the related entries. The other lower order tensors $\boldsymbol{F}_3$, $\boldsymbol{F}_2$ and $\boldsymbol{F}_1$ are generated in the same manner. We then apply Algorithm 5.2.1 to get a feasible solution with its objective value denote by v, which has a guaranteed worst-case performance ratio.

For the purpose of making a comparison, we also compute an upper bound of the optimal value of (EP_S). Like in (5.2), we may let $F(\bar{\boldsymbol{x}},\bar{\boldsymbol{x}},\bar{\boldsymbol{x}},\bar{\boldsymbol{x}}) = f(\bar{\boldsymbol{x}}) = p(\boldsymbol{x})$ when $x_h = 1$, and $\boldsymbol{F} \in \mathbb{R}^{(n+1)^4}$ is super-symmetric. (EP_S) can be relaxed to

$$\begin{array}{ll} \max & F(\bar{\boldsymbol{x}},\bar{\boldsymbol{x}},\bar{\boldsymbol{x}},\bar{\boldsymbol{x}}) \\ \text{s.t.} & \|\bar{\boldsymbol{x}}\| \le \sqrt{2},\ \bar{\boldsymbol{x}} \in \mathbb{R}^{n+1}. \end{array}$$

Let $\boldsymbol{y} = \mathrm{vec}(\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^{\mathrm{T}}) \in \mathbb{R}^{(n+1)^2}$, and rewrite $\boldsymbol{F}$ as an $(n+1)^2 \times (n+1)^2$ matrix $\boldsymbol{F}'$. (EP_S) is further relaxed to

$$\begin{array}{ll} \max & F'(\boldsymbol{y},\boldsymbol{y}) = \boldsymbol{y}^{\mathrm{T}}\boldsymbol{F}'\boldsymbol{y} \\ \text{s.t.} & \|\boldsymbol{y}\| \le 2,\ \boldsymbol{y} \in \mathbb{R}^{(n+1)^2}. \end{array}$$

The optimal value of the above problem is $\bar{v} = 4\,\lambda_{\max}(\boldsymbol{F}')$, which is taken as an upper bound of $v(EP_S)$.

By Theorem 5.2.2, Algorithm 5.2.1 possesses a theoretic worst-case relative performance ratio of $2^{-10} \cdot 5! \cdot 4^{-8}(n+1)^{-1} = \Omega(1/n)$. The numerical simulation results of (EP_S) are listed in Table 5.1. Based on the observation, by comparing with the upper bound $\bar{v}$ (which might be very loose), the absolute performance ratio $\tau := v/\bar{v}$ is about $\Omega(1/\sqrt{n})$, rather than a theoretical relative ratio $\Omega(1/n)$.

Table 5.1: Numerical results (average of 10 instances) of (EP_S)

n	3	5	10	20	30	40	50	60	70
v	0.342	0.434	0.409	0.915	0.671	0.499	0.529	0.663	0.734
$\bar{v}$	10.5	16.1	26.7	51.7	74.4	97.8	121.1	143.6	167.1
$\tau\,(\%)$	3.257	2.696	1.532	1.770	0.902	0.510	0.437	0.462	0.439
$n \cdot \tau$	0.098	0.135	0.153	0.354	0.271	0.204	0.218	0.277	0.307
$\sqrt{n} \cdot \tau$	0.056	0.060	0.048	0.079	0.049	0.032	0.031	0.036	0.037

With regard to the computational efforts, we report that Algorithm 5.2.1 ran fairly fast. For instance, for $n = 70$ we were able to get a feasible solution within seconds, while computing the upper bound $\bar{v}$ costed much more computational time. For $n \geq 80$, however, our computer reported to run out of memory in the experiments, a problem purely due to the sheer size of the input data.

5.6.2 Local Improvements

The theoretical worst-case performance ratios that we have developed so far are certainly very conservative, as observed in the previous subsection. It will be desirable to design a more realistic procedure to know how good the solutions actually are. One point to note is that we can always improve the quality of the solution by applying a local improvement procedure on our heuristic solutions. In the Matlab environment, such local search procedure is readily available, e.g., the `fmincon` function in Matlab 7.7.0 (R2008b), which finds a local KKT point starting from the feasible solution that we provide. In our experiments, we find that the `fmincon` function works well at least for the low dimensional problems. In particular, for our test cases, it works quite stably up to $n = 10$.

In order to evaluate the true quality of our approximate solutions it is desirable to probe the optimal values, instead of using the loose upper bounds. For this purpose we set up the following experiments. In this set of experiments we restrict ourselves to the low dimensional cases, say $n \leq 10$. First we take the feasible approximate solution (which has an objective v) as a starting point to be followed by the local improvement procedure of the `fmincon` function to obtain a KKT solution, and denote its objective value to be v_f. Then we use a brutal force approach to randomly sample 1,000 feasible

Table 5.2: Numerical objectives of (EP_S) with local improvements for $n = 5$

Instance	1	2	3	4	5	6	7	8	9	10
v	0.35	0.47	0.08	0.40	0.17	0.13	0.07	1.78	0.32	0.53
v_f	4.16	4.85	4.24	3.99	4.28	6.49	6.46	6.42	5.14	6.84
v_f^*	4.16	4.85	4.24	3.99	4.28	6.49	6.46	6.42	5.14	6.84
$\bar{v}$	14.33	14.92	14.88	15.62	17.59	14.34	15.60	19.12	13.63	15.01

Table 5.3: Numerical objectives of (EP_S) with local improvements for $n = 10$

Instance	1	2	3	4	5	6	7	8	9	10
v	1.51	1.24	0.80	0.28	0.09	0.12	0.30	0.36	0.35	0.61
v_f	8.98	7.74	9.74	8.71	8.14	11.24	9.82	7.75	9.18	11.08
v_f^*	9.70	7.74	9.74	8.90	8.14	11.24	9.82	7.85	9.18	11.08
$\bar{v}$	26.67	24.88	28.06	28.75	27.82	26.99	26.92	27.75	27.83	27.10

solutions, followed by the same local improving `fmincon` function in Matlab. We then pick the best one as the proxy of the true optimal solution, and denote its objective value to be v_f^*. This is doable for the case $n \leq 10$ in our computational environment.

For the case $n = 5$ and $n = 10$ respectively, we generate 10 random instances of (EP_S). The solutions obtained, as described above, are shown in Table 5.2 and Table 5.3 respectively. The results are quite telling: Algorithm 5.2.1 together with `fmincon` yields near optimal solutions, at least for low dimension problems. However for problems in high dimensions, a stable local improvement procedure is a nontrivial task, interested readers are referred to a recent paper by Chen et al. [25].

Chapter 6

Polynomial Optimization with Binary Constraints

6.1 Introduction

We shift the focus from continuous optimization models in previous chapters, to discrete optimizations. In fact, a very large class of discrete optimization problems have their objectives and constraints being polynomials, e.g., the graph partition problems, the network flow problems. In particular, this chapter is concerned with the models of optimizing a polynomial function subject to binary constraints, with the objective focusing on the four types of polynomial functions mentioned in Section 2.1.1. Specifically, the models are

$$(T_B) \quad \max \quad F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d)$$
$$\text{s.t.} \quad \boldsymbol{x}^k \in \mathbb{B}^{n_k},\ k = 1, 2, \ldots, d;$$

$$(H_B) \quad \max \quad f(\boldsymbol{x})$$
$$\text{s.t.} \quad \boldsymbol{x} \in \mathbb{B}^n;$$

$$(M_B) \quad \max \quad f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s)$$
$$\text{s.t.} \quad \boldsymbol{x}^k \in \mathbb{B}^{n_k},\ k = 1, 2, \ldots, s;$$

$$(P_B) \quad \max \quad p(\boldsymbol{x})$$
$$\text{s.t.} \quad \boldsymbol{x} \in \mathbb{B}^n.$$

These four models are discussed sequentially, each one in one chapter. The latter model generalizes the former one, and each generalization has its own approach

and technique to cope with. The last model, (P_B), is indeed a very general discrete optimization model, since in principle it can be used to model the following general polynomial optimization problem in discrete values:

$$\begin{aligned} \max \quad & p(\boldsymbol{x}) \\ \text{s.t.} \quad & x_i \in \{a_1^i, a_2^i, \cdots, a_{m_i}^i\},\ i = 1, 2, \ldots, n. \end{aligned}$$

We also discuss polynomial optimizations over hypercubes as some byproducts of this chapter. They are models (T_B), (H_B), (M_B) and (P_B), i.e., the respective models (T_B), (H_B), (M_B) and (P_B) with $\mathbb{B}$ being replaced by $\bar{\mathbb{B}}$.

All the models are unfortunately NP-hard when the degree of the objective polynomial $d \geq 2$, albeit they are trivial when $d = 1$. This is because each one includes computing the matrix $\infty \mapsto 1$-norm (see e.g., [5]) as a subclass, i.e.,

$$\begin{aligned} \|\boldsymbol{F}\|_{\infty \mapsto 1} = \quad \max \quad & (\boldsymbol{x}^1)^{\mathrm{T}} \boldsymbol{F} \boldsymbol{x}^2 \\ \text{s.t.} \quad & \boldsymbol{x}^1 \in \mathbb{B}^{n_1},\ \boldsymbol{x}^2 \in \mathbb{B}^{n_2}, \end{aligned}$$

which is also the exact model of (T_B) when $d = 2$. The matrix $\infty \mapsto 1$-norm is related to so-call the matrix cut-norm, the current best polynomial-time approximation ratio for matrix $\infty \mapsto 1$-norm as well as the matrix cut-norm is $\frac{2\ln(1+\sqrt{2})}{\pi} \approx 0.56$, due to Alon and Naor [5]. Huang and Zhang [59] considered similar problems for the complex discrete variables and derived constant approximation ratios. When $d = 3$, (T_B) is a slight generalization of the model considered by Khot and Naor [63], where $\boldsymbol{F}$ is assumed to be super-symmetric (implying $n_1 = n_2 = n_3$) and square-free ($F_{ijk} = 0$ whenever two of the three indices are equal). The approximation bound of the optimal value given in [63] is $\Omega\left(\sqrt{\frac{\log n_1}{n_1}}\right)$.

For the model (H_B), its NP-hardness for $d = 2$ can also be derived by reducing to the max-cut problem, where the matrix $\boldsymbol{F}$ is the Laplacian of a given graph. In a seminar work by Goemans and Williamson [40], a polynomial-time randomized approximation algorithm is given with approximation ratio 0.878, by the well known SDP relaxation and randomization technique. The method is then generalized by Nesterov, who in [88] proved a 0.63-approximation ratio for (H_B) when the matrix $\boldsymbol{F}$ is positive semidefinite. A more generalized result is due to Charikar and Wirth [24], where an $\Omega(1/\log n)$-approximate ratio for (H_B) is proposed when the matrix $\boldsymbol{F}$ is diagonal-free. If the degree of the objective polynomial goes higher, the only approximation result in the literature is due to Khot and Naor [63] in considering homogeneous cubic polynomial,

where an $\Omega\left(\sqrt{\frac{\log n}{n}}\right)$-approximation bound is provided when the tensor $\boldsymbol{F}$ is square-free. In fact, square-free (or in the matrix case diagonal-free) is some kind of necessary condition to derive polynomial-time approximation algorithms (see e.g., [4]). Even in the quadratic case, there is no polynomial-time approximation algorithm with a positive approximation ratio for the general model (H_B) unless $P = NP$.

In this chapter we propose polynomial-time randomized approximation algorithms with provable worst-case performance ratios for all the models mentioned in the beginning, provided that the degree of the objective polynomial is fixed. Section 6.2 discusses the model (T_B). Essentially, we apply a similar approach as in Chapter 3, by relaxing the multilinear form objective to a lower order multilinear form. However the discrete natural makes the problems quite different as the continuous ones in Chapter 3, and a novel decomposition routine is proposed in order to derive the approximation bound. Section 6.3 and Section 6.4 discuss models (H_B) and (M_B), respectively. Both of them use multilinear form relaxations, armed with two different versions of link identities, in order to preserve the approximation bounds under the square-free property. General model (P_B) is discussed in Section 6.5, where the homogenization technique in Chapter 5 is modified and applied. All these approximation algorithms can be applied to polynomial optimizations over hypercubes, and we also brief the results in Section 6.5 as some byproducts. Some specific applications for the discrete models and approximation algorithms proposed in this chapter will be discussed in Section 6.6. Finally, we report our numerical experiment results in Section 6.7.

6.2 Multilinear Form with Binary Constraints

Our first discrete model in question is to maximize a multilinear function in binary variables, specifically

$$\boxed{\begin{array}{lll} (T_B) & \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\ & \text{s.t.} & \boldsymbol{x}^k \in \mathbb{B}^{n_k},\ k = 1, 2, \ldots, d, \end{array}}$$

where $n_1 \le n_2 \le \cdots \le n_d$.

This model is NP-hard when $d \ge 2$, and we shall propose polynomial-time randomized approximation algorithms with worse-case performance ratios. The case of $d = 2$ is to compute $\|\boldsymbol{F}\|_{\infty\to 1}$, whose best approximation bound is $\frac{2\ln(1+\sqrt{2})}{\pi} \approx 0.56$, due to

Alon and Naor [5]. It also servers as a basis in our subsequence analysis. When $d = 3$, Khot and Naor [63] proposed a randomized procedure to compute the optimal value of (T_B) in polynomial-time, with approximation bound $\Omega\left(\sqrt{\frac{\log n_1}{n_1}}\right)$.

Our approximation algorithm works for general degree d based on recursion, and is fairly simple. We may take any approximation algorithm for the $d = 2$ case, say the algorithm by Alon and Naor [5], as a basis. When $d = 3$, noticing that any $n_1 \times n_2 \times n_3$ third order tensor can be rewritten as an $n_1 n_2 \times n_3$ matrix by combining its first and second modes, (T_B) can be relaxed to

$$\begin{array}{ll} \max & F(\boldsymbol{X}, \boldsymbol{x}^3) \\ \text{s.t.} & \boldsymbol{X} \in \mathbb{B}^{n_1 n_2},\ \boldsymbol{x}^3 \in \mathbb{B}^{n_3}. \end{array}$$

This problem is the exact form of (T_B) when $d = 2$, which can be solved approximately with approximation ratio $\frac{2\ln(1+\sqrt{2})}{\pi}$. Denote its approximate solution to be $(\hat{\boldsymbol{X}}, \hat{\boldsymbol{x}}^3)$. The next key step is to recover $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2)$ from $\hat{\boldsymbol{X}}$. For this purpose, we introduce the following decomposition routine, which plays a fundamental role in our algorithms for binary variables, similar as DR 3.2.1, DR 3.2.2 and DR 3.3.1 in Chapter 3.

Decomposition Routine 6.2.1

- *INPUT: matrices* $\boldsymbol{M} \in \mathbb{R}^{n_1 \times n_2}$, $\boldsymbol{X} \in \mathbb{B}^{n_1 \times n_2}$.

1 *Construct*

$$\tilde{\boldsymbol{X}} = \begin{bmatrix} \boldsymbol{I}_{n_1 \times n_1} & \boldsymbol{X}/\sqrt{n_1} \\ \boldsymbol{X}^{\mathrm{T}}/\sqrt{n_1} & \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}/n_1 \end{bmatrix} \succeq 0.$$

2 *Randomly generate*

$$\begin{pmatrix} \boldsymbol{\xi} \\ \boldsymbol{\eta} \end{pmatrix} \sim \mathcal{N}(\boldsymbol{0}_{n_1+n_2}, \tilde{\boldsymbol{X}})$$

and compute $\boldsymbol{x}^1 = \operatorname{sign}(\boldsymbol{\xi})$ *and* $\boldsymbol{x}^2 = \operatorname{sign}(\boldsymbol{\eta})$, *and repeat if necessary, until* $(\boldsymbol{x}^1)^{\mathrm{T}}\boldsymbol{M}\boldsymbol{x}^2 \geq \frac{2}{\pi\sqrt{n_1}}\boldsymbol{M} \bullet \boldsymbol{X}$.

- *OUTPUT: vectors* $\boldsymbol{x}^1 \in \mathbb{B}^{n_1}$, $\boldsymbol{x}^2 \in \mathbb{B}^{n_2}$.

The complexity for DR 6.2.1 is $O(n_1 n_2)$ in each trial with expectation. Now, if we let $(\boldsymbol{M}, \boldsymbol{X}) = (F(\cdot, \cdot, \hat{\boldsymbol{x}}^3), \hat{\boldsymbol{X}})$ and apply DR 6.2.1, then we can prove that the output

$(\boldsymbol{x}^1, \boldsymbol{x}^2)$ satisfies

$$\mathrm{E}[F(\boldsymbol{x}^1, \boldsymbol{x}^2, \hat{\boldsymbol{x}}^3)] = \mathrm{E}[(\boldsymbol{x}^1)^{\mathrm{T}} \boldsymbol{M} \boldsymbol{x}^2] \geq \frac{2\boldsymbol{M} \bullet \hat{\boldsymbol{X}}}{\pi\sqrt{n_1}} = \frac{2F(\hat{\boldsymbol{X}}, \hat{\boldsymbol{x}}^3)}{\pi\sqrt{n_1}} \geq \frac{4\ln(1+\sqrt{2})}{\pi^2\sqrt{n_1}}\, v(T_B),$$

which yields an approximation ratio for $d = 3$. By a recursive procedure, this approximation algorithm is readily extended to solve (T_B) with any fixed degree d.

Theorem 6.2.1 *(T_B) admits a polynomial-time randomized approximation algorithm with approximation ratio $\tau(T_B)$, where*

$$\tau(T_B) := \left(\frac{2}{\pi}\right)^{d-1} \ln\left(1+\sqrt{2}\right) \left(\prod_{k=1}^{d-2} n_k\right)^{-\frac{1}{2}} = \Omega\left(\left(\prod_{k=1}^{d-2} n_k\right)^{-\frac{1}{2}}\right).$$

Proof. The proof is based on mathematical induction on the degree d. For the case of $d = 2$, it is exactly the algorithm by Alon and Naor [5]. For general $d \geq 3$, let $\boldsymbol{X} = \boldsymbol{x}^1(\boldsymbol{x}^d)^{\mathrm{T}}$ and (T_B) is then relaxed to

$$\begin{array}{lll} (\tilde{T}_B) & \max & F(\boldsymbol{X}, \boldsymbol{x}^2, \boldsymbol{x}^3 \cdots, \boldsymbol{x}^{d-1}) \\ & \text{s.t.} & \boldsymbol{X} \in \mathbb{B}^{n_1 n_d}, \\ & & \boldsymbol{x}^k \in \mathbb{B}^{n_k},\ k = 2, 3, \ldots, d-1, \end{array}$$

where we treat $\boldsymbol{X}$ as an $n_1 n_d$-dimensional vector, and $\boldsymbol{F} \in \mathbb{R}^{n_1 n_d \times n_2 \times n_3 \times \cdots \times n_{d-1}}$ as a $(d-1)$-th order tensor. Observe that $(\tilde{T}_B)$ is the exact form of (T_B) in degree $d-1$, and so by induction we can find $\hat{\boldsymbol{X}} \in \mathbb{B}^{n_1 n_d}$ and $\hat{\boldsymbol{x}}^k \in \mathbb{B}^{n_k}$ $(k = 2, 3, \ldots, d-1)$ in polynomial-time, such that

$$\begin{array}{rcl} F\left(\hat{\boldsymbol{X}}, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \ldots, \hat{\boldsymbol{x}}^{d-1}\right) & \geq & (2/\pi)^{d-2} \ln\left(1+\sqrt{2}\right) \left(\prod_{k=2}^{d-2} n_k\right)^{-\frac{1}{2}} v(\tilde{T}_B) \\ & \geq & (2/\pi)^{d-2} \ln\left(1+\sqrt{2}\right) \left(\prod_{k=2}^{d-2} n_k\right)^{-\frac{1}{2}} v(T_B). \end{array}$$

Rewrite $\hat{\boldsymbol{X}}$ as an $n_1 \times n_d$ matrix, construct

$$\tilde{\boldsymbol{X}} = \begin{bmatrix} \boldsymbol{I}_{n_1 \times n_1} & \hat{\boldsymbol{X}}/\sqrt{n_1} \\ \hat{\boldsymbol{X}}^{\mathrm{T}}/\sqrt{n_1} & \hat{\boldsymbol{X}}^{\mathrm{T}}\hat{\boldsymbol{X}}/n_1 \end{bmatrix} \succeq 0,$$

as in DR 6.2.1, and randomly generate

$$\begin{pmatrix} \boldsymbol{\xi} \\ \boldsymbol{\eta} \end{pmatrix} \sim \mathcal{N}(\boldsymbol{0}_{n_1+n_d}, \tilde{\boldsymbol{X}}).$$

Let $\hat{\boldsymbol{x}}^1 = \mathrm{sign}(\boldsymbol{\xi})$ and $\hat{\boldsymbol{x}}^d = \mathrm{sign}(\boldsymbol{\eta})$. Noticing that the diagonal components of $\tilde{\boldsymbol{X}}$ are all ones, by an expectation identity in Goemans and Williamson [40], it follows that

$$\mathrm{E}\left[\hat{x}_i^1 \hat{x}_j^d\right] = \frac{2}{\pi} \arcsin \frac{\hat{X}_{ij}}{\sqrt{n_1}} = \frac{2}{\pi} \hat{X}_{ij} \arcsin \frac{1}{\sqrt{n_1}}, \quad \forall\, 1 \leq i \leq n_1,\ 1 \leq j \leq n_d,$$

where the last equality is due to $|\hat{X}_{ij}| = 1$. Let matrix $\hat{\boldsymbol{Q}} = F(\cdot, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1}, \cdot)$, and we have

$$\begin{aligned}
\mathrm{E}\left[F\left(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d\right)\right] &= \mathrm{E}\left[\sum_{1\le i\le n_1, 1\le j\le n_d} \hat{x}_i^1 \hat{Q}_{ij} \hat{x}_j^d\right] \\
&= \sum_{1\le i\le n_1, 1\le j\le n_d} \hat{Q}_{ij}\, \mathrm{E}\left[\hat{x}_i^1 \hat{x}_j^d\right] \\
&= \sum_{1\le i\le n_1, 1\le j\le n_d} \hat{Q}_{ij} \frac{2}{\pi} \hat{X}_{ij} \arcsin\frac{1}{\sqrt{n_1}} \\
&= \frac{2}{\pi}\arcsin\frac{1}{\sqrt{n_1}} \sum_{1\le i\le n_1, 1\le j\le n_d} \hat{Q}_{ij}\hat{X}_{ij} \\
&= \frac{2}{\pi}\arcsin\frac{1}{\sqrt{n_1}}\, F\left(\hat{\boldsymbol{X}}, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1}\right) \qquad (6.1) \\
&\ge \frac{2}{\pi\sqrt{n_1}} \left(\frac{2}{\pi}\right)^{d-2} \ln\left(1+\sqrt{2}\right) \left(\prod_{k=2}^{d-2} n_k\right)^{-\frac{1}{2}} v(T_B) \\
&= \left(\frac{2}{\pi}\right)^{d-1} \ln\left(1+\sqrt{2}\right) \left(\prod_{k=1}^{d-2} n_k\right)^{-\frac{1}{2}} v(T_B).
\end{aligned}$$

Thus $\hat{\boldsymbol{x}}^1$ and $\hat{\boldsymbol{x}}^d$ can be found by a randomization process, which concludes the induction step. □

To summarize this section, the algorithm for solving general model (T_B) is attached below. This algorithm is similar to Algorithm 3.2.3, with major differences lying in different decomposition routines and the computability for the case of $d = 2$.

