SPRINGER BRIEFS IN OPTIMIZATION

Zhening Li
Simai He
Shuzhong Zhang

Approximation Methods for Polynomial Optimization

Springer

Zhening Li • Simai He • Shuzhong Zhang

Approximation Methods for Polynomial Optimization

Models, Algorithms, and Applications

Springer

Zhening Li
Department of Mathematics
Shanghai University
Shanghai
China

Simai He
Department of Management Sciences
City University of Hong Kong
Kowloon Tong
Hong Kong

Shuzhong Zhang
Industrial and Systems Engineering
University of Minnesota
Minneapolis, MN
USA

ISSN 2190-8354 ISSN 2191-575X (electronic)
ISBN 978-1-4614-3983-7 ISBN 978-1-4614-3984-4 (eBook)
DOI 10.1007/978-1-4614-3984-4
Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2012936832

Mathematics Subject Classification (2010): 90C59, 68W25, 65Y20, 15A69, 15A72, 90C26, 68W20, 90C10, 90C11

Preface

Polynomial optimization, as its name suggests, is used to optimize a generic multivariate polynomial function, subject to some suitable polynomial equality and/or inequality constraints. Such problem formulation dates back to the nineteenth century when the relationship between nonnegative polynomials and sum of squares (SOS) was discussed by Hilbert. Polynomial optimization is one of the fundamental problems in Operations Research and has applications in a wide range of areas, including biomedical engineering, control theory, graph theory, investment science, material science, numerical linear algebra, quantum mechanics, signal processing, speech recognition, among many others. This brief discusses some important subclasses of polynomial optimization models arising from various applications. The focus is on optimizing a high degree polynomial function over some frequently encountered constraint sets, such as the Euclidean ball, the Euclidean sphere, intersection of co-centered ellipsoids, binary hypercube, general convex compact set, and possibly a combination of the above constraints. All the models under consideration are NP-hard in general. In particular, this brief presents a study on the design and analysis of polynomial-time approximation algorithms, with guaranteed worst-case performance ratios. We aim at deriving the worst-case performance/approximation ratios that are solely dependent on the problem dimensions, meaning that they are independent of any other types of the problem parameters or input data. The new techniques can be applied to solve even broader classes of polynomial/tensor optimization models. Given the wide applicability of the polynomial optimization models, the ability to solve such models—albeit approximately—is clearly beneficial. To illustrate how such benefits might be, we present a variety of examples in this brief so as to showcase the potential applications of polynomial optimization.

Shanghai, China	Zhening Li
Kowloon Tong, Hong Kong	Simai He
Minnesota, MN, USA	Shuzhong Zhang

Contents

Chapter 1
Introduction

Polynomial optimization is to optimize a polynomial function subject to polynomial equality and/or inequality constraints, specifically, the following generic optimization model:

$$\begin{array}{lll}(PO) & \min & p(\boldsymbol{x}) \\ & \text{s.t.} & f_i(\boldsymbol{x}) \le 0,\ i = 1,2,\dots,m_1, \\ & & g_j(\boldsymbol{x}) = 0,\ j = 1,2,\dots,m_2, \\ & & \boldsymbol{x} = (x_1,x_2,\dots,x_n)^{\mathrm{T}} \in \mathbb{R}^n,\end{array}$$

where $p(\boldsymbol{x})$, $f_i(\boldsymbol{x})$ $(i = 1,2,\dots,m_1)$ and $g_j(\boldsymbol{x})$ $(j = 1,2,\dots,m_2)$ 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 (PO), and specialized software packages have been developed.