Algorithm 6.2.2

- *INPUT: a d-th order tensor* $\boldsymbol{F} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$ *with* $n_1 \le n_2 \le \cdots \le n_d$.

1 *Rewrite* $\boldsymbol{F}$ *as a* $(d-1)$*-th order tensor* $\boldsymbol{F}' \in \mathbb{R}^{n_2\times n_3\times\cdots\times n_{d-1}\times n_d n_1}$ *by combing its first and last modes into one, and placing it in the last mode of* $\boldsymbol{F}'$*, i.e.,*

$$F_{i_1,i_2,\cdots,i_d} = F'_{i_2,i_3,\cdots,i_{d-1},(i_1-1)n_d+i_d} \quad \forall\, 1\le i_1\le n_1, 1\le i_2\le n_2, \cdots, 1\le i_d\le n_d.$$

2 *For* (T_B) *with the* $(d-1)$*-th order tensor* $\boldsymbol{F}'$*: if* $d-1=2$*, then apply SDP relaxation and randomization procedure (Alon and Naor [5]) to obtain an approximate solution* $(\hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^{1,d})$*; otherwise obtain a solution* $(\hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1}, \hat{\boldsymbol{x}}^{1,d})$ *by recursion.*

3 *Compute a matrix* $\boldsymbol{M}' = F(\cdot, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1}, \cdot)$ *and rewrite the vector* $\hat{\boldsymbol{x}}^{1,d}$ *as a matrix* $\hat{\boldsymbol{X}} \in \mathbb{B}^{n_1 \times n_d}$.

4 *Apply DR 6.2.1, with input* $(\boldsymbol{M}', \hat{\boldsymbol{X}}) = (\boldsymbol{M}, \boldsymbol{X})$ *and output* $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^d) = (\boldsymbol{x}^1, \boldsymbol{x}^2)$.

- *OUTPUT: a feasible solution* $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$.

6.3 Homogeneous Form with Binary Constraints

We now consider the model of maximizing a general homogeneous polynomial function in binary variables, i.e.,

$$\boxed{\begin{array}{lll} (H_B) & \max & f(\boldsymbol{x}) \\ & \text{s.t.} & \boldsymbol{x} \in \mathbb{B}^n, \end{array}}$$

where $f(\boldsymbol{x})$ is a d-th degree homogenous polynomial with associated super-symmetric tensor $\boldsymbol{F} \in \mathbb{R}^{n^d}$.

When $d = 2$, an $\Omega(1/\log n)$-approximate ratio for (H_B) is proposed when the matrix $\boldsymbol{F}$ is diagonal-free, by Charikar and Wirth [24]; When $d = 3$, an $\Omega\left(\sqrt{\frac{\log n}{n}}\right)$-approximation bound for the optimal value of (H_B) is provided if the tensor $\boldsymbol{F}$ is square-free, by Khot and Naor [63]. We remark that the square-free property is a necessary condition to derive the approximation ratios. Even in the quadratic and cubic cases for (H_B), there is no polynomial-time approximation algorithm with a positive approximation ratio unless $P = NP$ (see [4]).

As before, we propose polynomial-time randomized approximation algorithms of (H_B) for any fixed degree d. Like the model (T_S), the key link from multilinear form $F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d)$ to the homogeneous polynomial $f(\boldsymbol{x})$ is Lemma 4.2.1. The approximation ratios for (H_B) hold under the square-free condition. This is because under such conditions, the decision variables are actually in the multilinear form. Hence, one can replace any point in the hypercube $(\bar{\mathbb{B}}^n)$ by one of its vertices $(\mathbb{B}^n)$ without decreasing its objective value, due to the linearity. Before presenting our main results in this section, we first study a property of the square-free polynomial in binary variables, which will be used frequently in this chapter and the next chapter (Chapter 7).

Lemma 6.3.1 *If polynomial function* $p(\boldsymbol{x})$ *is square-free and* $\boldsymbol{z} \in \bar{\mathbb{B}}^n$, *then* $\hat{\boldsymbol{x}} \in \mathbb{B}^n$ *and* $\tilde{\boldsymbol{x}} \in \mathbb{B}^n$ *can be found in polynomial-time, such that* $p(\hat{\boldsymbol{x}}) \leq p(\boldsymbol{z}) \leq p(\tilde{\boldsymbol{x}})$.

Proof. Since $p(\boldsymbol{x})$ is square-free, by fixing $x_2, x_3, \cdots, x_n$ as constants and taking x_1 as an independent variable, we may write

$$p(\boldsymbol{x}) = g_1(x_2, x_3, \cdots, x_n) + x_1 g_2(x_2, x_3, \cdots, x_n).$$

Let

$$\hat{x}_1 = \begin{cases} -1 & g_2(z_2, z_3, \cdots, z_n) \geq 0, \\ 1 & g_2(z_2, z_3, \cdots, z_n) < 0. \end{cases}$$

Then

$$p\left((\hat{x}_1, z_2, z_3, \cdots, z_n)^{\mathrm{T}}\right) \leq p(\boldsymbol{z}).$$

Repeat the same procedures for $z_2, z_3, \cdots, z_n$, and let them be replaced by binary scales $\hat{x}_2, \hat{x}_3, \cdots, \hat{x}_n$, respectively. Then $\hat{\boldsymbol{x}} = (\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_n)^{\mathrm{T}} \in \mathbb{B}^n$ satisfies $p(\hat{\boldsymbol{x}}) \leq p(\boldsymbol{z})$. Using a similar procedure, we may find $\tilde{\boldsymbol{x}} \in \mathbb{B}^n$ with $p(\tilde{\boldsymbol{x}}) \geq p(\boldsymbol{z})$. □

Lemma 6.3.1 actually proposes a polynomial-time procedure in finding a point in $\mathbb{B}^n$ to replace a point in $\bar{\mathbb{B}}^n$, without decreasing (or increasing) its function value. Now, armed with Lemma 6.3.1 and the link Lemma 4.2.1, we present the main results in this section.

Theorem 6.3.2 *If $f(\boldsymbol{x})$ is square-free and $d \geq 3$ is odd, then (H_B) admits a polynomial-time randomized approximation algorithm with approximation ratio $\tau(H_B)$, where*

$$\tau(H_B) := \left(\frac{2}{\pi}\right)^{d-1} \ln\left(1+\sqrt{2}\right) d!\, d^{-d} n^{-\frac{d-2}{2}} = \Omega\left(n^{-\frac{d-2}{2}}\right).$$

Proof. Let $f(\boldsymbol{x}) = F(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x}}_{d})$ with $\boldsymbol{F}$ being super-symmetric, and (H_B) can be relaxed to

$$\begin{array}{lll} (\tilde{H}_B) & \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\ & \text{s.t.} & \boldsymbol{x}^k \in \mathbb{B}^n,\ k = 1, 2, \ldots, d. \end{array}$$

By Theorem 6.2.1 we are able to find a set of binary vectors $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$ in polynomial-time, such that

$$F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) \geq \left(\frac{2}{\pi}\right)^{d-1} \ln(1+\sqrt{2})\, n^{-\frac{d-2}{2}} v(\tilde{H}_B) \geq \left(\frac{2}{\pi}\right)^{d-1} \ln(1+\sqrt{2})\, n^{-\frac{d-2}{2}} v(H_B).$$

When d is odd, let $\xi_1, \xi_2, \cdots, \xi_d$ be i.i.d. random variables, each taking values 1 and -1 with equal probability 1/2. Then by Lemma 4.2.1 it follows that

$$d! F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) = \mathrm{E}\left[\prod_{i=1}^{d} \xi_i f\left(\sum_{k=1}^{d} \xi_k \hat{\boldsymbol{x}}^k\right)\right] = \mathrm{E}\left[f\left(\sum_{k=1}^{d}\left(\prod_{i \neq k} \xi_i\right) \hat{\boldsymbol{x}}^k\right)\right].$$

Thus we may find a binary vector $\boldsymbol{\beta} \in \mathbb{B}^d$, such that

$$f\left(\sum_{k=1}^{d}\left(\prod_{i\neq k}\beta_i\right)\hat{\boldsymbol{x}}^k\right) \geq d!F(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\ldots,\hat{\boldsymbol{x}}^d) \geq \left(\frac{2}{\pi}\right)^{d-1}\ln\left(1+\sqrt{2}\right)d!\,n^{-\frac{d-2}{2}}v(H_B).$$

Now we notice that $\frac{1}{d}\sum_{k=1}^{d}\left(\prod_{i\neq k}\beta_i\right)\hat{\boldsymbol{x}}^k \in \bar{\mathbb{B}}^n$, because for all $1 \leq j \leq n$,

$$\left|\left(\frac{1}{d}\sum_{k=1}^{d}\left(\prod_{i\neq k}\beta_i\right)\hat{\boldsymbol{x}}^k\right)_j\right| = \frac{1}{d}\left|\sum_{k=1}^{d}\left(\prod_{i\neq k}\beta_i\right)\hat{x}_j^k\right| \leq \frac{1}{d}\sum_{k=1}^{d}\left|\left(\prod_{i\neq k}\beta_i\right)\hat{x}_j^k\right| = 1. \quad (6.2)$$

Since $f(\boldsymbol{x})$ is square-free, by Lemma 6.3.1 we are able to find $\tilde{\boldsymbol{x}} \in \mathbb{B}^n$ in polynomial-time, such that

$$f(\tilde{\boldsymbol{x}}) \geq f\left(\frac{1}{d}\sum_{k=1}^{d}\left(\prod_{i\neq k}\beta_i\right)\hat{\boldsymbol{x}}^k\right) = d^{-d}f\left(\sum_{k=1}^{d}\left(\prod_{i\neq k}\beta_i\right)\hat{\boldsymbol{x}}^k\right) \geq \tau(H_B)\,v(H_B).$$

□

Theorem 6.3.3 *If $f(\boldsymbol{x})$ is square-free and $d \geq 4$ is even, then (H_B) admits a polynomial-time randomized approximation algorithm with relative approximation ratio $\tau(H_B)$.*

Proof. Like in the proof of Theorem 6.3.2, by relaxing (H_B) to $(\tilde{H}_B)$, we are able to find a set of binary vectors $(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\cdots,\hat{\boldsymbol{x}}^d)$ with

$$F(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\cdots,\hat{\boldsymbol{x}}^d) \geq \left(\frac{2}{\pi}\right)^{d-1}\ln\left(1+\sqrt{2}\right)n^{-\frac{d-2}{2}}v(\tilde{H}_B).$$

Besides, we observe that $v(H_B) \leq v(\tilde{H}_B)$ and $\underline{v}(H_B) \geq \underline{v}(\tilde{H}_B) = -v(\tilde{H}_B)$. Therefore

$$2\,v(\tilde{H}_B) \geq v(H_B) - \underline{v}(H_B).$$

Let $\xi_1,\xi_2,\cdots,\xi_d$ be i.i.d. random variables, each taking values 1 and -1 with equal probability 1/2. Use a similar argument of (6.2), we have $\frac{1}{d}\sum_{k=1}^{d}\xi_k\hat{\boldsymbol{x}}^k \in \bar{\mathbb{B}}^n$. Then by Lemma 6.3.1, there exists $\hat{\boldsymbol{x}} \in \mathbb{B}^n$ such that

$$f\left(\frac{1}{d}\sum_{k=1}^{d}\xi_k\hat{\boldsymbol{x}}^k\right) \geq f(\hat{\boldsymbol{x}}) \geq \underline{v}(H_B).$$

Applying Lemma 4.2.1 and we have

$$\begin{aligned}
& \frac{1}{2}\mathrm{E}\left[f\left(\frac{1}{d}\sum_{k=1}^{d}\xi_k\hat{\boldsymbol{x}}^k\right)-\underline{v}(H_B)\,\middle|\,\prod_{i=1}^{d}\xi_i=1\right] \\
\geq\ & \frac{1}{2}\mathrm{E}\left[f\left(\frac{1}{d}\sum_{k=1}^{d}\xi_k\hat{\boldsymbol{x}}^k\right)-\underline{v}(H_B)\,\middle|\,\prod_{i=1}^{d}\xi_i=1\right] \\
& -\frac{1}{2}\mathrm{E}\left[f\left(\frac{1}{d}\sum_{k=1}^{d}\xi_k\hat{\boldsymbol{x}}^k\right)-\underline{v}(H_B)\,\middle|\,\prod_{i=1}^{d}\xi_i=-1\right] \\
=\ & \mathrm{E}\left[\prod_{i=1}^{d}\xi_i\left(f\left(\frac{1}{d}\sum_{k=1}^{d}\xi_k\hat{\boldsymbol{x}}^k\right)-\underline{v}(H_B)\right)\right] \\
=\ & d^{-d}\mathrm{E}\left[\prod_{i=1}^{d}\xi_i f\left(\sum_{k=1}^{d}\xi_k\hat{\boldsymbol{x}}^k\right)\right]-\underline{v}(H_B)\,\mathrm{E}\left[\prod_{i=1}^{d}\xi_i\right] \\
=\ & d^{-d}d!F(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\ldots,\hat{\boldsymbol{x}}^d)\geq\tau(H_B)\,v(\tilde{H}_B)\geq(\tau(H_B)/2)\,(v(H_B)-\underline{v}(H_B))\,.
\end{aligned}$$

Thus we may find a binary vector $\boldsymbol{\beta}\in\mathbb{B}^d$ with $\prod_{i=1}^{d}\beta_i=1$, such that

$$f\left(\frac{1}{d}\sum_{k=1}^{d}\beta_k\hat{\boldsymbol{x}}^k\right)-\underline{v}(H_B)\geq\tau(H_B)\,(v(H_B)-\underline{v}(H_B))\,.$$

Noticing that $\frac{1}{d}\sum_{k=1}^{d}\beta_k\hat{\boldsymbol{x}}^k\in\bar{\mathbb{B}}^n$ and applying Lemma 6.3.1, by the square-free property of $f(\boldsymbol{x})$, we are able to find $\tilde{\boldsymbol{x}}\in\mathbb{B}^n$ with

$$f(\tilde{\boldsymbol{x}})-\underline{v}(H_B)\geq f\left(\frac{1}{d}\sum_{k=1}^{d}\beta_k\hat{\boldsymbol{x}}^k\right)-\underline{v}(H_B)\geq\tau(H_B)\,(v(H_B)-\underline{v}(H_B))\,.$$

□

To conclude this section, we summarize the algorithm for approximately solving (H_B) below (no matter d is odd or even).

Algorithm 6.3.1

- *INPUT: a d-th order super-symmetric square-free tensor* $\boldsymbol{F}\in\mathbb{R}^{n^d}$.

1 *Apply Algorithm 6.2.2 to solve the problem*

$$\begin{aligned}
\max\quad & F(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^d) \\
\text{s.t.}\quad & \boldsymbol{x}^k\in\mathbb{B}^n,\ k=1,2,\ldots,d
\end{aligned}$$

approximately, with input $\boldsymbol{F}$ *and output* $(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\cdots,\hat{\boldsymbol{x}}^d)$.

2 *Compute* $\hat{\boldsymbol{x}} = \arg\max\left\{f\left(\frac{1}{d}\sum_{k=1}^{d}\xi_k\hat{\boldsymbol{x}}^k\right), \boldsymbol{\xi}\in\mathbb{B}^d\right\}$.

3 *Apply the procedure in Lemma 6.3.1, with input* $\hat{\boldsymbol{x}}\in\mathbb{B}^n$ *and polynomial function* $f(\boldsymbol{x})$, *and output* $\tilde{\boldsymbol{x}}\in\mathbb{B}^n$ *satisfying* $f(\tilde{\boldsymbol{x}})\geq f(\hat{\boldsymbol{x}})$.

- *OUTPUT: a feasible solution* $\tilde{\boldsymbol{x}}\in\mathbb{B}^n$.

6.4 Mixed Form with Binary Constraints

We further move on to consider the mixed form of discrete polynomial optimization model

$$\boxed{\begin{array}{lll}(M_B) & \max & f(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^s)\\ & \text{s.t.} & \boldsymbol{x}^k\in\mathbb{B}^{n_k},\ k=1,2,\ldots,s,\end{array}}$$

where associated with function f is a tensor $\boldsymbol{F}\in\mathbb{R}^{{n_1}^{d_1}\times{n_2}^{d_2}\times\cdots\times{n_s}^{d_s}}$ with partial symmetric property, $n_1\leq n_2\leq\cdots\leq n_s$, and $d=d_1+d_2+\cdots+d_s$ is deemed as a fixed constant. This model is a generalization of (T_B) in Section 6.2 and (H_B) in Section 6.3, making the model applicable to a wider range of practical problems.

Here again we focus on polynomial-time approximation algorithms. Similar as the approach in dealing with (H_B), we relax the objective function $f(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^s)$ of (M_B) to a multilinear function, which leads to (T_B). After solving (T_B) approximately by Theorem 6.2.1, we are able to adjust the solutions one by one, using Lemma 4.4.3. The following approximation results are presented, which are comparable to that in Section 6.3.

Theorem 6.4.1 *If* $f(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^s)$ *is square-free in each* $\boldsymbol{x}^k$ $(k=1,2,\ldots,s)$, $d\geq 3$ *and one of* d_k $(k=1,2,\ldots,s)$ *is odd, then* (M_B) *admits a polynomial-time randomized approximation algorithm with approximation ratio* $\tau(M_B)$, *where*

$$\begin{aligned}\tau(M_B) &:= \tilde{\tau}(M_S)\left(\frac{2}{\pi}\right)^{d-1}\ln\left(1+\sqrt{2}\right)\prod_{k=1}^{s}\frac{d_k!}{{d_k}^{d_k}} = \Omega\left(\tilde{\tau}(M_S)\right)\\ &= \begin{cases}\left(\frac{2}{\pi}\right)^{d-1}\ln\left(1+\sqrt{2}\right)\left(\prod_{k=1}^{s}\frac{d_k!}{{d_k}^{d_k}}\right)\left(\frac{\prod_{k=1}^{s-1}{n_k}^{d_k}}{n_{s-1}}\right)^{-\frac{1}{2}} & d_s=1,\\ \left(\frac{2}{\pi}\right)^{d-1}\ln\left(1+\sqrt{2}\right)\left(\prod_{k=1}^{s}\frac{d_k!}{{d_k}^{d_k}}\right)\left(\frac{\prod_{k=1}^{s}{n_k}^{d_k}}{{n_s}^2}\right)^{-\frac{1}{2}} & d_s\geq 2.\end{cases}\end{aligned}$$

Proof. Like in the proof of Theorem 6.3.2, by relaxing (M_B) to (T_B), we are able to find a set of binary vectors $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$ with

$$F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) \geq \tau(M_B) \left(\prod_{k=1}^{s} \frac{{d_k}^{d_k}}{d_k!} \right) v(M_B).$$

Let $\boldsymbol{\xi} \triangleq (\xi_1, \xi_2, \cdots, \xi_d)^{\mathrm{T}}$, whose components are i.i.d. random variables, taking values 1 and -1 with equal probability $1/2$. Similar as (4.7), we denote

$$\hat{\boldsymbol{x}}_\xi^1 = \sum_{k=1}^{d_1} \xi_k \hat{\boldsymbol{x}}^k, \ \hat{\boldsymbol{x}}_\xi^2 = \sum_{k=d_1+1}^{d_1+d_2} \xi_k \hat{\boldsymbol{x}}^k, \ \cdots, \ \hat{\boldsymbol{x}}_\xi^s = \sum_{k=d_1+d_2+\cdots+d_{s-1}+1}^{d} \xi_k \hat{\boldsymbol{x}}^k.$$

Without loss of generality, we assume d_1 to be odd. By applying Lemma 4.4.3 we have

$$\prod_{k=1}^{s} d_k! F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) = \mathrm{E}\left[\prod_{i=1}^{d} \xi_i f\left(\hat{\boldsymbol{x}}_\xi^1, \hat{\boldsymbol{x}}_\xi^2, \cdots, \hat{\boldsymbol{x}}_\xi^s \right) \right] = \mathrm{E}\left[f\left(\prod_{i=1}^{d} \xi_i \hat{\boldsymbol{x}}_\xi^1, \hat{\boldsymbol{x}}_\xi^2, \cdots, \hat{\boldsymbol{x}}_\xi^s \right) \right].$$

Therefore we are able to find a binary vector $\boldsymbol{\beta} \in \mathbb{B}^d$, such that

$$f\left(\prod_{i=1}^{d} \beta_i \frac{\hat{\boldsymbol{x}}_\beta^1}{d_1}, \frac{\hat{\boldsymbol{x}}_\beta^2}{d_2}, \cdots, \frac{\hat{\boldsymbol{x}}_\beta^s}{d_s} \right) \geq \prod_{k=1}^{s} d_k! {d_k}^{-d_k} F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) \geq \tau(M_B)\, v(M_B).$$

Similar as (6.2), it is not hard to verify that $\prod_{i=1}^{d} \beta_i \hat{\boldsymbol{x}}_\beta^1 / d_1 \in \bar{\mathbb{B}}^{n_1}$, and $\hat{\boldsymbol{x}}_\beta^k / d_k \in \bar{\mathbb{B}}^{n_k}$ for $k = 2, 3, \ldots, s$. By the square-free property of the function f and applying Lemma 6.3.1, we are able to find a set of binary vectors $(\tilde{\boldsymbol{x}}^1, \tilde{\boldsymbol{x}}^2, \cdots, \tilde{\boldsymbol{x}}^s)$ in polynomial-time, such that

$$f(\tilde{\boldsymbol{x}}^1, \tilde{\boldsymbol{x}}^2, \cdots, \tilde{\boldsymbol{x}}^s) \geq f\left(\prod_{i=1}^{d} \beta_i \frac{\hat{\boldsymbol{x}}_\beta^1}{d_1}, \frac{\hat{\boldsymbol{x}}_\beta^2}{d_2}, \cdots, \frac{\hat{\boldsymbol{x}}_\beta^s}{d_s} \right) \geq \tau(M_B)\, v(M_B).$$

□

Theorem 6.4.2 *If $f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s)$ is square-free in each $\boldsymbol{x}^k$ $(k = 1, 2, \ldots, s)$, $d \geq 4$ and all d_k $(k = 1, 2, \ldots, s)$ are even, then (M_B) admits a polynomial-time randomized approximation algorithm with relative approximation ratio $\tau(M_B)$.*

Proof. The proof is analogous to that of Theorem 6.3.3. The main differences are: (i) we use Lemma 4.4.3 instead of invoking Lemma 4.2.1 directly; and (ii) we use $f\left(\frac{1}{d_1} \hat{\boldsymbol{x}}_\xi^1, \frac{1}{d_2} \hat{\boldsymbol{x}}_\xi^2, \cdots, \frac{1}{d_s} \hat{\boldsymbol{x}}_\xi^s \right)$ instead of $f\left(\frac{1}{d} \sum_{k=1}^{d} \xi_k \hat{\boldsymbol{x}}^k \right)$ during the randomization process.

□

6.5 Polynomial with Binary Constraints

Finally, we consider binary integer programming model to the optimization on a generic (inhomogeneous) polynomial function, i.e.,

$$\boxed{\begin{array}{lll}(P_B) & \max & p(\boldsymbol{x}) \\ & \text{s.t.} & \boldsymbol{x} \in \mathbb{B}^n.\end{array}}$$

Extending the approximation algorithms and the corresponding analysis for *homogeneous* polynomial optimization to general *inhomogeneous* polynomials is not straightforward. Technically it is also a way to get around the square-free property, which is a requirement for all the homogeneous polynomial optimizations discussed in previous sections. The analysis here, is similar as that in Chapter 5, to directly deal with *homogenization*. An important observation here is that $p(\boldsymbol{x})$ can always be rewritten as a square-free polynomial, since we have $x_i^2 = 1$ for $i = 1, 2, \ldots, n$, which allows us to reduce the power of x_i to 0 or 1 in each monomial of $p(\boldsymbol{x})$. We now propose the following algorithm for approximately solving (P_B).

Algorithm 6.5.1

- *INPUT: an n-dimensional d-th degree polynomial function $p(\boldsymbol{x})$.*

1 *Rewrite $p(\boldsymbol{x})$ as a square-free polynomial function $p_0(\boldsymbol{x})$, and then rewrite $p_0(\boldsymbol{x}) - p_0(\mathbf{0}) = F(\underbrace{\bar{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \cdots, \bar{\boldsymbol{x}}}_{d})$ when $x_h = 1$ as in (5.2), with $\boldsymbol{F}$ being an $(n+1)$-dimensional d-th order super-symmetric tensor.*

2 *Apply Algorithm 6.2.2 to solve the problem*

$$\begin{array}{ll}\max & F(\bar{\boldsymbol{x}}^1, \bar{\boldsymbol{x}}^2, \cdots, \bar{\boldsymbol{x}}^d) \\ \text{s.t.} & \bar{\boldsymbol{x}}^k \in \mathbb{B}^{n+1},\ k = 1, 2, \ldots, d\end{array}$$

approximately, with input $\boldsymbol{F}$ and output $(\bar{\boldsymbol{u}}^1, \bar{\boldsymbol{u}}^2, \cdots, \bar{\boldsymbol{u}}^d)$.

3 *Compute $(\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \cdots, \bar{\boldsymbol{z}}^d) = \arg\max \left\{ F\left(\binom{\xi_1 \boldsymbol{u}^{1/d}}{1}, \binom{\xi_2 \boldsymbol{u}^{2/d}}{1}, \cdots, \binom{\xi_d \boldsymbol{u}^{d/d}}{1} \right), \boldsymbol{\xi} \in \mathbb{B}^d \right\}$.*

4 *Compute $\boldsymbol{z} = \arg\max \left\{ p_0(\mathbf{0});\ p_0\left(\boldsymbol{z}(\beta)/z_h(\beta)\right), \boldsymbol{\beta} \in \mathbb{B}^d \text{ and } \beta_1 = \prod_{k=2}^{d} \beta_k = 1 \right\}$, with $\bar{\boldsymbol{z}}(\beta) = \beta_1(d+1)\bar{\boldsymbol{z}}^1 + \sum_{k=2}^{d} \beta_k \bar{\boldsymbol{z}}^k$.*

5 *Apply the procedure in Lemma 6.3.1, with input* $\boldsymbol{z} \in \bar{\mathbb{B}}^n$ *and polynomial function* $p_0(\boldsymbol{x})$, *and output* $\boldsymbol{y} \in \mathbb{B}^n$ *satisfying* $p_0(\boldsymbol{y}) \geq p_0(\boldsymbol{z})$.

- *OUTPUT: a feasible solution* $\boldsymbol{y} \in \mathbb{B}^n$.

Before presenting the main result and analyzing Algorithm 6.5.1, we first study another property of the square-free polynomial. Namely, the overall average of the function values on the support set $\mathbb{B}^n$ is zero, and this plays an important role in analyzing the algorithm for (P_B).

Lemma 6.5.1 *If the polynomial function* $p(\boldsymbol{x})$ *in* (P_B) : $\max_{\boldsymbol{x}\in\mathbb{B}^n} p(\boldsymbol{x})$ *is square-free and has no constant term, then* $\underline{v}(P_B) \leq 0 \leq v(P_B)$, *and a binary vector* $\tilde{\boldsymbol{x}} \in \mathbb{B}^n$ *can be found in polynomial-time with* $p(\tilde{\boldsymbol{x}}) \geq 0$.

Proof. Let $\xi_1, \xi_2, \cdots, \xi_n$ be i.i.d. random variables, each taking values 1 and -1 with equal probability 1/2. For any monomial $F_{i_1 i_2 \ldots i_k} x_{i_1} x_{i_2} \cdots x_{i_k}$ with degree k $(1 \leq k \leq d)$ of $p(\boldsymbol{x})$, by the square-free property, it follows that

$$\mathrm{E}[F_{i_1 i_2 \ldots i_k} \xi_{i_1} \xi_{i_2} \cdots \xi_{i_k}] = F_{i_1 i_2 \cdots i_k} \mathrm{E}[\xi_{i_1}] \mathrm{E}[\xi_{i_2}] \cdots \mathrm{E}[\xi_{i_k}] = 0.$$

This implies $\mathrm{E}[p(\boldsymbol{\xi})] = 0$, and consequently $\underline{v}(P_B) \leq 0 \leq v(P_B)$. By a randomization process, a binary vector $\tilde{\boldsymbol{x}} \in \mathbb{B}^n$ can be found in polynomial-time with $p(\tilde{\boldsymbol{x}}) \geq 0$. □

We remark that the second part of Lemma 6.5.1 can also be proven by conducting the procedure in Lemma 6.3.1 with the input vector $\mathbf{0} \in \bar{\mathbb{B}}^n$, since $p(\mathbf{0}) = 0$. Therefore, finding a binary vector $\tilde{\boldsymbol{x}} \in \mathbb{B}^n$ with $p(\tilde{\boldsymbol{x}}) \geq 0$ can be done by either a randomized process (Lemma 6.5.1) or a deterministic process (Lemma 6.3.1). We now present the main result in this section.

Theorem 6.5.2 (P_B) *admits a polynomial-time randomized approximation algorithm with relative approximation ratio* $\tau(P_B)$, *where*

$$\tau(P_B) := \frac{\ln\left(1+\sqrt{2}\right)}{2(1+e)\pi^{d-1}} (d+1)!\, d^{-2d} (n+1)^{-\frac{d-2}{2}} = \Omega\left(n^{-\frac{d-2}{2}}\right).$$

Proof. The main idea of the proof is quite similar as that of Theorem 5.2.2. However the discrete nature of the problem as well as the non-convex feasible region requires us to be more careful dealing with the specific details. As we are working with relative

approximation ratio, by Step 1 of Algorithm 6.5.1, we may assume that $p(\boldsymbol{x})$ is square-free and has no constant term. Then by homogenization as (5.2)

$$p(\boldsymbol{x}) = \tilde{F}\Big(\underbrace{\begin{pmatrix}\boldsymbol{x}\\ x_h\end{pmatrix}, \begin{pmatrix}\boldsymbol{x}\\ x_h\end{pmatrix}, \cdots, \begin{pmatrix}\boldsymbol{x}\\ x_h\end{pmatrix}}_{d}\Big) = F(\underbrace{\bar{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \cdots, \bar{\boldsymbol{x}}}_{d}) = f(\bar{\boldsymbol{x}}),$$

where $f(\bar{\boldsymbol{x}}) = p(\boldsymbol{x})$ if $x_h = 1$, and $f(\bar{\boldsymbol{x}})$ is an $(n+1)$-dimensional homogeneous polynomial function with associated super-symmetric tensor $\boldsymbol{F} \in \mathbb{R}^{(n+1)^d}$ whose last component is 0. (P_B) is then equivalent to

$$\begin{array}{ll}\max & f(\bar{\boldsymbol{x}})\\ \text{s.t.} & \bar{\boldsymbol{x}} = \begin{pmatrix}\boldsymbol{x}\\ x_h\end{pmatrix},\ \boldsymbol{x} \in \mathbb{B}^n,\ x_h = 1,\end{array}$$

which can be relaxed to an instance of (T_B) as follows

$$\begin{array}{lll}(\tilde{P}_B) & \max & F(\bar{\boldsymbol{x}}^1, \bar{\boldsymbol{x}}^2, \cdots, \bar{\boldsymbol{x}}^d)\\ & \text{s.t.} & \bar{\boldsymbol{x}}^k \in \mathbb{B}^{n+1},\ k = 1, 2, \ldots, d.\end{array}$$

Let $(\bar{\boldsymbol{u}}^1, \bar{\boldsymbol{u}}^2, \cdots, \bar{\boldsymbol{u}}^d)$ be the feasible solution for $(\tilde{P}_B)$ found by Theorem 6.2.1 with

$$\begin{aligned}F(\bar{\boldsymbol{u}}^1, \bar{\boldsymbol{u}}^2, \cdots, \bar{\boldsymbol{u}}^d) &\geq (2/\pi)^{d-1} \ln(1+\sqrt{2})(n+1)^{-\frac{d-2}{2}} v(\tilde{P}_B)\\ &\geq (2/\pi)^{d-1} \ln(1+\sqrt{2})(n+1)^{-\frac{d-2}{2}} v(P_B).\end{aligned}$$

Denote $\bar{\boldsymbol{v}}^k = \bar{\boldsymbol{u}}^k/d$ for $k = 1, 2, \ldots, d$, and consequently

$$F(\bar{\boldsymbol{v}}^1, \bar{\boldsymbol{v}}^2, \cdots, \bar{\boldsymbol{v}}^d) = d^{-d} F(\bar{\boldsymbol{u}}^1, \bar{\boldsymbol{u}}^2, \cdots, \bar{\boldsymbol{u}}^d) \geq (2/\pi)^{d-1} \ln(1+\sqrt{2}) d^{-d} (n+1)^{-\frac{d-2}{2}} v(P_B).$$

Notice that for all $1 \leq k \leq d$, $|v_h^k| = |u_h^k/d| = 1/d \leq 1$ and the last component of tensor $\boldsymbol{F}$ is 0. By applying Lemma 5.2.4, it follows that

$$\mathrm{E}\left[\prod_{k=1}^{d} \eta_k F\left(\begin{pmatrix}\eta_1 \boldsymbol{v}^1\\ 1\end{pmatrix}, \begin{pmatrix}\eta_2 \boldsymbol{v}^2\\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\eta_d \boldsymbol{v}^d\\ 1\end{pmatrix}\right)\right] = F(\bar{\boldsymbol{v}}^1, \bar{\boldsymbol{v}}^2, \cdots, \bar{\boldsymbol{v}}^d)$$

and

$$\mathrm{E}\left[F\left(\begin{pmatrix}\xi_1 \boldsymbol{v}^1\\ 1\end{pmatrix}, \begin{pmatrix}\xi_2 \boldsymbol{v}^2\\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\xi_d \boldsymbol{v}^d\\ 1\end{pmatrix}\right)\right] = 0,$$

where $\eta_1, \eta_2, \ldots, \eta_d$ are independent random variables, each taking values 1 and -1 with $\mathrm{E}[\eta_k] = v_h^k$ for $k = 1, 2, \ldots, d$, and $\xi_1, \xi_2, \cdots, \xi_d$ are i.i.d. random variables, each taking values 1 and -1 with equal probability 1/2. Combining the two identities, we

have, for any constant c, the following identity

$$\begin{aligned} & F(\bar{\boldsymbol{v}}^1, \bar{\boldsymbol{v}}^2, \cdots, \bar{\boldsymbol{v}}^d) \\ = & \sum_{\boldsymbol{\beta}\in\mathbb{B}^d, \prod_{k=1}^d \beta_k = -1} (c - \operatorname{Prob}\{\boldsymbol{\eta} = \boldsymbol{\beta}\}) F\left(\begin{pmatrix}\beta_1 \boldsymbol{v}^1 \\ 1\end{pmatrix}, \begin{pmatrix}\beta_2 \boldsymbol{v}^2 \\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\beta_d \boldsymbol{v}^d \\ 1\end{pmatrix}\right) \\ & + \sum_{\boldsymbol{\beta}\in\mathbb{B}^d, \prod_{k=1}^d \beta_k = 1} (c + \operatorname{Prob}\{\boldsymbol{\eta} = \boldsymbol{\beta}\}) F\left(\begin{pmatrix}\beta_1 \boldsymbol{v}^1 \\ 1\end{pmatrix}, \begin{pmatrix}\beta_2 \boldsymbol{v}^2 \\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\beta_d \boldsymbol{v}^d \\ 1\end{pmatrix}\right). \end{aligned}$$

If we let $c = \max_{\boldsymbol{\beta}\in\mathbb{B}^d, \prod_{k=1}^d \beta_k = -1} \operatorname{Prob}\{\boldsymbol{\eta} = \boldsymbol{\beta}\}$, then the coefficient of each term F in the above is nonnegative. Therefore, a binary vector $\boldsymbol{\beta}' \in \mathbb{B}^d$ can be found, such that

$$F\left(\begin{pmatrix}\beta_1' \boldsymbol{v}^1 \\ 1\end{pmatrix}, \begin{pmatrix}\beta_2' \boldsymbol{v}^2 \\ 1\end{pmatrix}, \cdots, \begin{pmatrix}\beta_d' \boldsymbol{v}^d \\ 1\end{pmatrix}\right) \geq \tau_0 F(\bar{\boldsymbol{v}}^1, \bar{\boldsymbol{v}}^2, \cdots, \bar{\boldsymbol{v}}^d),$$

with

$$\begin{aligned} \tau_0 &= \left(\sum_{\boldsymbol{\beta}\in\mathbb{B}^d, \prod_{k=1}^d \beta_k = 1} (c + \operatorname{Prob}\{\boldsymbol{\eta} = \boldsymbol{\beta}\}) + \sum_{\boldsymbol{\beta}\in\mathbb{B}^d, \prod_{k=1}^d \beta_k = -1} (c - \operatorname{Prob}\{\boldsymbol{\eta} = \boldsymbol{\beta}\}) \right)^{-1} \\ &\geq \left(2^d c + 1\right)^{-1} \geq \left(2^d \left(\frac{1}{2} + \frac{1}{2d}\right)^d + 1\right)^{-1} \geq \frac{1}{1+e}, \end{aligned}$$

where $c \leq \left(\frac{1}{2} + \frac{1}{2d}\right)^d$ is applied, since $\mathrm{E}[\eta_k] = v_h^k = \pm 1/d$ for $k = 1, 2, \ldots, d$. Denote $\bar{\boldsymbol{z}}^k = \binom{\boldsymbol{z}^k}{z_h^k} = \binom{\beta_k' \boldsymbol{v}^k}{1}$ for $k = 1, 2, \ldots, d$, and we have

$$F(\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \cdots, \bar{\boldsymbol{z}}^d) \geq \tau_0 F(\bar{\boldsymbol{v}}^1, \bar{\boldsymbol{v}}^2, \cdots, \bar{\boldsymbol{v}}^d) \geq \left(\frac{2}{\pi}\right)^{d-1} \frac{\ln\left(1+\sqrt{2}\right)}{1+e} d^{-d} (n+1)^{-\frac{d-2}{2}} v(P_B).$$

For any $\boldsymbol{\beta} \in \mathbb{B}^d$, denote $\bar{\boldsymbol{z}}(\beta) = \beta_1 (d+1) \bar{\boldsymbol{z}}^1 + \sum_{k=2}^d \beta_k \bar{\boldsymbol{z}}^k$. By noticing $z_h^k = 1$ and $|z_i^k| = |v_i^k| = |u_i^k|/d = 1/d$ for all $1 \leq k \leq d$ and $1 \leq i \leq n$, it follows that

$$2 \leq |z_h(\beta)| \leq 2d \text{ and } |z_i(\beta)| \leq (d+1)/d + (d-1)/d = 2 \quad \forall\, 1 \leq i \leq n.$$

Thus $\boldsymbol{z}(\beta)/z_h(\beta) \in \bar{\mathbb{B}}^n$. By Lemma 6.3.1, there exists $\boldsymbol{x}' \in \mathbb{B}^n$, such that

$$\underline{v}(P_B) \leq p(\boldsymbol{x}') \leq p\left(\boldsymbol{z}(\beta)/z_h(\beta)\right) = f\left(\bar{\boldsymbol{z}}(\beta)/z_h(\beta)\right).$$

Moreover, we shall argue below that

$$\beta_1 = 1 \Longrightarrow f(\bar{\boldsymbol{z}}(\beta)) \geq (2d)^d \underline{v}(P_B). \tag{6.3}$$

If this were not the case, then by Lemma 6.5.1 $f\left(\bar{\boldsymbol{z}}(\beta)/(2d)\right) < \underline{v}(P_B) \leq 0$. Notice that $\beta_1 = 1$ implies $z_h(\beta) > 0$, and thus we have

$$f\left(\frac{\bar{\boldsymbol{z}}(\beta)}{z_h(\beta)}\right) = \left(\frac{2d}{z_h(\beta)}\right)^d f\left(\frac{\bar{\boldsymbol{z}}(\beta)}{2d}\right) \leq f\left(\frac{\bar{\boldsymbol{z}}(\beta)}{2d}\right) < \underline{v}(P_B),$$

which is a contradiction.

Suppose $\boldsymbol{\xi} = (\xi_1, \xi_2, \cdots, \xi_d)^{\mathrm{T}}$, whose components are i.i.d. random variables, each taking values 1 and -1 with equal probability 1/2. Noticing that (6.3) holds and using the same argument as (5.12), we get

$$\frac{1}{2}\mathrm{E}\left[\left(f\left(\bar{\boldsymbol{z}}(\xi)\right) - (2d)^d \underline{v}(P_B)\right) \,\middle|\, \xi_1 = 1, \prod_{k=2}^{d} \xi_k = 1\right] \geq d! F\left((d+1)\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \cdots, \bar{\boldsymbol{z}}^d\right).$$

Therefore, a binary vector $\boldsymbol{\beta}'' \in \mathbb{B}^d$ with $\beta_1'' = \prod_{k=2}^{d} \beta_k'' = 1$ can be found, such that

$$\begin{aligned} f(\bar{\boldsymbol{z}}(\beta'')) - (2d)^d \underline{v}(P_B) &\geq 2d! F((d+1)\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \cdots, \bar{\boldsymbol{z}}^d) \\ &\geq \left(\frac{2}{\pi}\right)^{d-1} \frac{2\ln\left(1+\sqrt{2}\right)}{1+e}(d+1)!\, d^{-d}(n+1)^{-\frac{d-2}{2}} v(P_B). \end{aligned}$$

By Lemma 6.5.1, a binary vector $\boldsymbol{x}' \in \mathbb{B}^n$ can be found in polynomial-time with $p(\boldsymbol{x}') \geq 0$. Moreover, as $\boldsymbol{z}(\beta'')/z_h(\beta'') \in \bar{\mathbb{B}}^n$, by Lemma 6.3.1, another binary vector $\boldsymbol{x}'' \in \mathbb{B}^n$ can be found in polynomial-time with $p(\boldsymbol{x}'') \geq p(\boldsymbol{z}(\beta'')/z_h(\beta''))$. Below we shall prove at least one of $\boldsymbol{x}'$ and $\boldsymbol{x}''$ satisfies

$$p(\boldsymbol{x}) - \underline{v}(P_B) \geq \tau(P_B)\left(v(P_B) - \underline{v}(P_B)\right). \tag{6.4}$$

Indeed, if $-\underline{v}(P_B) \geq \tau(P_B)\left(v(P_B) - \underline{v}(P_B)\right)$, then $\boldsymbol{x}'$ satisfies (6.4) in this case. Otherwise we shall have $-\underline{v}(P_B) < \tau(P_B)\left(v(P_B) - \underline{v}(P_B)\right)$, then

$$v(P_B) > (1 - \tau(P_B))\left(v(P_B) - \underline{v}(P_B)\right) \geq \left(v(P_B) - \underline{v}(P_B)\right)/2,$$

which implies

$$\begin{aligned} f\left(\frac{\bar{\boldsymbol{z}}(\beta'')}{2d}\right) - \underline{v}(P_B) &\geq (2d)^{-d}\left(\frac{2}{\pi}\right)^{d-1} \frac{2\ln\left(1+\sqrt{2}\right)}{1+e}(d+1)!\, d^{-d}(n+1)^{-\frac{d-2}{2}} v(P_B) \\ &\geq \tau(P_B)\left(v(P_B) - \underline{v}(P_B)\right). \end{aligned}$$

The above inequality also implies that $f\left(\bar{\boldsymbol{z}}(\beta'')/(2d)\right) > 0$. Recall that $\beta_1'' = 1$ implies $z_h(\beta'') > 0$. Therefore,

$$p(\boldsymbol{x}'') \geq p\left(\frac{\boldsymbol{z}(\beta'')}{z_h(\beta'')}\right) = f\left(\frac{\bar{\boldsymbol{z}}(\beta'')}{z_h(\beta'')}\right) = \left(\frac{2d}{z_h(\beta'')}\right)^d f\left(\frac{\bar{\boldsymbol{z}}(\beta'')}{2d}\right) \geq f\left(\frac{\bar{\boldsymbol{z}}(\beta'')}{2d}\right),$$

which implies $\boldsymbol{x}''$ satisfies (6.4). Finally, $\arg\max\{p(\boldsymbol{x}'), p(\boldsymbol{x}'')\}$ satisfies (6.4) in both cases. □

We remark that (P_B) is indeed a very general discrete optimization model. For example, it can be used to model the following general polynomial optimization problem

in discrete values:

$$(PD) \quad \max \quad p(\boldsymbol{x})$$
$$\text{s.t.} \quad x_i \in \{a_1^i, a_2^i, \cdots, a_{m_i}^i\},\ i = 1, 2, \ldots, n.$$

To see this, we observe that by adopting the Lagrange interpolation technique and letting

$$x_i = \sum_{j=1}^{m_i} a_j^i \prod_{1 \le k \le m_i, k \ne j} \frac{u_i - k}{j - k} \quad \forall 1 \le i \le n,$$

the original decision variables can be equivalently transformed to

$$u_i = j \Longrightarrow x_i = a_j^i \quad \forall 1 \le i \le n,\ 1 \le j \le m_i,$$

where $u_i \in \{1, 2, \ldots, m_i\}$, which can be further represented by $\lceil \log_2 m_i \rceil$ independent binary variables. Combining these two steps of substitution, (PD) is then reformulated as (P_B), with the degree of its objective polynomial function no larger than $\max_{1 \le i \le n}\{d(m_i - 1)\}$, and the dimension of its decision variables being $\sum_{i=1}^{n} \lceil \log_2 m_i \rceil$.

In many real world applications, the data $\{a_1^i, a_2^i, \cdots, a_{m_i}^i\}$ $(i = 1, 2, \ldots, n)$ in (PD) are arithmetic sequences. Then it is much easier to transform (PD) to (P_B), without going through the Lagrange interpolation. It keeps the same degree of its objective polynomial function, and the dimension of its decision variables is $\sum_{i=1}^{n} \lceil \log_2 m_i \rceil$.

Finally, we remark that all the approximation algorithms proposed in this chapter are also applicable for the polynomial optimizations over hypercubes $(\bar{\mathbb{B}}^n)$, which are models $(T_B), (H_B), (M_B)$ and (P_B), i.e., the respective models $(T_B), (H_B), (M_B)$ and (P_B) with $\mathbb{B}$ being replaced by $\bar{\mathbb{B}}$. In particular, the square-free conditions are no longer required for homogeneous form objectives and mixed form objectives. Therefore Algorithm 6.3.1 and Algorithm 6.5.1 can be made simpler without going through the process in Lemma 6.3.1. We now conclude this section, as well as the theoretical part of this chapter, by the following theorem without proof.

Theorem 6.5.3 *The following approximation results hold for polynomial optimizations over hypercubes:*

1. $(T_{\bar{B}})$ *admits a polynomial-time randomized approximation algorithm with approximation ratio* $\tau(T_B)$;

2. *If* $d \ge 3$ *is odd, then* $(H_{\bar{B}})$ *admits a polynomial-time randomized approximation algorithm with approximation ratio* $\tau(H_B)$; *Otherwise* $d \ge 4$ *is even, then* $(H_{\bar{B}})$

admits a polynomial-time randomized approximation algorithm with relative approximation ratio $\tau(H_B)$;

3. *If one of $d_k (k = 1, 2, \ldots, s)$ is odd, then (M_B) admits a polynomial-time randomized approximation algorithm with approximation ratio $\tau(M_B)$; Otherwise all d_k $(k = 1, 2, \ldots, s)$ are even, then (M_B) admits a polynomial-time randomized approximation algorithm with relative approximation ratio $\tau(M_B)$;*

4. *(P_B) admits a polynomial-time randomized approximation algorithm with relative approximation ratio $\tau(P_B)$.*

6.6 Applications

The models studied in this chapter have versatile applications. Given the generic nature of the discrete polynomial optimization models, this point is perhaps self-evident. However, we believe it is helpful to present a few examples at this point with more details, to illustrate the potential modeling opportunities with the new optimization models. We shall present three problems in this section and show that they are readily formulated by the discrete polynomial optimization models in this chapter.

6.6.1 Cut-Norm of Tensors

The concept of *cut-norm* is initially defined on a real matrix $\boldsymbol{A} = (A_{ij}) \in \mathbb{R}^{n_1 \times n_2}$, denoted by $\|\boldsymbol{A}\|_C$, the maximum over all $I \subset \{1, 2, \ldots, n_1\}$ and $J \subset \{1, 2, \ldots, n_2\}$, of the quantity $|\sum_{i \in I, j \in J} A_{ij}|$. This concept plays a major role in the design of efficient approximation algorithms for dense graph and matrix problems (see e.g., [36, 3]). Alon and Naor in [5] proposed a polynomial-time randomized approximation algorithm that approximates the cut-norm with a factor at least 0.56, which is currently the best available approximation ratio. Since a matrix is a second order tensor, it is natural to extend the cut-norm to general higher order tensors, e.g., a recent paper by Kannan [62]. Specifically, given a d-th order tensor $\boldsymbol{F} = (F_{i_1 i_2 \cdots i_d}) \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, its cut-norm is defined as

$$\|\boldsymbol{F}\|_C := \max_{I_k \subset \{1,2,\ldots,n_k\}, k=1,2,\ldots,d} \left| \sum_{i_k \in I_k, k=1,2,\ldots,d} F_{i_1 i_2 \cdots i_d} \right|.$$

In fact, the cut-norm $\|\boldsymbol{F}\|_C$ is closely related to $\|\boldsymbol{F}\|_{\infty \mapsto 1}$, which is exactly in the form of (T_B). By Theorem 6.2.1, there is a polynomial-time randomized approximation

algorithm which computes $\|\boldsymbol{F}\|_{\infty\mapsto 1}$ with a factor at least $\Omega\left(\left(\prod_{k=1}^{d-2} n_k\right)^{-\frac{1}{2}}\right)$, where we assume $n_1 \le n_2 \le \cdots \le n_d$. The following proposition, asserts that the cut-norm of a general d-th order tensor can also be approximated by a factor of $\Omega\left(\left(\prod_{k=1}^{d-2} n_k\right)^{-\frac{1}{2}}\right)$.

Proposition 6.6.1 *For any d-th order tensor $\boldsymbol{F} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$, $\|\boldsymbol{F}\|_C \le \|\boldsymbol{F}\|_{\infty\mapsto 1} \le 2^d\|\boldsymbol{F}\|_C$.*

Proof. Recall that $\|\boldsymbol{F}\|_{\infty\mapsto 1} = \max_{\boldsymbol{x}^k\in\mathbb{B}^{n_k},k=1,2,\ldots,d} F(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^d)$. For any $\boldsymbol{x}^k \in \mathbb{B}^{n_k}$ $(k=1,2,\ldots,d)$, it follows that

$$\begin{aligned}
F(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^d) &= \sum_{1\le i_k\le n_k,\,k=1,2,\ldots,d} F_{i_1i_2\cdots i_d}x^1_{i_1}x^2_{i_2}\cdots x^d_{i_d}\\
&= \sum_{\boldsymbol{\beta}\in\mathbb{B}^d}\ \sum_{i_k\in\{j|x^k_j=\beta_k,1\le j\le n_k\},\,k=1,2,\ldots,d} F_{i_1i_2\cdots i_d}x^1_{i_1}x^2_{i_2}\cdots x^d_{i_d}\\
&= \sum_{\boldsymbol{\beta}\in\mathbb{B}^d}\left(\prod_{1\le k\le d}\beta_k \sum_{i_k\in\{j|x^k_j=\beta_k,1\le j\le n_k\},\,k=1,2,\ldots,d} F_{i_1i_2\cdots i_d}\right)\\
&\le \sum_{\boldsymbol{\beta}\in\mathbb{B}^d}\left|\sum_{i_k\in\{j|x^k_j=\beta_k,1\le j\le n_k\},\,k=1,2,\ldots,d} F_{i_1i_2\cdots i_d}\right|\\
&\le \sum_{\boldsymbol{\beta}\in\mathbb{B}^d}\|\boldsymbol{F}\|_C = 2^d\|\boldsymbol{F}\|_C,
\end{aligned}$$

which implies $\|\boldsymbol{F}\|_{\infty\mapsto 1} \le 2^d\|\boldsymbol{F}\|_C$.

Observe that $\|\boldsymbol{F}\|_C = \max_{\boldsymbol{z}^k\in\{0,1\}^{n_k},k=1,2,\ldots,d}|F(\boldsymbol{z}^1,\boldsymbol{z}^2,\cdots,\boldsymbol{z}^d)|$. For any $\boldsymbol{z}^k \in \{0,1\}^{n_k}$ $(k=1,2,\ldots,d)$, let $\boldsymbol{z}^k = (\boldsymbol{e}+\boldsymbol{x}^k)/2$, where $\boldsymbol{e}$ is the all one vector. Clearly $\boldsymbol{x}^k \in \mathbb{B}^{n_k}$ for $k=1,2,\ldots,d$, and thus

$$\begin{aligned}
F(\boldsymbol{z}^1,\boldsymbol{z}^2,\cdots,\boldsymbol{z}^d) &= F\left(\frac{\boldsymbol{e}+\boldsymbol{x}^1}{2},\frac{\boldsymbol{e}+\boldsymbol{x}^2}{2},\cdots,\frac{\boldsymbol{e}+\boldsymbol{x}^d}{2}\right)\\
&= \frac{F(\boldsymbol{e},\boldsymbol{e},\cdots,\boldsymbol{e})+F(\boldsymbol{x}^1,\boldsymbol{e},\cdots,\boldsymbol{e})+\cdots+F(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^d)}{2^d}\\
&\le \frac{1}{2^d}\cdot\|\boldsymbol{F}\|_{\infty\mapsto 1}\cdot 2^d = \|\boldsymbol{F}\|_{\infty\mapsto 1},
\end{aligned}$$

which implies $\|\boldsymbol{F}\|_C \le \|\boldsymbol{F}\|_{\infty\mapsto 1}$. □

6.6.2 Maximum Complete Satisfiability

The usual maximum satisfiability problem (see e.g., [38]) is to find the boolean values of the literals, so as to maximize the total weighted sum of the satisfied clauses. The

key point of the problem is that each clause is in the *disjunctive* form, namely if one of the literals is assigned the TRUE value, then the clause is called satisfied. If the literals are also *conjunctive*, then this form of satisfiability problem is easy to solve. However, if not all the clauses can be satisfied, and we alternatively look for an assignment that maximizes the weighted sum of the satisfied clauses, then the problem is quite different. To make a distinction from the usual Max-SAT problem, let us call the new problem to be *maximum complete satisfiability*, or to be abbreviated as Max-C-SAT. It is immediately clear that Max-C-SAT is NP-hard, since we can easily reduce the max-cut problem to it. The reduction can be done as follows. For each edge (v_i, v_j) we consider two clauses $\{x_i, \bar{x}_j\}$ and $\{\bar{x}_i, x_j\}$, both having weight w_{ij}. Then the Max-C-SAT solution leads to a solution for the max-cut problem.

Now consider an instance of the Max-C-SAT problem with m clauses, each clause containing no more than d literals. Suppose that clause k $(1 \leq k \leq m)$ has the following form

$$\{x_{k_1}, x_{k_2}, \cdots, x_{k_{s_k}}, \bar{x}_{k'_1}, \bar{x}_{k'_2}, \ldots, \bar{x}_{k'_{t_k}}\},$$

where $s_k + t_k \leq d$, associated with a weight $w_k \geq 0$ for $k = 1, 2, \ldots, m$. Then, the Max-C-SAT problem can be formulated in the form of (P_B) as

$$\begin{array}{ll} \max & \sum_{k=1}^{m} w_k \prod_{j=1}^{s_k} \frac{1+x_{k_j}}{2} \cdot \prod_{i=1}^{t_k} \frac{1-x_{k'_i}}{2} \\ \text{s.t.} & \boldsymbol{x} \in \mathbb{B}^n. \end{array}$$

According to Theorem 6.5.2 and the nonnegativity of the objective function, the above problem admits a polynomial-time randomized approximation algorithm with approximation ratio $\Omega\left(n^{-\frac{d-2}{2}}\right)$, which is independent of the number of clauses m.

6.6.3 Box-Constrained Diophantine Equation

Solving a system of linear equations where the variables are integers and constrained to a hypercube is an important problem in discrete optimization and linear algebra. Examples of applications include the classical Frobenius problem (see e.g., [2, 16]), and the market split problem [26], other from engineering applications in integrated circuits design and video signal processing. For more details, one is referred to Aardal et al. [1]. Essentially, the problem is to find an integer-valued $\boldsymbol{x} \in \mathbb{Z}^n$ and $\mathbf{0} \leq \boldsymbol{x} \leq \boldsymbol{u}$, such that

$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. The problem can be formulated by the least square method as

$$\begin{aligned} (DE) \quad \max \quad & -(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathbb{Z}^n, \, \boldsymbol{0} \leq \boldsymbol{x} \leq \boldsymbol{u}. \end{aligned}$$

According to the discussion at the end of Section 6.5, the above problem can be reformulated as a form of (P_B), whose objective function is quadratic polynomial and number of decision variables is $\sum_{i=1}^{n} \lceil \log_2(u_i + 1) \rceil$. By applying Theorem 6.5.2, (DE) admits a polynomial-time randomized approximation algorithm with a constant relative approximation ratio.

Generally speaking, the Diophantine equations are polynomial equations. The box-constrained polynomial equations can also be formulated by the least square method as of (DE). Suppose the highest degree of the polynomial equations is d. Then, this least square problem can be reformulated as a form of (P_B), with the degree of the objective polynomial being $2d$ and number of decision variables being $\sum_{i=1}^{n} \lceil \log_2(u_i + 1) \rceil$. By applying Theorem 6.5.2, this problem admits a polynomial-time randomized approximation algorithm with a relative approximation ratio $\Omega\left(\left(\sum_{i=1}^{n} \log u_i \right)^{-(d-1)} \right)$.

6.7 Numerical Experiments

In this section we are going to test the numerical performance of the algorithms proposed in this chapter. Our experiments focus on the model (T_B) with $d = 4$ as a typical case. Specifically the problem to be tested is

$$\begin{aligned} (ET_B) \quad \max \quad & F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{w}) = \textstyle\sum_{1 \leq i,j,k,\ell \leq n} F_{ijk\ell} \, x_i y_j z_k w_\ell \\ \text{s.t.} \quad & \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{w} \in \mathbb{B}^n. \end{aligned}$$

6.7.1 Randomly Simulated Data

The input data of (ET_B) is generated in the same way as that of (ET_S), with entries of $\boldsymbol{F}$ following i.i.d. standard normals. The first relaxation model for Algorithm 6.2.2 to approximately solve (ET_B) is

$$\begin{aligned} (\hat{ET_B}) \quad \max \quad & F(\boldsymbol{X}, \boldsymbol{w}) = \textstyle\sum_{1 \leq i,j,k,\ell \leq n} F_{ijk\ell} X_{ijk} w_\ell \\ \text{s.t.} \quad & \boldsymbol{X} \in \mathbb{B}^{n \times n \times n}, \, \boldsymbol{w} \in \mathbb{B}^n, \end{aligned}$$

which can be solved approximately using SDP relaxation and randomization method proposed by Alon and Naor [5]. However, the size of the SDP relaxation problem is

$(n^3+n)\times(n^3+n)$, which is intractable for current SDP solvers even when $n=8$. Therefore in our testings, we further relax the above problem to

$$\begin{array}{ll}\max & F(\boldsymbol{X})=\sum_{1\le i,j,k,\ell\le n}F_{ijk\ell}X_{ijk\ell}\\ \text{s.t.} & \boldsymbol{X}\in\mathbb{B}^{n\times n\times n\times n},\end{array}$$

whose optimal solution is trivially $\operatorname{sign}(\boldsymbol{F})$ with optimal value $\bar{v}_B := \|\boldsymbol{F}\|_1$. This optimal solution can be rewritten as an $n^3\times n$ matrix, followed by applying DR 6.2.1 to get a feasible solution of $(E\hat{T}_B)$. Then we can apply the recursion procedures of Algorithm 6.2.2 to get a feasible solution of the original model (ET_B), with its objective value being denoted by v.

According to Theorem 6.2.1, the theoretical worst-case performance ratio of (ET_B) by Algorithm 6.2.2 is $\Omega(1/n)$. However, the theoretical ratio for the above method is indeed $\Omega(1/n^{1.5})$ because of a deeper relaxation, which can be proven by using the same argument in Theorem 6.2.1. However, this deeper relaxation allows us to skip the SDP relaxation of $(E\hat{T}_B)$, and make the method applicable for large dimensions. In general, the trivial upper bound of $v(ET_B)$ generated by this method, $\bar{v}_B$, may not good, and we may seek a tighter one. For this purpose we turn to the model (T_S) discussed in Section 3.2. Noticing that an n-dimensional binary vector has a norm $\sqrt{n}$, we may also relax (ET_B) to

$$\begin{array}{ll}\max & F(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{w})=\sum_{1\le i,j,k,\ell\le n}F_{ijk\ell}\,x_iy_jz_kw_\ell\\ \text{s.t.} & \|\boldsymbol{x}\|=\|\boldsymbol{y}\|=\|\boldsymbol{z}\|=\|\boldsymbol{w}\|=\sqrt{n},\\ & \boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{w}\in\mathbb{R}^n,\end{array}$$

which can be further relaxed to

$$\begin{array}{ll}\max & F(\boldsymbol{X},\boldsymbol{Z})=\sum_{1\le i,j,k,\ell\le n}F_{ijk\ell}X_{ij}Z_{k\ell}\\ \text{s.t.} & \|\boldsymbol{X}\|=\|\boldsymbol{Z}\|=n,\\ & \boldsymbol{X},\boldsymbol{Z}\in\mathbb{R}^{n\times n}.\end{array}$$

The above problem is the largest singular value problem, whose optimal value can be computed efficiently. Denoted its optimal value to be $\bar{v}_S$, which is taken as another upper bound of $v(ET_B)$.

The numerical results of 10 randomly generated instances for the upper bounds $\bar{v}_B$ and $\bar{v}_S$, as well as the objective values of the approximate solutions generated are listed in Table 6.1, which clearly shows that $\bar{v}_S$ outperforms $\bar{v}_B$ significantly. Therefore in the

Table 6.1: Numerical upper bounds of $v(ET_B)$ for $n = 13$

Instance	1	2	3	4	5	6	7	8	9	10
v	619	637	603	664	682	572	613	662	591	752
$\bar{v}_B$	22742	22588	22775	22711	22827	22905	22593	22966	22789	22678
$\bar{v}_S$	4251	4314	4346	4368	4294	4338	4295	4330	4330	4303

Table 6.2: Numerical ratios (average of 10 instances) of (ET_B)

n	5	10	20	30	40	50	60	70	80	90
τ (%)	35.42	18.51	9.94	7.06	5.45	4.09	3.93	3.06	2.99	2.58
$\tau \cdot n$	1.77	1.85	1.99	2.12	2.18	2.04	2.36	2.14	2.39	2.32
$\tau \cdot n^{0.9}$	1.51	1.47	1.47	1.51	1.51	1.38	1.56	1.40	1.54	1.48
$\tau \cdot n^{0.5}$	0.79	0.59	0.44	0.39	0.35	0.29	0.30	0.26	0.27	0.25

following general testings, we shall choose $\bar{v}_S$ as our candidate of the upper bound, to test the quality of the approximation solution, i.e., $\tau := v/\bar{v}_S$. The simulation results are listed in Table 6.2. By observation, the performance ratio is better than $\Omega(1/n)$, and is quite close to $\Omega(1/n^{0.9})$. It is clearly better than the theoretical ratio $\Omega(1/n^{1.5})$.

The computational cost for our method is quite low. In fact, for $n = 80$, we are able to get a feasible solution within 2 minutes, while computing the upper bound $\bar{v}_S$ costs much more time. For $n \geq 95$, however, our computer reports to run out of memory in the experiments, a problem purely due to the sheer size of the input data.

6.7.2 Data of Low-Rank Tensors

The numerical tests conducted so far are based on the data generating from i.i.d. standard normals. It would be interesting to investigate the practicability of our algorithms using other data settings. In particular, we shall test some low-rank tensors.

As mentioned in Section 3.4.2 (see also [68]), a fourth order tensor has rank r if it can be written as a summation of r number of rank-one tensors, and cannot be written as a summation of $r-1$ number of rank-one tensors. Specifically, the data we generate here is

$$\boldsymbol{F} := \sum_{i=1}^{r} \boldsymbol{a}_i^1 \otimes \boldsymbol{a}_i^2 \otimes \boldsymbol{a}_i^3 \otimes \boldsymbol{a}_i^4,$$

Table 6.3: Numerical ratios (average of 10 instances) of (ET_B) with low-rank tensors

r (rank)	1	2	3	4	5	6	7	8	9	10	15	20
τ (%) for $n = 10$	34.6	30.9	28.0	32.4	26.7	25.8	27.0	28.3	27.2	26.5	25.9	26.2
τ (%) for $n = 20$	14.8	15.0	14.0	15.7	11.3	11.1	11.2	11.8	12.1	11.7	11.2	12.0
τ (%) for $n = 30$	9.1	7.3	7.5	6.9	7.2	7.2	6.6	6.6	7.2	7.7	6.2	5.7

where all $\boldsymbol{a}_i^k$ $(k = 1, 2, 3, 4,\ i = 1, 2, \ldots, r)$ are independent of each other, each of which following i.i.d. standard normals.

We again use the method discussed in the previous subsection to approximately solve the model (ET_B), and compare its objective v with the upper bound $\bar{v}_S$, i.e., $\tau = v/\bar{v}_S$. The performance ratios for such data settings are shown in Table 6.3 for $n = 10, 20$ and 30. By observation, we find that low-rank tensors $\boldsymbol{F}$ improve the approximation ratios significantly. The lower the rank of the tensor $\boldsymbol{F}$, the better the performance ratio.

Chapter 7

Homogeneous Form Optimization with Mixed Constraints

7.1 Introduction

This chapter brings most of the results in previous chapters together, to discuss mixed integer programming problems. The objective functions are all homogenous polynomial functions, while the constraints are a combination of two most widely used ones, the spherical constraint and the binary constraint. In particular, the models considered include:

$$\begin{array}{lll} (T_{BS}) & \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d, \boldsymbol{y}^1, \boldsymbol{y}^2, \ldots, \boldsymbol{y}^{d'}) \\ & \text{s.t.} & \boldsymbol{x}^k \in \mathbb{B}^{n_k},\ k = 1, 2, \ldots, d, \\ & & \boldsymbol{y}^\ell \in \mathbb{S}^{m_\ell},\ \ell = 1, 2, \ldots, d'; \end{array}$$

$$\begin{array}{lll} (H_{BS}) & \max & f(\boldsymbol{x}, \boldsymbol{y}) \\ & \text{s.t.} & \boldsymbol{x} \in \mathbb{B}^n, \\ & & \boldsymbol{y} \in \mathbb{S}^m; \end{array}$$

$$\begin{array}{lll} (M_{BS}) & \max & f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s, \boldsymbol{y}^1, \boldsymbol{y}^2, \cdots, \boldsymbol{y}^t) \\ & \text{s.t.} & \boldsymbol{x}^k \in \mathbb{B}^{n_k},\ k = 1, 2, \ldots, s, \\ & & \boldsymbol{y}^\ell \in \mathbb{S}^{m_\ell},\ \ell = 1, 2, \ldots, t. \end{array}$$

The model (M_{BS}) is a generalization of the models (T_{BS}) and (H_{BS}). In fact, it can also be taken as generalization of most of the homogenous polynomial optimization

models discussed in previous chapters, namely (T_S) of Chapter 3, (H_S) and (M_S) of Chapter 4, and (T_B), (H_B) and (M_B) of Chapter 6 as well.

These mixed models have versatile applications, e.g., matrix combinatorial problem, vector-valued max-cut problem, whose details will be discussed in Section 7.5. Essentially, in many discrete optimization problems, if the objective to be optimized is extended from a scalar to a vector or a matrix, then we may turn to optimize the Euclidean norm of the vector, or the spectrum norm of the matrix, which turns out to be the mixed integer programming models proposed above.

All these models are NP-hard in general, even in the simplest case of one spherical constraint and one binary constraint, i.e., the model (T_{BS}) with $d = d' = 1$. As we will see later, it is actually equivalent to the maximization of a positive semidefinite form in binary variables, which includes max-cut as a subproblem and is thus NP-hard. In fact, this simplest form of (T_{BS}) serves as a basis for all these mixed integer programming models. By using this basis and mathematical induction, we are able to derive polynomial-time randomized approximation algorithms with worst-case performance ratios for (T_{BS}) with any fixed degree. The techniques are similar to that of Chapter 3, and two types of decomposition routines are called, one for decomposition of the spherical constraints, and one for decomposition of the binary constraints. Moreover, in order to extend the results from (T_{BS}) to (H_{BS}) and (M_{BS}), the multilinear tensor form relaxation method is again applied. Armed with the link lemmas (Lemma 4.2.1 and Lemma 4.4.3), we are able to derived approximation algorithms under some mild square-free conditions.

This chapter is organized as follows. We shall discuss models (T_{BS}), (H_{BS}) and (M_{BS}) in Sections 7.2, 7.3 and 7.4 respectively, and propose polynomial-time randomized approximation algorithms with provable approximation ratios or relative approximation ratios for the respective models. In Section 7.5, we shall discuss a few specific problems where these mixed models can be directly applied. For the easy of reading, in this chapter, we shall exclusively use vector $\boldsymbol{x}$ $(\in \mathbb{B}^n)$ to denote discrete variables, and vector $\boldsymbol{y}$ $(\in \mathbb{S}^m)$ to denote continuous variables. Throughout our discussion, we shall fix the degree of the objective polynomial function in these mixed models, $d + d'$, to be a constant.

7.2 Multilinear Form with Binary and Spherical Constraints

Our first mixed model is to maximize a multilinear function, with some variables being binary and some in the unit sphere, namely,

$$\begin{array}{lll} (T_{BS}) & \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d, \boldsymbol{y}^1, \boldsymbol{y}^2, \ldots, \boldsymbol{y}^{d'}) \\ & \text{s.t.} & \boldsymbol{x}^k \in \mathbb{B}^{n_k},\ k = 1, 2, \ldots, d, \\ & & \boldsymbol{y}^\ell \in \mathbb{S}^{m_\ell},\ \ell = 1, 2, \ldots, d', \end{array}$$

where $n_1 \le n_2 \le \cdots \le n_d$ and $m_1 \le m_2 \le \cdots \le m_{d'}$. This model is a generalization of (T_S) in Section 3.2 and (T_B) in Section 6.2.

The simplest case of (T_{BS}), $d = d' = 1$, is worth mention, as it plays an essential role in the whole chapter. Based on this case, we shall derive polynomial-time approximation algorithm with worst-case performance ratio for (T_{BS}) with any fixed degree $d + d'$.

Proposition 7.2.1 *If $d = d' = 1$, then (T_{BS}) is NP-hard, and admits a polynomial-time randomized approximation algorithm with approximation ratio $\sqrt{2/\pi}$.*

Proof. When $d = d' = 1$, (T_{BS}) can be written as

$$\begin{array}{lll} (\hat{T}_{BS}) & \max & \boldsymbol{x}^{\mathrm{T}} \boldsymbol{F} \boldsymbol{y} \\ & \text{s.t.} & \boldsymbol{x} \in \mathbb{B}^{n_1},\ \boldsymbol{y} \in \mathbb{S}^{m_1}. \end{array}$$

For any fixed $\boldsymbol{x}$ in $(\hat{T}_{BS})$, the corresponding optimal $\boldsymbol{y}$ must be $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{x}/\|\boldsymbol{F}^{\mathrm{T}}\boldsymbol{x}\|$ due to the Cauchy-Schwartz inequality, and accordingly,

$$\boldsymbol{x}^{\mathrm{T}} \boldsymbol{F} \boldsymbol{y} = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{F} \frac{\boldsymbol{F}^{\mathrm{T}} \boldsymbol{x}}{\|\boldsymbol{F}^{\mathrm{T}} \boldsymbol{x}\|} = \|\boldsymbol{F}^{\mathrm{T}} \boldsymbol{x}\| = \sqrt{\boldsymbol{x}^{\mathrm{T}} \boldsymbol{F} \boldsymbol{F}^{\mathrm{T}} \boldsymbol{x}}.$$

Thus $(\hat{T}_{BS})$ is equivalent to

$$\begin{array}{lll} & \max & \boldsymbol{x}^{\mathrm{T}} \boldsymbol{F} \boldsymbol{F}^{\mathrm{T}} \boldsymbol{x} \\ & \text{s.t.} & \boldsymbol{x} \in \mathbb{B}^{n_1}. \end{array}$$

Noticing that matrix $\boldsymbol{F}\boldsymbol{F}^{\mathrm{T}}$ is positive semidefinite, the above problem includes the max-cut problem (see e.g., [40]) as a subclass. Therefore it is NP-hard. Moreover, according to the result of Nesterov [88], it admits a polynomial-time randomized approximation algorithm (SDP relaxation and randomization) with approximation ratio $2/\pi$. This implies that $(\hat{T}_{BS})$ admits a polynomial-time randomized approximation algorithm with approximation ratio $\sqrt{2/\pi}$. □

Proposition 7.2.1 is the foundation to establish the basic relaxation in solving (T_{BS}) recursively for general degree d and d'. In processing to the high degree cases, for the recursion on d, with discrete variables $\boldsymbol{x}^k\,(k = 1, 2, \ldots, d)$, DR 6.2.1 is applied in each recursive step; while for the recursion on d' with continuous variables $\boldsymbol{y}^\ell\,(\ell = 1, 2, \ldots, d')$, two decomposition routines in Section 3.2 are readily available, namely the eigenvalue decomposition approach DR 3.2.2 and the randomized decomposition approach DR 3.2.1, either one of them will serve the purpose here. The main result in this section is the following:

Theorem 7.2.2 (T_{BS}) *admits a polynomial-time randomized approximation algorithm with approximation ratio* $\tau(T_{BS})$, *where*

$$\tau(T_{BS}) := \left(\frac{2}{\pi}\right)^{\frac{2d-1}{2}} \left(\prod_{k=1}^{d-1} n_k \prod_{\ell=1}^{d'-1} m_\ell\right)^{-\frac{1}{2}} = \Omega\left(\left(\prod_{k=1}^{d-1} n_k \prod_{\ell=1}^{d'-1} m_\ell\right)^{-\frac{1}{2}}\right).$$

Proof. The proof is based on mathematical induction on the degree $d + d'$, and Proposition 7.2.1 can be used as the base for the induction process when $d + d' = 1$.

For general $d + d' \geq 3$, if $d' \geq 2$, let $\boldsymbol{Y} = \boldsymbol{y}^1(\boldsymbol{y}^{d'})^{\mathrm{T}}$. Noticing that $\|\boldsymbol{Y}\|^2 = \|\boldsymbol{y}^1\|^2\|\boldsymbol{y}^{d'}\|^2 = 1$, similar to the relaxation in the proof of Theorem 3.2.4, (T_{BS}) can be relaxed to a case with degree $d + d' - 1$, i.e.,

$$\begin{array}{ll} \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d, \boldsymbol{Y}, \boldsymbol{y}^2, \boldsymbol{y}^3, \cdots, \boldsymbol{y}^{d'-1}) \\ \text{s.t.} & \boldsymbol{x}^k \in \mathbb{B}^{n_k},\ k = 1, 2, \ldots, d, \\ & \boldsymbol{Y} \in \mathbb{S}^{m_1 m_{d'}},\ \boldsymbol{y}^\ell \in \mathbb{S}^{m_\ell},\ \ell = 2, 3, \ldots, d' - 1. \end{array}$$

By induction, a feasible solution $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d, \hat{\boldsymbol{Y}}, \hat{\boldsymbol{y}}^2, \hat{\boldsymbol{y}}^3, \cdots, \hat{\boldsymbol{y}}^{d'-1})$ can be found in polynomial-time, such that

$$F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d, \hat{\boldsymbol{Y}}, \hat{\boldsymbol{y}}^2, \hat{\boldsymbol{y}}^3, \cdots, \hat{\boldsymbol{y}}^{d'-1}) \geq \left(\frac{2}{\pi}\right)^{\frac{2d-1}{2}} \left(\prod_{k=1}^{d-1} n_k \prod_{\ell=2}^{d'-1} m_\ell\right)^{-\frac{1}{2}} v(T_{BS}).$$

Let us denote matrix $\boldsymbol{Q} = F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d, \cdot, \hat{\boldsymbol{y}}^2, \hat{\boldsymbol{y}}^3, \cdots, \hat{\boldsymbol{y}}^{d'-1}, \cdot) \in \mathbb{R}^{m_1 \times m_{d'}}$. Then by Proposition 3.2.1 (used in DR 3.2.2), $\max_{\boldsymbol{y}^1 \in \mathbb{S}^{m_1}, \boldsymbol{y}^{d'} \in \mathbb{S}^{m_{d'}}} (\boldsymbol{y}^1)^{\mathrm{T}} \boldsymbol{Q}\, \boldsymbol{y}^{d'}$ can be solved in polynomial-time, with its optimal solution $(\hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^{d'})$ satisfying

$$F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d, \hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2, \cdots, \hat{\boldsymbol{y}}^{d'}) = (\hat{\boldsymbol{y}}^1)^{\mathrm{T}} \boldsymbol{Q} \hat{\boldsymbol{y}}^{d'} \geq \|\boldsymbol{Q}\| / \sqrt{m_1}.$$

By the Cauchy-Schwartz inequality, it follows that

$$\|\boldsymbol{Q}\| = \max_{\boldsymbol{Y} \in \mathbb{S}^{m_1 m_{d'}}} \boldsymbol{Q} \bullet \boldsymbol{Y} \geq \boldsymbol{Q} \bullet \hat{\boldsymbol{Y}} = F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d, \hat{\boldsymbol{Y}}, \hat{\boldsymbol{y}}^2, \hat{\boldsymbol{y}}^3, \cdots, \hat{\boldsymbol{y}}^{d'-1}).$$

Thus we concludes that

$$\begin{aligned} F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d, \hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2, \cdots, \hat{\boldsymbol{y}}^{d'}) &\geq \|\boldsymbol{Q}\|/\sqrt{m_1} \\ &\geq F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d, \hat{\boldsymbol{Y}}, \hat{\boldsymbol{y}}^2, \hat{\boldsymbol{y}}^3, \ldots, \hat{\boldsymbol{y}}^{d'-1})/\sqrt{m_1} \\ &\geq \tau(T_{BS})\, v(T_{BS}). \end{aligned}$$

For $d + d' \geq 3$ and $d \geq 2$, let $\boldsymbol{X} = \boldsymbol{x}^1(\boldsymbol{x}^d)^{\mathrm{T}}$, and (T_{BS}) can be relaxed to the other case with degree $d - 1 + d'$, i.e.,

$$\begin{aligned} \max \quad & F(\boldsymbol{X}, \boldsymbol{x}^2, \boldsymbol{x}^3, \cdots, \boldsymbol{x}^{d-1}, \boldsymbol{y}^1, \boldsymbol{y}^2, \cdots, \boldsymbol{y}^{d'}) \\ \text{s.t.} \quad & \boldsymbol{X} \in \mathbb{B}^{n_1 n_d}, \boldsymbol{x}^k \in \mathbb{B}^{n_k},\ k = 2, 3, \ldots, d-1, \\ & \boldsymbol{y}^\ell \in \mathbb{S}^{m_\ell},\ \ell = 1, 2, \ldots, d'. \end{aligned}$$

By induction, it admits a polynomial-time randomized approximation algorithm with approximation ratio $\left(\frac{2}{\pi}\right)^{\frac{2d-3}{2}} \left(\prod_{k=2}^{d-1} n_k \prod_{\ell=1}^{d'-1} m_\ell\right)^{-\frac{1}{2}}$. In order to decompose $\boldsymbol{X}$ into $\boldsymbol{x}^1$ and $\boldsymbol{x}^d$, we shall conduct the randomization procedure as in Step 2 of DR 6.2.1, which will further deteriorate by an additional factor of $\frac{2}{\pi\sqrt{n_1}}$ in expectation, as shown in (6.1). Combining these two factors, we are led to the ratio $\tau(T_{BS})$. □

We end this section by summarizing the algorithm for solving (T_{BS}) below.

Algorithm 7.2.1

- *INPUT: a $(d + d')$-th order tensor $\boldsymbol{F} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d \times m_1 \times m_2 \times \cdots \times m_{d'}}$ with $n_1 \leq n_2 \leq \cdots \leq n_d$ and $m_1 \leq m_2 \leq \cdots \leq m_{d'}$.*

1 *Rewrite $\boldsymbol{F}$ as a matrix $\boldsymbol{M} \in \mathbb{R}^{n_1 n_2 \cdots n_d \times m_1 m_2 \cdots m_{d'}}$ by combining its first d modes into the matrix row, and last d' modes into the matrix column.*

2 *Apply the procedure in Proposition 7.2.1, with input $\boldsymbol{M}$ and output $\hat{\boldsymbol{x}} \in \mathbb{B}^{n_1 n_2 \cdots n_d}$.*

3 *Rewrite the vector $\hat{\boldsymbol{x}}$ as a d-th order tensor $\hat{\boldsymbol{X}} \in \mathbb{B}^{n_1 \times n_2 \times \cdots \times n_d}$ and compute a d'-th order tensor $\boldsymbol{F}' = F(\hat{\boldsymbol{X}}, \cdot, \cdot, \cdots, \cdot) \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_{d'}}$.*

4 *Apply Algorithm 3.2.3, with input $\boldsymbol{F}'$ and output $(\hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2, \cdots, \hat{\boldsymbol{y}}^{d'})$.*

5 *Compute a d-th order tensor $\boldsymbol{F}'' = F(\cdot, \cdot, \cdots, \cdot, \hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2, \cdots, \hat{\boldsymbol{y}}^{d'}) \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$.*

6 *Apply Algorithm 6.2.2, with input $\boldsymbol{F}''$ and output $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$.*

- *OUTPUT: a feasible solution* $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d, \hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2, \cdots, \hat{\boldsymbol{y}}^{d'})$.

7.3 Homogeneous Form with Binary and Spherical Constraints

We further extend the mixed model in previous section to the homogeneous polynomial case, namely,

$$\boxed{\begin{array}{lll} (H_{BS}) & \max & f(\boldsymbol{x}, \boldsymbol{y}) \\ & \text{s.t.} & \boldsymbol{x} \in \mathbb{B}^n, \, \boldsymbol{y} \in \mathbb{S}^m, \end{array}}$$

where $f(\boldsymbol{x}, \boldsymbol{y}) = F(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x}}_{d}, \underbrace{\boldsymbol{y}, \boldsymbol{y}, \cdots, \boldsymbol{y}}_{d'})$, and $\boldsymbol{F} \in \mathbb{R}^{n^d \times m^{d'}}$ is a $(d + d')$-th order tensor with partial symmetric property. This model is a generalization of the model (H_S) in Section 4.2 and the model (H_B) in Section 6.3. We shall derive polynomial-time approximation algorithms with worst-case performance ratios. The method here is also multilinear function relaxation (T_{BS}), which admits a polynomial-time randomized approximation algorithm by Theorem 7.2.2. Then by applying Lemma 4.2.1 as a link, together with the square-free property for the discrete variables $\boldsymbol{x}$, we are led to the following results regarding (H_{BS}).

Theorem 7.3.1 *If $f(\boldsymbol{x}, \boldsymbol{y})$ is square-free in $\boldsymbol{x}$, and either d or d' is odd, then (H_{BS}) admits a polynomial-time randomized approximation algorithm with approximation ratio $\tau(H_{BS})$, where*

$$\tau(H_{BS}) := \left(\frac{2}{\pi}\right)^{\frac{2d-1}{2}} d! \, d^{-d} d'! \, d'^{-d'} n^{-\frac{d-1}{2}} m^{-\frac{d'-1}{2}} = \Omega\left(n^{-\frac{d-1}{2}} m^{-\frac{d'-1}{2}}\right).$$

Proof. Like in the proof of Theorem 6.3.2, by relaxing (H_{BS}) to (T_{BS}), we are able to find $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d, \hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2, \cdots, \hat{\boldsymbol{y}}^{d'})$ with $\hat{\boldsymbol{x}}^k \in \mathbb{B}^n$ for all $1 \leq k \leq d$ and $\hat{\boldsymbol{y}}^\ell \in \mathbb{S}^m$ for all $1 \leq \ell \leq d'$ in polynomial-time, such that

$$F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d, \hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2, \cdots, \hat{\boldsymbol{y}}^{d'}) \geq (2/\pi)^{\frac{2d-1}{2}} n^{-\frac{d-1}{2}} m^{-\frac{d'-1}{2}} v(H_{BS}).$$

Let $\xi_1, \xi_2, \cdots, \xi_d, \eta_1, \eta_2, \cdots, \eta_{d'}$ be i.i.d. random variables, each taking values 1 and -1 with equal probability $1/2$. By applying Lemma 4.4.3 (or Lemma 4.2.1 twice), we have

$$\mathrm{E}\left[\prod_{i=1}^{d} \xi_i \prod_{j=1}^{d'} \eta_j \, f\left(\sum_{k=1}^{d} \xi_k \hat{\boldsymbol{x}}^k, \sum_{\ell=1}^{d'} \eta_\ell \hat{\boldsymbol{y}}^\ell\right)\right] = d! d'! F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d, \hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2, \cdots, \hat{\boldsymbol{y}}^{d'}). \tag{7.1}$$

Thus we are able to find binary vectors $\boldsymbol{\beta} \in \mathbb{B}^d$ and $\boldsymbol{\beta}' \in \mathbb{B}^{d'}$, such that

$$\prod_{i=1}^{d} \beta_i \prod_{j=1}^{d'} \beta'_j f\left(\sum_{k=1}^{d} \beta_k \hat{\boldsymbol{x}}^k, \sum_{\ell=1}^{d'} \beta'_\ell \hat{\boldsymbol{y}}^\ell\right) \geq d!d'!F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d, \hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2, \cdots, \hat{\boldsymbol{y}}^{d'}).$$

Denote

$$(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) := \begin{cases} \left(\prod_{i=1}^{d} \beta_i \prod_{j=1}^{d'} \beta'_j \sum_{k=1}^{d} \beta_k \hat{\boldsymbol{x}}^k, \sum_{\ell=1}^{d'} \beta'_\ell \hat{\boldsymbol{y}}^\ell\right) & d \text{ is odd,} \\ \left(\sum_{k=1}^{d} \beta_k \hat{\boldsymbol{x}}^k, \prod_{i=1}^{d} \beta_i \prod_{j=1}^{d'} \beta'_j \sum_{\ell=1}^{d'} \beta'_\ell \hat{\boldsymbol{y}}^\ell\right) & d' \text{ is odd.} \end{cases}$$

Noticing $\|\hat{\boldsymbol{y}}\| \leq d'$ and combining the previous two inequalities, it follows that

$$f\left(\frac{\hat{\boldsymbol{x}}}{d}, \frac{\hat{\boldsymbol{y}}}{\|\hat{\boldsymbol{y}}\|}\right) \geq d^{-d} d'^{-d'} \prod_{i=1}^{d} \beta_i \prod_{j=1}^{d'} \beta'_j f\left(\sum_{k=1}^{d} \beta_k \hat{\boldsymbol{x}}^k, \sum_{\ell=1}^{d'} \beta'_\ell \hat{\boldsymbol{y}}^\ell\right) \geq \tau(H_{BS})\, v(H_{BS}).$$

Denote $\tilde{\boldsymbol{y}} = \hat{\boldsymbol{y}}/\|\hat{\boldsymbol{y}}\| \in \mathbb{S}^m$. Since $\hat{\boldsymbol{x}}/d \in \bar{\mathbb{B}}^n$ by a similar argument as (6.2), and $f(\boldsymbol{x}, \tilde{\boldsymbol{y}})$ is square-free in $\boldsymbol{x}$, by applying Lemma 6.3.1, $\tilde{\boldsymbol{x}} \in \mathbb{B}^n$ can be found in polynomial-time, such that

$$f(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}) \geq f(\hat{\boldsymbol{x}}/d, \tilde{\boldsymbol{y}}) \geq \tau(H_{BS})\, v(H_{BS}).$$

□

We remark that in Theorem 7.3.1, if $d' = 2$ and d is odd, then the factor $d'!\, d'^{-d'}$ in $\tau(H_{BS})$ can be removed for the same argument in the proof of Theorem 4.4.2 (basically the corresponding adjustment is an eigenvalue problem), and this improves the ratio $\tau(H_{BS})$ to $\left(\frac{2}{\pi}\right)^{\frac{2d-1}{2}} d!\, d^{-d} n^{-\frac{d-1}{2}} m^{-\frac{1}{2}}$. Now we present the approximation result for the even degree case.

Theorem 7.3.2 *If $f(\boldsymbol{x}, \boldsymbol{y})$ is square-free in $\boldsymbol{x}$, and both d and d' are even, then (H_{BS}) admits a polynomial-time randomized approximation algorithm with relative approximation ratio $\tau(H_{BS})$.*

Proof. Following the same argument as in the proof of Theorem 7.3.1, we shall get (7.1), which implies

$$\mathrm{E}\left[\prod_{i=1}^{d} \xi_i \prod_{j=1}^{d'} \eta_j f\left(\sum_{k=1}^{d} \xi_k \hat{\boldsymbol{x}}^k, \sum_{\ell=1}^{d'} \eta_\ell \hat{\boldsymbol{y}}^\ell\right)\right] \geq \left(\frac{2}{\pi}\right)^{\frac{2d-1}{2}} d!\, d'!\, n^{-\frac{d-1}{2}} m^{-\frac{d'-1}{2}} v(H_{BS}).$$

Denote $\hat{\boldsymbol{x}}_\xi := \frac{1}{d} \sum_{k=1}^{d} \xi_k \hat{\boldsymbol{x}}^k$ and $\hat{\boldsymbol{y}}_\eta := \frac{1}{d'} \sum_{\ell=1}^{d'} \eta_\ell \hat{\boldsymbol{y}}^\ell$. Clearly we have

$$\mathrm{E}\left[\prod_{i=1}^{d} \xi_i \prod_{j=1}^{d'} \eta_j f\left(\hat{\boldsymbol{x}}_\xi, \hat{\boldsymbol{y}}_\eta\right)\right] \geq \tau(H_{BS})\, v(H_{BS}).$$

Pick any fixed $\tilde{\boldsymbol{y}} \in \mathbb{S}^m$ and consider the following problem

$$\begin{aligned} (\tilde{H}_{BS}) \quad \max \quad & f(\boldsymbol{x}, \tilde{\boldsymbol{y}}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathbb{B}^n. \end{aligned}$$

Since $f(\boldsymbol{x}, \tilde{\boldsymbol{y}})$ is square-free in $\boldsymbol{x}$ and has no constant term, by Lemma 6.5.1, a binary vector $\tilde{\boldsymbol{x}} \in \mathbb{B}^n$ can be found in polynomial-time with

$$f(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}) \geq 0 \geq \underline{v}(\tilde{H}_{BS}) \geq \underline{v}(H_{BS}).$$

Next we shall argue $f\left(\hat{\boldsymbol{x}}_\xi, \hat{\boldsymbol{y}}_\eta\right) \geq \underline{v}(H_{BS})$. If this were not the case, then $f\left(\hat{\boldsymbol{x}}_\xi, \hat{\boldsymbol{y}}_\eta\right) < \underline{v}(H_{BS}) \leq 0$. By noticing $\|\hat{\boldsymbol{y}}_\eta\| \leq 1$, this leads to

$$f\left(\hat{\boldsymbol{x}}_\xi, \hat{\boldsymbol{y}}_\eta / \|\hat{\boldsymbol{y}}_\eta\|\right) = \|\hat{\boldsymbol{y}}_\eta\|^{-d'} f\left(\hat{\boldsymbol{x}}_\xi, \hat{\boldsymbol{y}}_\eta\right) \leq f\left(\hat{\boldsymbol{x}}_\xi, \hat{\boldsymbol{y}}_\eta\right) < \underline{v}(H_{BS}).$$

Also noticing $\hat{\boldsymbol{x}}_\xi \in \bar{\mathbb{B}}^n$, by applying Lemma 6.3.1, a binary vector $\hat{\boldsymbol{x}} \in \mathbb{B}^n$ can be found with

$$\underline{v}(H_{BS}) \leq f(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}_\eta / \|\hat{\boldsymbol{y}}_\eta\|) \leq f\left(\hat{\boldsymbol{x}}_\xi, \hat{\boldsymbol{y}}_\eta / \|\hat{\boldsymbol{y}}_\eta\|\right) < \underline{v}(H_{BS})$$

resulting in a contradiction.

By that $f\left(\hat{\boldsymbol{x}}_\xi, \hat{\boldsymbol{y}}_\eta\right) - \underline{v}(H_{BS}) \geq 0$, it follows

$$\begin{aligned} & \frac{1}{2}\mathrm{E}\left[f\left(\hat{\boldsymbol{x}}_\xi, \hat{\boldsymbol{y}}_\eta\right) - \underline{v}(H_{BS}) \,\middle|\, \prod_{i=1}^{d}\xi_i \prod_{j=1}^{d'}\eta_j = 1\right] \\ \geq \; & \frac{1}{2}\mathrm{E}\left[f\left(\hat{\boldsymbol{x}}_\xi, \hat{\boldsymbol{y}}_\eta\right) - \underline{v}(H_{BS}) \,\middle|\, \prod_{i=1}^{d}\xi_i \prod_{j=1}^{d'}\eta_j = 1\right] \\ & -\frac{1}{2}\mathrm{E}\left[f\left(\hat{\boldsymbol{x}}_\xi, \hat{\boldsymbol{y}}_\eta\right) - \underline{v}(H_{BS}) \,\middle|\, \prod_{i=1}^{d}\xi_i \prod_{j=1}^{d'}\eta_j = -1\right] \\ = \; & \mathrm{E}\left[\prod_{i=1}^{d}\xi_i \prod_{j=1}^{d'}\eta_j \left(f\left(\hat{\boldsymbol{x}}_\xi, \hat{\boldsymbol{y}}_\eta\right) - \underline{v}(H_{BS})\right)\right] \\ = \; & \mathrm{E}\left[\prod_{i=1}^{d}\xi_i \prod_{j=1}^{d'}\eta_j f\left(\hat{\boldsymbol{x}}_\xi, \hat{\boldsymbol{y}}_\eta\right)\right] \geq \tau(H_{BS})\, v(H_{BS}). \end{aligned}$$

Thus we are able to find $\boldsymbol{\beta} \in \mathbb{B}^d$ and $\boldsymbol{\beta}' \in \mathbb{B}^{d'}$ with $\prod_{i=1}^{d}\beta_i \prod_{j=1}^{d'}\beta'_j = 1$, such that

$$f\left(\hat{\boldsymbol{x}}_\beta, \hat{\boldsymbol{y}}_{\beta'}\right) - \underline{v}(H_{BS}) \geq 2\,\tau(H_{BS})\, v(H_{BS}).$$

Denote $\hat{\boldsymbol{y}} = \hat{\boldsymbol{y}}_{\beta'} / \|\hat{\boldsymbol{y}}_{\beta'}\| \in \mathbb{S}^m$. Since $\hat{\boldsymbol{x}}_\beta \in \bar{\mathbb{B}}^n$, by Lemma 6.3.1, a binary vector $\hat{\boldsymbol{x}} \in \mathbb{B}^n$ can be found in polynomial-time with $f(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) \geq f\left(\hat{\boldsymbol{x}}_\beta, \hat{\boldsymbol{y}}\right)$.

Below we shall prove either $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$ or $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ will satisfy

$$f(\boldsymbol{x}, \boldsymbol{y}) - \underline{v}(H_{BS}) \geq \tau(H_{BS}) \left(v(H_{BS}) - \underline{v}(H_{BS})\right). \tag{7.2}$$

Indeed, if $-\underline{v}(H_{BS}) \geq \tau(H_{BS}) \left(v(H_{BS}) - \underline{v}(H_{BS})\right)$, then $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$ satisfies (7.2) in this case since $f(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}) \geq 0$. Otherwise, if $-\underline{v}(H_{BS}) < \tau(H_{BS}) \left(v(H_{BS}) - \underline{v}(H_{BS})\right)$, then

$$v(H_{BS}) > (1 - \tau(H_{BS})) \left(v(H_{BS}) - \underline{v}(H_{BS})\right) \geq \left(v(H_{BS}) - \underline{v}(H_{BS})\right)/2,$$

which implies

$$f\left(\hat{\boldsymbol{x}}_\beta, \hat{\boldsymbol{y}}_{\beta'}\right) - \underline{v}(H_{BS}) \geq 2\tau(H_{BS})\, v(H_{BS}) \geq \tau(H_{BS}) \left(v(H_{BS}) - \underline{v}(H_{BS})\right).$$

The above inequality also implies that $f\left(\hat{\boldsymbol{x}}_\beta, \hat{\boldsymbol{y}}_{\beta'}\right) > 0$. Therefore, we have

$$f(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) \geq f\left(\hat{\boldsymbol{x}}_\beta, \hat{\boldsymbol{y}}\right) = \|\hat{\boldsymbol{y}}_{\beta'}\|^{-d'} f\left(\hat{\boldsymbol{x}}_\beta, \hat{\boldsymbol{y}}_{\beta'}\right) \geq f\left(\hat{\boldsymbol{x}}_\beta, \hat{\boldsymbol{y}}_{\beta'}\right),$$

which implies $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ satisfies (7.2). Finally, $\arg\max\{f(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}), f(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})\}$ satisfies (7.2) in both cases. □

7.4 Mixed Form with Binary and Spherical Constraints

The final story of polynomial optimization problems in this thesis brings together a bunch of models discussed in previous sections and chapters, as a generalization of a large family, which includes (T_S), (H_S), (M_S), (T_B), (H_B), (M_B), (T_{BS}) and (H_{BS}) all as its subclasses. The model is to maximize a mixed form over variables in binary constraints, mixed with variables in spherical constraints, i.e.,

$$\begin{array}{lll} (M_{BS}) & \max & f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s, \boldsymbol{y}^1, \boldsymbol{y}^2, \cdots, \boldsymbol{y}^t) \\ & \text{s.t.} & \boldsymbol{x}^k \in \mathbb{B}^{n_k},\ k = 1, 2, \ldots, s, \\ & & \boldsymbol{y}^\ell \in \mathbb{S}^{m_\ell},\ \ell = 1, 2, \ldots, t, \end{array}$$

where associated with function f is a tensor $\boldsymbol{F} \in \mathbb{R}^{n_1{}^{d_1} \times n_2{}^{d_2} \times \cdots \times n_s{}^{d_s} \times m_1{}^{d'_1} \times m_2{}^{d'_2} \times \cdots \times m_t{}^{d'_t}}$ with partial symmetric property, $n_1 \leq n_2 \leq \cdots \leq n_s$ and $m_1 \leq m_2 \leq \cdots \leq m_t$, and $d = d_1 + d_2 + \cdots + d_s$ and $d' = d'_1 + d'_2 + \cdots + d'_t$ are deemed as fixed constants.

We shall derive polynomial-time approximation algorithms for this general model. By relaxing (M_{BS}) to the multilinear function optimization model (T_{BS}) and solving it approximately using Theorem 7.2.2, we may further adjust its solution one by one using the link Lemma 4.2.1 or Lemma 4.4.3, leading to the following general results in two settings.

Theorem 7.4.1 *If $f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s, \boldsymbol{y}^1, \boldsymbol{y}^2, \cdots, \boldsymbol{y}^t)$ is square-free in each $\boldsymbol{x}^k$ $(k = 1, 2, \ldots, s)$, and one of d_k $(k = 1, 2, \ldots, s)$ or one of d'_ℓ $(\ell = 1, 2, \ldots, t)$ is odd, then (M_{BS}) admits a polynomial-time randomized approximation algorithm with approximation ratio $\hat{\tau}(M_{BS})$, where*

$$\begin{aligned}\hat{\tau}(M_{BS}) &:= \left(\frac{2}{\pi}\right)^{\frac{2d-1}{2}} \left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}} \prod_{1\le \ell \le t, 3 \le d'_\ell} \frac{d'_\ell!}{{d'_\ell}^{d'_\ell}}\right) \left(\frac{\prod_{k=1}^{s} {n_k}^{d_k} \prod_{\ell=1}^{t} {m_\ell}^{d'_\ell}}{n_s\, m_t}\right)^{-\frac{1}{2}} \\ &= \Omega\left(\left(\frac{\prod_{k=1}^{s} {n_k}^{d_k} \prod_{\ell=1}^{t} {m_\ell}^{d'_\ell}}{n_s\, m_t}\right)^{-\frac{1}{2}}\right).\end{aligned}$$

Proof. The proof is analogous to that of Theorem 7.3.1. We first relax (M_{BS}) to (T_{BS}) and get its approximate solution $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d, \hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2, \cdots, \hat{\boldsymbol{y}}^{d'})$ using Theorem 7.2.2. Let $\xi_1, \xi_2, \cdots, \xi_d, \eta_1, \eta_2, \cdots, \eta_{d'}$ be i.i.d. random variables, each taking values 1 and -1 with equal probability 1/2. By applying Lemma 4.4.3, we have

$$\begin{aligned}&\mathrm{E}\left[\prod_{i=1}^{d} \xi_i \prod_{j=1}^{d'} \eta_j f\left(\hat{\boldsymbol{x}}_\xi^1, \hat{\boldsymbol{x}}_\xi^2, \cdots, \hat{\boldsymbol{x}}_\xi^s, \hat{\boldsymbol{y}}_\eta^1, \hat{\boldsymbol{y}}_\eta^2, \cdots, \hat{\boldsymbol{y}}_\eta^t\right)\right] \\ = &\prod_{k=1}^{s} d_k! \prod_{\ell=1}^{t} d'_\ell!\, F\left(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d, \hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2, \cdots, \hat{\boldsymbol{y}}^{d'}\right), \qquad (7.3)\end{aligned}$$

where

$$\hat{\boldsymbol{x}}_\xi^1 := \sum_{k=1}^{d_1} \xi_k \hat{\boldsymbol{x}}^k,\ \hat{\boldsymbol{x}}_\xi^2 := \sum_{k=d_1+1}^{d_1+d_2} \xi_k \hat{\boldsymbol{x}}^k,\ \cdots,\ \hat{\boldsymbol{x}}_\xi^s := \sum_{k=d_1+d_2+\cdots+d_{s-1}+1}^{d} \xi_k \hat{\boldsymbol{x}}^k, \qquad (7.4)$$

and

$$\hat{\boldsymbol{y}}_\eta^1 := \sum_{\ell=1}^{d'_1} \eta_\ell\, \hat{\boldsymbol{y}}^\ell,\ \hat{\boldsymbol{y}}_\eta^2 := \sum_{\ell=d'_1+1}^{d'_1+d'_2} \eta_\ell\, \hat{\boldsymbol{y}}^\ell,\ \cdots,\ \hat{\boldsymbol{y}}_\eta^t := \sum_{\ell=d'_1+d'_2+\cdots+d'_{t-1}+1}^{d'} \eta_\ell\, \hat{\boldsymbol{y}}^\ell. \qquad (7.5)$$

In (7.3), as one of d_k $(k = 1, 2, \ldots, s)$ or one of d'_ℓ $(\ell = 1, 2, \ldots, t)$ is odd, we are able to move $\prod_{i=1}^{d} \xi_i \prod_{j=1}^{d'} \eta_j$ into the coefficient of the corresponding vector ($\hat{\boldsymbol{x}}_\xi^k$ or $\hat{\boldsymbol{y}}_\eta^\ell$ whenever appropriate) in the function f. Other derivations are essentially the same as the proof of Theorem 7.3.1. Besides, we only loose a ratio of $d'_\ell!/{d'_\ell}^{d'_\ell}$ when $d'_\ell \ge 3$ in $\hat{\tau}(M_{BS})$. This is because when $d'_\ell \le 2$, the corresponding adjustments can be done without deteriorating the ratio, like in the proof of Theorem 4.4.2. □

Theorem 7.4.2 *If $f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s, \boldsymbol{y}^1, \boldsymbol{y}^2, \cdots, \boldsymbol{y}^t)$ is square-free in each $\boldsymbol{x}^k$ $(k = 1, 2, \ldots, s)$, and all d_k $(k = 1, 2, \ldots, s)$ and all d'_ℓ $(\ell = 1, 2, \ldots, t)$ are even, then (M_{BS})*

admits a polynomial-time randomized approximation algorithm with relative approximation ratio $\tau(M_{BS})$, where

$$\begin{aligned}\tau(M_{BS}) &:= \left(\frac{2}{\pi}\right)^{\frac{2d-1}{2}} \left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}} \prod_{\ell=1}^{t} \frac{d'_\ell!}{{d'_\ell}^{d'_\ell}}\right) \left(\frac{\prod_{k=1}^{s} {n_k}^{d_k} \prod_{\ell=1}^{t} {m_\ell}^{d'_\ell}}{n_s\, m_t}\right)^{-\frac{1}{2}} \\ &= \Omega\left(\left(\frac{\prod_{k=1}^{s} {n_k}^{d_k} \prod_{\ell=1}^{t} {m_\ell}^{d'_\ell}}{n_s\, m_t}\right)^{-\frac{1}{2}}\right).\end{aligned}$$

Proof. The proof is analogous to that of Theorem 7.3.2. The main differences are: (i) we use (7.3) instead of (7.1); and (ii) we use $f\left(\frac{\hat{\boldsymbol{x}}_\xi^1}{d_1}, \frac{\hat{\boldsymbol{x}}_\xi^2}{d_2}, \cdots, \frac{\hat{\boldsymbol{x}}_\xi^s}{d_s}, \frac{\hat{\boldsymbol{y}}_\eta^1}{d'_1}, \frac{\hat{\boldsymbol{y}}_\eta^2}{d'_2}, \cdots, \frac{\hat{\boldsymbol{y}}_\eta^t}{d'_t}\right)$ instead of $f\left(\hat{\boldsymbol{x}}_\xi, \hat{\boldsymbol{y}}_\eta\right)$, where $\left(\hat{\boldsymbol{x}}_\xi^1, \hat{\boldsymbol{x}}_\xi^2, \cdots, \hat{\boldsymbol{x}}_\xi^s, \hat{\boldsymbol{y}}_\eta^1, \hat{\boldsymbol{y}}_\eta^2, \cdots, \hat{\boldsymbol{y}}_\eta^t\right)$ are defined in (7.4) and (7.5). □

7.5 Applications

The generality of the mixed integer polynomial optimizations studied in this chapter gives rises to some succinct and interesting problems, apart from their versatile applications. Nevertheless, it should be useful and helpful to present a few examples at this point with more details, to illustrate the potential modeling opportunities with the new optimization models. In this section, we shall discuss the matrix combinatorial problem and some extended version of the max-cut problem, and show that they are readily formulated by the mixed integer programming problems in this chapter.

7.5.1 Matrix Combinatorial Problem

We discuss a succinct and interesting matrix combinatorial problem. Given n matrices $\boldsymbol{A}_i \in \mathbb{R}^{m_1 \times m_2}$ for $i = 1, 2, \ldots, n$, find a binary combination of them so as to maximize the combined matrix in terms of spectral norm. Specifically, the following optimization model

$$\begin{array}{lll}(MCP) & \max & \sigma_{\max}(\sum_{i=1}^{n} x_i \boldsymbol{A}_i) \\ & \text{s.t.} & x_i \in \{1, -1\},\ i = 1, 2, \ldots, n,\end{array}$$

where $\sigma_{\max}$ denotes the largest singular value of a matrix. Problem (MCP) is NP-hard, even in a special case of $m_2 = 1$. In this case, the matrix $\boldsymbol{A}_i$ is replace by an m_1-dimensional vector $\boldsymbol{a}^i$, with the spectral norm being identical to the Euclidean norm of a vector. The vector version combinatorial problem is then

$$\begin{array}{ll}\max & \|\sum_{i=1}^{n} x_i \boldsymbol{a}^i\| \\ \text{s.t.} & x_i \in \{1, -1\},\ i = 1, 2, \ldots, n.\end{array}$$

This is equivalent to the model (T_{BS}) with $d = d' = 1$, whose NP-hardness is asserted by Proposition 7.2.1.

Turning back to the general matrix version (MCP), the problem has an equivalent formulation

$$\begin{aligned} \max \quad & (\boldsymbol{y}^1)^{\mathrm{T}} \left(\textstyle\sum_{i=1}^n x_i \boldsymbol{A}_i\right) \boldsymbol{y}^2 \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathbb{B}^n,\ \boldsymbol{y}^1 \in \mathbb{S}^{m_1},\ \boldsymbol{y}^2 \in \mathbb{S}^{m_2}, \end{aligned}$$

which is essentially the model (T_{BS}) with $d = 1$ and $d' = 2$

$$\begin{aligned} \max \quad & F(\boldsymbol{x}, \boldsymbol{y}^1, \boldsymbol{y}^2) \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathbb{B}^n,\ \boldsymbol{y}^1 \in \mathbb{S}^{m_1},\ \boldsymbol{y}^2 \in \mathbb{S}^{m_2}, \end{aligned}$$

where associated with the trilinear function F is a third order tensor $\boldsymbol{F} \in \mathbb{R}^{n \times m_1 \times m_2}$, whose (i, j, k)-th entry is (j, k)-th entry of the matrix $\boldsymbol{A}_i$. According to Theorem 7.2.2, the largest matrix (in terms of spectral norm in (MCP) formulation) can be approximated with a factor of $\sqrt{\frac{2}{\pi \min\{m_1, m_2\}}}$.

If the given n matrices $\boldsymbol{A}_i\,(i = 1, 2, \ldots, n)$ are symmetric, then the maximization criterion can be set for the largest eigenvalue in stead of the largest singular value, i.e.,

$$\begin{aligned} \max \quad & \lambda_{\max}\left(\textstyle\sum_{i=1}^n x_i \boldsymbol{A}_i\right) \\ \text{s.t.} \quad & x_i \in \{1, -1\},\ i = 1, 2, \ldots, n. \end{aligned}$$

It is also easy to formulate this problem as the model (H_{BS}) with $d = 1$ and $d' = 2$

$$\begin{aligned} \max \quad & F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{y}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathbb{B}^n,\ \boldsymbol{y} \in \mathbb{S}^m, \end{aligned}$$

whose optimal value can also be approximated with a factor of $\sqrt{\frac{2}{\pi m}}$ by Theorem 7.3.1 and the remarks that followed.

7.5.2 Vector-Valued Maximum Cut

Consider an undirected graph $G = (V, E)$ where $V = \{v_1, v_2, \cdots, v_n\}$ is the set of the vertices, and $E \subset V \times V$ is the set of the edges. On each edge $e \in E$ there is an associated weight, which is a *nonnegative vector* in this case, i.e., $\boldsymbol{w}_e \in \mathbb{R}^m, \boldsymbol{w}_e \geq \boldsymbol{0}$ for all $e \in E$. The problem now is to find a cut in such a way that the total sum of the weights, which is a vector in this case, has a maximum norm. More formally, this problem can be formulated as

$$\max_{C \text{ is a cut of } G} \left\| \sum_{e \in C} \boldsymbol{w}_e \right\|.$$

Note that the usual max-cut problem is a special case of the above model where each weight $w_e \geq 0$ is a scalar. Similar to the scalar case (see [40]), we may reformulate the above problem in binary variables as

$$\begin{array}{ll} \max & \left\| \sum_{1 \leq i,j \leq n} x_i x_j \boldsymbol{w}'_{ij} \right\| \\ \text{s.t.} & \boldsymbol{x} \in \mathbb{B}^n, \end{array}$$

where

$$\boldsymbol{w}'_{ij} = \begin{cases} -\boldsymbol{w}_{ij} & i \neq j, \\ -\boldsymbol{w}_{ij} + \sum_{k=1}^{n} \boldsymbol{w}_{ik} & i = j. \end{cases} \tag{7.6}$$

Observing the Cauchy-Schwartz inequality, we further formulate the above problem as

$$\begin{array}{ll} \max & \left(\sum_{1 \leq i,j \leq n} x_i x_j \boldsymbol{w}'_{ij} \right)^{\mathrm{T}} \boldsymbol{y} = F(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{y}) \\ \text{s.t.} & \boldsymbol{x} \in \mathbb{B}^n, \, \boldsymbol{y} \in \mathbb{S}^m. \end{array}$$

This is the exact form of (H_{BS}) with $d = 2$ and $d' = 1$. Although the square-free property in $\boldsymbol{x}$ does not hold in this model (which is a condition of Theorem 7.3.1), one can still replace any point in the hypercube $(\bar{\mathbb{B}}^n)$ by one of its vertices $(\mathbb{B}^n)$ without decreasing its objective function value, since the matrix $F(\cdot, \cdot, \boldsymbol{e}_k) = \left((w'_{ij})_k \right)_{n \times n}$ is diagonal dominant for $k = 1, 2, \ldots, m$. Therefore, the vector-valued max-cut problem admits an approximation ratio of $\frac{1}{2} \left(\frac{2}{\pi} \right)^{\frac{3}{2}} n^{-\frac{1}{2}}$ by Theorem 7.3.1.

If the weights on edges are *positive semidefinite matrices* (i.e., $\boldsymbol{W}_{ij} \in \mathbb{R}^{m \times m}$, $\boldsymbol{W}_{ij} \succeq 0$ for all $(i, j) \in E$), then the matrix-valued max-cut problem can also be formulated as

$$\begin{array}{ll} \max & \lambda_{\max} \left(\sum_{1 \leq i,j \leq n} x_i x_j \boldsymbol{W}'_{ij} \right) \\ \text{s.t.} & \boldsymbol{x} \in \mathbb{B}^n, \end{array}$$

where $\boldsymbol{W}'_{ij}$ is defined similarly as (7.6); or equivalently,

$$\begin{array}{ll} \max & \boldsymbol{y}^{\mathrm{T}} \left(\sum_{1 \leq i,j \leq n} x_i x_j \boldsymbol{W}'_{ij} \right) \boldsymbol{y} \\ \text{s.t.} & \boldsymbol{x} \in \mathbb{B}^n, \, \boldsymbol{y} \in \mathbb{S}^m, \end{array}$$

the model (H_{BS}) with $d = d' = 2$. Similar to the vector-valued case, by the diagonal dominant property and Theorem 7.3.2, the above problem admits an approximation ratio of $\frac{1}{4} \left(\frac{2}{\pi} \right)^{\frac{3}{2}} (mn)^{-\frac{1}{2}}$. Notice that Theorem 7.3.2 only asserts a relative approximation ratio. However for this problem the optimal value of its minimization counterpart is obviously nonnegative, and thus a relative approximation ratio implies a usual approximation ratio.

Chapter 8

Conclusion and Recent Developments

This thesis discusses various subclasses of polynomial optimization problems, with a focus on deriving polynomial-time approximation algorithms with worst-case performance guarantees. These subclasses include many frequently encountered constraints in the literature, such as the Euclidean spherical constraints, the Euclidean ball constraints, the ellipsoidal constraints, the binary constraints, and a mixture of them. The objective functions range from multilinear tensor functions, homogeneous polynomials, to general inhomogeneous polynomials. Multilinear tensor function optimizations play the key role in these algorithms, whose ideas are based on lower order multilinear form relaxations and decomposition routines. Connections between multilinear functions, homogenous polynomials, and inhomogeneous polynomials are established in preserving the approximation ratios. All the approximation results are listed in Table 8.1. The applications of these polynomial optimization models are discussed, which open up a door to many potential modeling opportunities. Reports on numerical testings show that the algorithms proposed are actually very effective, and they typically produce high quality solutions. The worst-case performance analysis offers a theoretical 'safety net', which is usually far from the typical performance. Table 8.1 summarizes the whole structure of the thesis and the approximation ratios.

Most of the results presented in this thesis have been documented and submitted for publications in research papers [47, 48, 49], which are all joint works with He and Zhang. Chapter 3 and Chapter 4 are mainly based on [47], Chapter 5 is mainly based

Table 8.1: Thesis organization and theoretical approximation ratios

Section	Model	Theorem	Approximation performance ratio
3.2	(T_S)	3.2.4	$\left(\prod_{k=1}^{d-2} n_k\right)^{-\frac{1}{2}}$
3.3	(T_Q)	3.3.4	$\left(\prod_{k=1}^{d-2} n_k\right)^{-\frac{1}{2}} \Omega\left(\log^{-(d-1)} \max_{1\le k\le d} m_k\right)$
4.2	(H_S)	4.2.2, 4.2.4	$d!\, d^{-d} n^{-\frac{d-2}{2}}$
4.3	(H_Q)	4.3.1, 4.3.2	$d!\, d^{-d} n^{-\frac{d-2}{2}} \Omega\left(\log^{-(d-1)} m\right)$
4.4.1	(M_S)	4.4.2, 4.4.4	$\begin{cases} \left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}}\right)\left(\frac{\prod_{k=1}^{s-1} {n_k}^{d_k}}{n_{s-1}}\right)^{-\frac{1}{2}} \\ \left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}}\right)\left(\frac{\prod_{k=1}^{s} {n_k}^{d_k}}{{n_s}^2}\right)^{-\frac{1}{2}} \end{cases}$
4.4.2	(M_Q)	4.4.5, 4.4.6	$\begin{cases} \left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}}\right)\left(\frac{\prod_{k=1}^{s-1} {n_k}^{d_k}}{n_{s-1}}\right)^{-\frac{1}{2}} \Omega\left(\log^{-(d-1)} \max_{1\le k\le s} m_k\right) \\ \left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}}\right)\left(\frac{\prod_{k=1}^{s} {n_k}^{d_k}}{{n_s}^2}\right)^{-\frac{1}{2}} \Omega\left(\log^{-(d-1)} \max_{1\le k\le s} m_k\right) \end{cases}$
5.2	(P_S)	5.2.2	$2^{-\frac{5d}{2}}(d+1)!\, d^{-2d}(n+1)^{-\frac{d-2}{2}}$
5.3	(P_Q)	5.3.1	$2^{-\frac{5d}{2}}(d+1)!\, d^{-2d}(n+1)^{-\frac{d-2}{2}} \Omega\left(\log^{-(d-1)} m\right)$
5.4	(P_G)	5.4.2, 5.4.3	$2^{-2d}(d+1)!\, d^{-2d}(n+1)^{-\frac{d-2}{2}}(t^2+1)^{-\frac{d}{2}}$
6.2	(T_B)	6.2.1	$\left(\frac{2}{\pi}\right)^{d-1} \ln\left(1+\sqrt{2}\right)\left(\prod_{k=1}^{d-2} n_k\right)^{-\frac{1}{2}}$
6.3	(H_B)	6.3.2, 6.3.3	$\left(\frac{2}{\pi}\right)^{d-1} \ln\left(1+\sqrt{2}\right) d!\, d^{-d} n^{-\frac{d-2}{2}}$
6.4	(M_B)	6.4.1, 6.4.2	$\begin{cases} \left(\frac{2}{\pi}\right)^{d-1} \ln\left(1+\sqrt{2}\right)\left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}}\right)\left(\frac{\prod_{k=1}^{s-1} {n_k}^{d_k}}{n_{s-1}}\right)^{-\frac{1}{2}} \\ \left(\frac{2}{\pi}\right)^{d-1} \ln\left(1+\sqrt{2}\right)\left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}}\right)\left(\frac{\prod_{k=1}^{s} {n_k}^{d_k}}{{n_s}^2}\right)^{-\frac{1}{2}} \end{cases}$
6.5	(P_B)	6.5.2	$\frac{\ln\left(1+\sqrt{2}\right)}{2(1+e)\pi^{d-1}}(d+1)!\, d^{-2d}(n+1)^{-\frac{d-2}{2}}$
7.2	(T_{BS})	7.2.2	$\left(\frac{2}{\pi}\right)^{\frac{2d-1}{2}}\left(\prod_{k=1}^{d-1} n_k \prod_{\ell=1}^{d'-1} m_\ell\right)^{-\frac{1}{2}}$
7.3	(H_{BS})	7.3.1, 7.3.2	$\left(\frac{2}{\pi}\right)^{\frac{2d-1}{2}} d!\, d^{-d} d'!\, d'^{-d'} n^{-\frac{d-1}{2}} m^{-\frac{d'-1}{2}}$
7.4	(M_{BS})	7.4.1, 7.4.2	$\left(\frac{2}{\pi}\right)^{\frac{2d-1}{2}}\left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}} \prod_{\ell=1}^{t} \frac{d'_\ell!}{{d'_\ell}^{d'_\ell}}\right)\left(\frac{\prod_{k=1}^{s} {n_k}^{d_k} \prod_{\ell=1}^{t} {m_\ell}^{d'_\ell}}{n_s\, m_t}\right)^{-\frac{1}{2}}$

on [48], and Chapter 6 and Chapter 7 are mainly based on [49]. The results not only enhanced approximation algorithms for high degree polynomial optimizations, but also opened up a wide range of new research topics for modeling and novel solution methods. The research works have attracted some follow-up studies on the topic. For instance, So [108] improved the approximation ratios of the models (T_S) and (H_S) to $\Omega\left(\prod_{k=1}^{d-2}\sqrt{\frac{\log n_k}{n_k}}\right)$ and $\Omega\left(\left(\frac{\log n}{n}\right)^{\frac{d-2}{2}}\right)$, respectively. Very recently, He et al. [46] proposed some fairly simple randomized approaches, which improved the approximation ratios of homogeneous polynomial optimizations with spherical constraints and/or binary constraints, and the orders of the ratios were comparable to that in [108]. Apart from the improvements of the approximation ratios, Chen et al. [25] established the tightness result of multilinear form relaxation for the model (H_S), and also derived some local improvement algorithms for solving general polynomial optimization problems, especially for the model (P_Q). Meanwhile, many other research topics are also currently under investigations, including the extensions of polynomial optimization to complex variables; the minimization counterparts of the models discussed in this thesis; the inapproximability results of these models; and of course issues from practical applications of these models.

Bibliography

[1] K. Aardal, C. A. J. Hurkens, and A. K. Lenstra, *Solving a System of Linear Diophantine Equations with Lower and Upper Bounds on the Variables*, Mathematics of Operations Research, 25, 427–442, 2000. 117

[2] J. L. R. Alfonsín, *The Diophantine Frobenius Problem*, Oxford University Press, Oxford, UK, 2005. 117

[3] N. Alon, W. F. de la Vega, R. Kannan, and M. Karpinski, *Random Sampling and Approximation of MAX-CSP Problems*, Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 232–239, 2002. 115

[4] N. Alon, K. Makarychev, Y. Makarychev, and A. Naor, *Quadratic Forms on Graphs*, Inventiones Mathematicae, 163, 499–522, 2006. 99, 103

[5] N. Alon and A. Naor, *Approximating the Cut-Norm via Grothendieck's Inequality*, SIAM Journal on Computing, 35, 787–803, 2006. 5, 8, 27, 98, 100, 101, 102, 115, 118

[6] N. Ansari and E. Hou, *Computational Intelligence for Optimization*, Kluwer Academic Publishers, Norwell, MA, 1997. 5

[7] S. Arora, E. Berger, E. Hazan, G. Kindler, and M. Safra, *On Non-Approximability for Quadratic Programs*, Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, 206–215, 2005. 8

[8] E. Artin, *Über die Zerlegung Definiter Funktionen in Quadrate*, Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg, 5, 100–115, 1927. 2

[9] A. Atamturk, G. L. Nemhauser, and M. W. P. Savelsbergh, *Conflict Graphs in Solving Integer Programming Problems*, European Journal of Operational Research, 121, 40–55, 2000. 5

[10] G. M. de Athayde and R. G. Flôres, Jr., *Incorporating Skewness and Kurtosis in Portfolio Optimization: A Multidimensional Efficient Set*, in S. Satchell and A. Scowcroft (Eds.), Advances in Portfolio Construction and Implementation, Butterworth-Heinemann, Oxford, UK, Chapter 10, 243–257, 2003. 3, 92

[11] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties*, Springer-Verlag, Berlin, Germany, 1999. 22

[12] M. L. Balinski, *On a Selection Problem*, Management Sciences, 17, 230–231, 1970. 5, 7

[13] I. Bárány and Z. Füredi, *Computing the Volume Is Difficult*, Discrete & Computational Geometry, 2, 319–326, 1987. 24

[14] A. Barmpoutis, B. Jian, B. C. Vemuri, and T. M. Shepherd, *Symmetric Positive 4th Order Tensors & Their Estimation from Diffusion Weighted MRI*, Proceedings of the 20th International Conference on Information Processing in Medical Imaging, 308–319, 2007. 2

[15] A. Barvinok, *Integration and Optimization of Multivariate Polynomials by Restriction onto a Random Subspace*, Foundations of Computational Mathematics, 7, 229–244, 2006. 9

[16] D. Beihoffer, J. Hendry, A. Nijenhuis, and S. Wagon, *Faster Algorithms for Frobenius Numbers*, The Electronic Journal of Combinatorics, 12, R27, 2005. 117

[17] S. J. Benson and Y. Ye, *Algorithm 875: DSDP5—Software for Semidefinite Programming*, ACM Transactions on Mathematical Software, 34, Artical 16, 2008. 26

[18] B. Bernhardsson and J. Peetre, *Singular Values of Trilinear Forms*, Experimental Mathematics, 10, 509–517, 2001. 43

[19] B. Borchers, *CSDP, A C Library for Semidefinite Programming*, Optimization Methods and Software, 11, 613–623, 1999. 26

[20] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004. 30, 88, 90

[21] J. Bruck and M. Blaum, *Neural Networks, Error-Correcting Codes, and Polynomials over the Binary n-Cube*, IEEE Transactions on Information Theory, 35, 976–987, 1989. 5

[22] P. H. Calamai and J. J. Moré, *Projected Gradient Methods for Linearly Constrained Problems*, Mathematical Programming, 39, 93–116, 1987. 71

[23] J. D. Carroll and J.-J. Chang, *Analysis of Individual Differences in Multidimensional Scaling via an N-Way Generalization of "Eckart-Young" Decomposition*, Psychometrika, 35, 283–319, 1970. 44

[24] M. Charikar and A. Wirth, *Maximizing Quadratic Programs: Extending Grothendieck's Inequality*, Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, 54–60, 2004. 8, 27, 98, 103

[25] B. Chen, S. He, Z. Li, and S. Zhang, *Maximum Block Improvement and Polynomial Optimization*, Technical Report, Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong, 2011. 30, 71, 79, 96, 137

[26] G. Cornuéjols and M. Dawande, *A Class of Hard Small 0-1 Programs*, Informs Journal of Computing, 11, 205–210, 1999. 117

[27] G. Dahl, J. M. Leinaas, J. Myrheim, and E. Ovrum, *A Tensor Product Matrix Approximation Problem in Quantum Physics*, Linear Algebra and Its Applications, 420, 711–725, 2007. 2, 3, 67, 68

[28] L. De Lathauwer, B. De Moor, and J. Vandewalle, *A Multilinear Singular Value Decomposition*, SIAM Journal on Matrix Analysis and Applications, 21, 1253–1278, 2000. 43

[29] L. De Lathauwer, B. De Moor, and J. Vandewalle, *On the Best Rank-1 and Rank-($R_1, R_2, \cdots, R_N$) Approximation of Higher-Order Tensors*, SIAM Journal on Matrix Analysis and Applications, 21, 1324–1342, 2000. 45

[30] C. N. Delzell, *A Continuous, Constructive Solution to Hilbert's 17th Problem*, Inventiones Mathematicae, 76, 365–384, 1984. 2

[31] M. Dyer, A. M. Frieze, and R. Kannan, *A Random Polynomial-Time Algorithm for Approximating the Volume of Convex Bodies*, Journal of the ACM, 38, 1–17, 1991. 24

[32] U. Feige, *Relations between Average Case Complexity and Approximation Complexity*, Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 534–543, 2002. 5

[33] U. Feige, J. H. Kim, and E. Ofek. *Witnesses for Non-Satisfiability of Dense Random 3CNF Formulas*, Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, 497–508, 2006. 5

[34] U. Feige and E. Ofek, *Easily Refutable Subformulas of Large Random 3CNF Forumlas*, Theory of Computing, 3, 25–43, 2007. 5

[35] J. Friedman, A. Goerdt, and M. Krivelevich, *Recognizing More Unsatisfiable Random k-SAT Instances Efficiently*, SIAM Journal on Computing, 35, 408–430, 2005. 5

[36] A. M. Frieze and R. Kannan, *Quick Approximation to Matrices and Applications*, Combinatorica, 19, 175–200, 1999. 115

[37] K. Fujisawa, M. Kojima, K. Nakata, and M. Yamashita, *SDPA (SemiDefinite Programming Algorithm) User's Manual—Version 6.2.0*, Research Report B-308, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Tokyo, Japan, 1995. 26

[38] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, NY, 1979. 20, 24, 27, 116

[39] A. Ghosh, E. Tsigaridas, M. Descoteaux, P. Comon, B. Mourrain, and R. Deriche, *A Polynomial Based Approach to Extract the Maxima of an Antipodally Symmetric Spherical Function and Its Application to Extract Fiber Directions from the Orientation Distribution Function in Diffusion MRI*, Proceedings of the 11th International Conference on Medical Image Computing and Computer Assisted Intervention, 237–248, 2008. 3, 7

[40] M. X. Goemans and D. P. Williamson, *Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems using Semidefinite Programming*, Journal of the ACM, 42, 1115–1145, 1995. 4, 5, 6, 8, 24, 26, 98, 101, 124, 134

[41] M. Grant and S. Boyd, *CVX: Matlab Software for Disciplined Convex Programming, Version 1.2*, `http://cvxr.com/cvx`, 2010. 69

[42] L. Gurvits, *Classical Deterministic Complexity of Edmonds' Problem and Quantum Entanglement*. Proceedings of the 35th Annual ACM Symposium on Theory of Computing, 10–19, 2003. 3

[43] P. L. Hammer and S. Rudeanu, *Boolean Methods in Operations Research*, Springer-Verlag, New York, NY, 1968. 5

[44] P. Hansen, *Methods of Nonlinear 0-1 Programming*, Annals of Discrete Mathematics, 5, 53–70, 1979. 5

[45] R. A. Harshman, *Foundations of the PARAFAC Procedure: Models and Conditions for an "Explanatory" Multi-Modal Factor Analysis*, UCLA Working Papers in Phonetics, 16, 1–84, 1970. 44

[46] S. He, B. Jiang, Z. Li, and S. Zhang, *Probability Bounds for Polynomial Functions in Random Variables*, Technical Report, Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong, 2011. 137

[47] S. He, Z. Li, and S. Zhang, *Approximation Algorithms for Homogeneous Polynomial Optimization with Quadratic Constraints*, Mathematical Programming, Series B, 125, 353–383, 2010. 135

[48] S. He, Z. Li, and S. Zhang, *General Constrained Polynomial Optimization: An Approximation Approach*, Technical Report, Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong, 2010. 135, 137

[49] S. He, Z. Li, and S. Zhang, *Approximation Algorithms for Discrete Polynomial Optimization*, Technical Report, Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong, 2010. 135, 137

[50] S. He, Z.-Q. Luo, J. Nie, and S. Zhang, *Semidefinite Relaxation Bounds for Indefinite Homogeneous Quadratic Optimization*, SIAM Journal on Optimization, 19, 503–523, 2008. 8, 27, 28, 33, 37, 73, 77

[51] C. Helmberg, *Semidefinite Programming for Combinatorial Optimization*, ZIB-Report 00-34, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Berlin, Germany, 2000. 25

[52] D. Henrion and J. B. Lasserre, *GloptiPoly: Global Optimization over Polynomials with Matlab and SeDuMi*, ACM Transactions on Mathematical Software, 29, 165–194, 2003. 8

[53] D. Henrion, J. B. Lasserre, and J. Loefberg, *GloptiPoly 3: Moments, Optimization and Semidefinite Programming*, Optimization Methods and Software, 24, 761–779, 2009. 8, 70

[54] D. Hilbert, *Über die Darstellung Definiter Formen als Summe von Formenquadraten*, Mathematische Annalen, 32, 342–350, 1888. 1

[55] F. L. Hitchcock, *The Expression of a Tensor or a Polyadic as a Sum of Products*, Journal of Mathematical Physics, 6, 164–189, 1927. 44

[56] F. L. Hitchcock, *Multilple Invariants and Generalized Rank of a p-Way Matrix or Tensor*, Journal of Mathematical Physics, 6, 39–79, 2007. 44

[57] P. M. J. van den Hof, C. Scherer, and P. S. C. Heuberger, *Model-Based Control: Bridging Rigorous Theory and Advanced Technology*, 49–68. Springer-Verlag, Berlin, Germany, 2009. 7

[58] J. J. Hopfield and D. W. Tank, *"Neural" Computation of Decisions in Optimization Problem*, Biological Cybernetics, 52, 141–152, 1985. 5

[59] Y. Huang and S. Zhang, *Approximation Algorithms for Indefinite Complex Quadratic Maximization Problems*, Science China Mathematics, 53, 2697–2708, 2010. 98

[60] E. Jondeau and M. Rockinger, *Optimal Portfolio Allocation under Higher Moments*, European Financial Management, 12, 29–55, 2006. 3, 92

[61] V. Kann, *On the Approximability of NP-Complete Optimization Problems*, Ph.D. Dissertation, Royal Institute of Technology, Stockholm, Sweden, 1992. 22

[62] R. Kannan, *Spectral Methods for Matrices and Tensors*, Proceedings of the 42nd Annual ACM Symposium on Theory of Computing, 1–12, 2010. 115

[63] S. Khot and A. Naor, *Linear Equations Modulo 2 and the L_1 Diameter of Convex Bodies*, Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science, 318–328, 2007. 5, 9, 98, 100, 103

[64] P. M. Kleniati, P. Parpas, and B. Rustem *Partitioning Procedure for Polynomial Optimization: Application to Portfolio Decisions with Higher Order Moments*, COMISEF Working Papers Series, WPS-023, 2009. 3, 4, 92

[65] E. de Klerk, *The Complexity of Optimizing over a Simplex, Hypercube or Sphere: A Short Survey*, Central European Journal of Operations Research, 16, 111–125, 2008. 6

[66] E. de Klerk, M. Laurent, and P. A. Parrilo, *A PTAS for the Minimization of Polynomials of Fixed Degree over the Simplex*, Theoretical Computer Science, 261, 210–225, 2006. 6, 9

[67] E. Kofidis and Ph. Regalia, *On the Best Rank-1 Approximation of Higher Order Supersymmetric Tensors*, SIAM Journal on Matrix Analysis and Applications, 23, 863–884, 2002. 3, 12, 44, 45

[68] T. G. Kolda and B. W. Bader, *Tensor Decompositions and Applications*, SIAM Review, 51, 455–500, 2009. 2, 3, 18, 44, 79, 120

[69] A. Kroó and J. Szabados, *Joackson-Type Theorems in Homogeneous Approximation*, Journal of Approximation Theory, 152, 1–19, 2008. 3, 30

[70] J. B. Lasserre, *Global Optimization with Polynomials and the Problem of Moments*, SIAM Journal on Optimization, 11, 796–817, 2001. 2, 7, 70

[71] J. B. Lasserre, *Polynomials Nonnegative on a Grid and Discrete Representations*, Transactions of the American Mathematical Society, 354, 631–649, 2002. 2, 7, 70

[72] M. Laurent, *Sums of Squares, Moment Matrices and Optimization over Polynomials*, in M. Putinar and S. Sullivant (Eds.), Emerging Applications of Algebraic Geometry, The IMA Volumes in Mathematics and its Applications, 149, 1–114, 2009. 8

[73] C. Ling, J. Nie, L. Qi, and Y. Ye, *Biquadratic Optimization over Unit Spheres and Semidefinite Programming Relaxations*, SIAM Journal on Optimization, 20, 1286–1310, 2009. 9, 27, 51, 60, 62, 73

[74] C. Ling, X. Zhang, and L. Qi, *Semidefinite Relaxation Approximation for Multivariate Bi-Quadratic Optimization with Quadratic Constraints*, Numerical Linear Algebra with Applications, DOI: 10.1002/nla.781, 2011. 9, 51

[75] Z.-Q. Luo, N. D. Sidiropoulos, P. Tseng, and S. Zhang, *Approximation Bounds for Quadratic Optimization with Homogeneous Quadratic Constraints*, SIAM Journal on Optimization, 18, 1–28, 2007. 8, 27, 69, 73, 77

[76] Z.-Q. Luo, J. F. Sturm, and S. Zhang, *Multivariate Nonnegative Quadratic Mappings*, SIAM Journal on Optimization, 14, 1140–1162, 2004. 3

[77] Z.-Q. Luo and S. Zhang, *A Semidefinite Relaxation Scheme for Multivariate Quartic Polynomial Optimization with Quadratic Constraints*, SIAM Journal on Optimization, 20, 1716–1736, 2010. 4, 9, 27, 34, 50, 51, 73, 77, 93

[78] B. B. Mandelbrot and R. L. Hudson, *The (Mis)Behavior of Markets: A Fractal View of Risk, Ruin, and Reward*, Basic Books, New York, NY, 2004. 3, 92

[79] B. Maricic, Z.-Q. Luo, and T. N. Davidson, *Blind Constant Modulus Equalization via Convex Optimization*, IEEE Transactions on Signal Processing, 51, 805–818, 2003. 2

[80] D. Maringer and P. Parpas, *Global Optimization of Higher Order Moments in Portfolio Selection*, Journal of Global Optimization, 43, 219–230, 2009. 7

[81] H. M. Markowitz, *Portfolio Selection*, Journal of Finance, 7, 79–91, 1952. 2, 3, 92

[82] C. A. Micchelli and P. Olsen, *Penalized Maximum-Likelihood Estimation, the Baum-Welch Algorithm, Diagonal Balancing of Symmetric Matrices and Applications to Training Acoustic Data*, Journal of Computational and Applied Mathematics, 119, 301–331, 2000. 3

[83] G. L. Miller, *Riemann's Hypothesis and Tests for Primality*, Journal of Computer and System Sciences, 13, 300–317, 1976. 23

[84] B. Mourrain and J. P. Pavone, *Subdivision Methods for Solving Polynomial Equations*, Journal of Symbolic Computation, 44, 292–306, 2009. 7

[85] B. Mourrain and P. Trébuchet, *Generalized Normal Forms and Polynomial System Solving*. Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation, 253–260, 2005. 7

[86] A. Nemirovski, *Lectures on Modern Convex Optimization*, The H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, 2005. 30, 88, 90

[87] A. Nemirovski, C. Roos, and T. Terlaky, *On Maximization of Quadratic Form over Intersection of Ellipsoids with Common Center*, Mathematical Programming, Series A, 86, 463–473, 1999. 8, 27, 28, 30, 37, 42, 50, 73, 77

[88] Yu. Nesterov, *Semidefinite Relaxation and Nonconvex Quadratic Optimization*, Optimization Methods and Software, 9, 141–160, 1998. 5, 8, 27, 73, 98, 124

[89] Yu. Nesterov, *Squared Functional Systems and Optimization Problems*, in H. Frenk, K. Roos, T. Terlaky, and S. Zhang (Eds.), High Performance Optimization, Kluwer Academic Press, Dordrecht, The Netherlands, 405–440, 2000. 8

[90] Yu. Nesterov, *Random Walk in a Simplex and Quadratic Optimization over Convex Polytopes*, CORE Discussion Paper 2003/71, Université catholique de Louvain, Louvain-la-Neuve, Belgium, 2003. 5, 6, 31, 50, 60, 74

[91] Q. Ni, L. Qi, and F. Wang, *An Eigenvalue Method for Testing Positive Definiteness of a Multivariate Form*, IEEE Transactions on Automatic Control, 53, 1096–1107, 2008. 3, 66

[92] P. Parpas and B. Rustem, *Global Optimization of the Scenario Generation and Portfolio Selection Problems*, Proceedings of the International Conference on Computational Science and Its Applications, 908–917, 2006. 7

[93] P. A. Parrilo, *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*, Ph.D. Dissertation, California Institute of Technology, Pasadena, CA, 2000. 2, 7

[94] P. A. Parrilo, *Semidefinite Programming Relaxations for Semialgebraic Problems*, Mathematical Programming, Series B, 96, 293–320, 2003. 2, 7

[95] L. Peng and M. W. Wong, *Compensated Compactness and Paracommutators*, Journal of the London Mathematical Society, 62, 505–520, 2000. 43

[96] A. J. Prakash, C.-H. Chang, and T. E. Pactwa, *Selecting a Portfolio with Skewness: Recent Evidence from US, European, and Latin American Equity Markets*, Journal of Banking & Finance, 27, 1375–1390, 2003. 3, 92

[97] M. Purser, *Introduction to Error-Correcting Codes*, Artech House, Norwood, MA, 1995. 5

[98] L. Qi, *Extrema of a Real Polynomial*, Journal of Global Optimization, 30, 405–433, 2004. 7, 50, 66

[99] L. Qi, *Eigenvalues of a Real Supersymmetric Tensor*, Journal of Symbolic Computation, 40, 1302–1324, 2005. 3, 66

[100] L. Qi, *Eigenvalues and Invariants of Tensors*, Journal of Mathematical Analysis and Applications, 325, 1363–1377, 2007. 3, 66

[101] L. Qi and K. L. Teo, *Multivariate Polynomial Minimization and Its Applications in Signal Processing*, Journal of Global Optimization, 26, 419–433, 2003. 2

[102] L. Qi, Z. Wan, and Y.-F. Yang, *Global Minimization of Normal Quadratic Polynomials Based on Global Descent Directions*, SIAM Journal on Optimization, 15, 275–302, 2004. 7

[103] L. Qi, F. Wang, and Y. Wang, *Z-eigenvalue Methods for a Global Polynomial Optimization Problem*, Mathematical Programming, Series A, 118, 301–316, 2009. 7

[104] M. O. Rabin, *Probabilistic Algorithms*, in J. F. Traub (Eds.), Algorithms and Complexity: New Directions and Recent Results, Academic Press, New York, NY, 21–39, 1976. 23

[105] M. O. Rabin, *Probabilistic Algorithm for Testing Primality*, Journal of Number Theory, 12, 128–138, 1980. 23

[106] J. M. W. Rhys, *A Selection Problem of Shared Fixed Costs and Network Flows*, Management Science, 17, 200–207, 1970. 5, 7

[107] A. P. Roberts and M. M. Newmann, *Polynomial Optimization of Stochastic Feedback Control for Stable Plants*, IMA Journal of Mathematical Control & Information, 5, 243–257, 1988. 2

[108] A. M.-C. So, *Deterministic Approximation Algorithms for Sphere Constrained Homogeneous Polynomial Optimization Problems*, Mathematical Programming, Series B, DOI: 10.1007/s10107-011-0464-0, 2011. 79, 137

[109] A. M.-C. So, Y. Ye, and J. Zhang, *A Unified Theorem on SDP Rank Reduction*, Mathematics of Operations Research, 33, 910–920, 2008. 40

[110] S. Soare, J. W. Yoon, and O. Cazacu, *On the Use of Homogeneous Polynomials to Develop Anisotropic Yield Functions with Applications to Sheet Forming*, International Journal of Plasticity, 24, 915–944, 2008. 3

[111] R. M. Solovay and V. Strassen, *A Fast Monte-Carlo Test for Primality*, SIAM Journal on Computing, 6, 84–85, 1977. 23

[112] J. F. Sturm, *SeDuMi 1.02, A Matlab Toolbox for Optimization over Symmetric Cones*, Optimization Methods and Software, 11 & 12, 625–653, 1999. 8, 26

[113] J. F. Sturm and S. Zhang, *On Cones of Nonnegative Quadratic Functions*, Mathematics of Operations Research, 28, 246–267, 2003. 67, 74

[114] W. Sun and Y.-X. Yuan, *Optimization Theory and Methods: Nonlinear Programming*, Springer-Verlag, New York, NY, 2006. 74

[115] K. C. Toh, M. J. Todd, and R. H. Tütüncü, *SDPT3—A Matlab Software Package for Semidefinite Programming, Version 1.3*, Optimization Methods and Software, 11, 545–581, 1999. 8, 26

[116] L. Vandenberghe and S. Boyd, *Semidefinite Programming*, SIAM Review, 38, 49–95, 1996. 25

[117] P. P. Varjú, *Approximation by Homogeneous Polynomials*, Constructive Approximation, 26, 317–337, 2007. 3, 30

[118] Y. Ye, *Approximating Quadratic Programming with Bound and Quadratic Constraints*, Mathematical Programming, 84, 219–226, 1999. 8, 27

[119] Y. Ye, *Approximating Global Quadratic Optimization with Convex Quadratic Constraints*, Journal of Global Optimization, 15, 1–17, 1999. 8, 27

[120] S. Zhang, *Quadratic Maximization and Semidefinite Relaxation*, Mathematical Programming, Series A, 87, 453–465, 2000. 8, 27, 73

[121] S. Zhang and Y. Huang, *Complex Quadratic Optimization and Semidefinite Programming*, SIAM Journal on Optimization, 16, 871–890, 2006. 8, 27

[122] T. Zhang and G. H. Golub, *Rank-One Approximation to High Order Tensors*, SIAM Journal on Matrix Analysis and Applications, 23, 534–550, 2001. 7, 45

[123] X. Zhang, C. Ling, and L. Qi, *Semidefinite Relaxation Bounds for Bi-Quadratic Optimization Problems with Quadratic Constraints*, Journal of Global Optimization, 49, 293–311, 2011. 9, 51