1.1 History

The modern history of polynomial optimization may date back to the nineteenth century when the relationship between nonnegative polynomial function and the sum of squares (SOS) of polynomials was studied. Given a multivariate polynomial function that takes only nonnegative values over the real domain, can it be represented as an SOS of polynomial functions? Hilbert [51] gave a concrete answer in 1888, which asserted that the only cases for a nonnegative polynomial to be a SOS are: univariate polynomials; multivariate quadratic polynomials; and bivariate quartic polynomials. Later, the issue about nonnegative polynomials was formulated in Hilbert's 17th problem—one of the famous 23 problems that Hilbert addressed in his celebrated speech in 1900 at the Paris conference of the International Congress of Mathematicians. Hilbert conjectured that a nonnegative polynomial entails expression of definite rational functions as quotients of two sums of squares.

To be precise, the question is: Given a multivariate polynomial function that takes only nonnegative values over the real numbers, can it be represented as an SOS of rational functions? This was solved in the affirmative, by Artin [8] in 1927. A constructive algorithm was later found by Delzell [29] in 1984. About 10 years ago, Lasserre [67, 68] and Parrilo [93, 94] proposed a method called the SOS to solve general polynomial optimization problems. The method is based on the fact that deciding whether a given polynomial is an SOS 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.

1.1.1 Applications

Polynomial optimization has 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 (PO), 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 (PO), especially for high degree polynomial optimization.

Polynomial optimization has immediate applications in investment science. For instance, the celebrated mean–variance model was proposed by Markowitz [80] 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 [105] studied polynomial optimization of stochastic feedback control for stable plants. In diffusion magnetic resonance imaging (MRI), Barmpoutis et al. [13] presented a case for the fourth order tensor approximation. In fact, there are a large class of (PO) arising from tensor approximations and decompositions, which are originated from applications in psychometrics and chemometrics (see an excellent survey by Kolda and Bader [65]). Polynomial optimization also has applications in sinal processing. Maricic et al. [78] 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. [26] proposed a polynomial optimization model to verify whether a physical system is entangled or not, which is an important problem in quantum physics. Gurvits [40] showed that the entanglement verification is NP-hard in general. In fact, the model discussed in [26] is related to the nonnegative quadratic mappings studied by Luo et al. [75].

Among generic polynomial functions, homogeneous polynomials play an important role in approximation theory (see, e.g., two recent papers by Kroó and Szabados [66] and Varjú [113]). Essentially their results state that the homogeneous polynomial functions are fairly "dense" among continuous functions in a certain well-defined sense. As such, optimization of homogeneous polynomials becomes important. As an example, Ghosh et al. [38] formulated a fiber-detection problem in diffusion MRI by maximizing a homogenous polynomial function subject to the Euclidean spherical constraint, i.e.,

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

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) plays an important role in the following examples. In material sciences, Soare et al. [108] proposed some 4th-, 6th-, and 8th-order homogeneous polynomials to model the plastic anisotropy of orthotropic sheet metal. In statistics, Micchelli and Olsen [81] 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 Lim [71] and Qi [99]). Another widely used application of (H_S) regards the best rank-one approximation of higher order tensors (see [64, 65]).

In fact, Markowitz's mean–variance model [80] 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 optimization. Mandelbrot and Hudson [77] 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 [56], and Kleniati et al. [60]. However, in those models, the polynomial functions involved are no longer homogeneous. In particular, a very general model in [60] is

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

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 [76]. 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 optimization over variables in discrete values, in particular binary variables, is also widely studied. For example, maximize a polynomial function over variables picking from 1 or -1, i.e.,

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

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 [39]. 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), 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., [41, 42]). It is also known as the Fourier support graph problem. Mathematically, a polynomial function $p : \{-1,1\}^n \to \mathbb{R}$ has Fourier expansion $p(\boldsymbol{x}) = \sum_{S\subset\{1,2,\ldots,n\}} \hat{p}(S) \prod_{i\in S} x_i$, which is also called the Fourier support graph. By assuming that $p(\boldsymbol{x})$ has only succinct (polynomially many) nonzero 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 brief. 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, 104]). The transformation uses the concept of a *conflict graph* of a 0-1 polynomial function, for details, one is referred to [9, 21]. Beyond its connection to the graph problems, (P_B) also has applications in neural networks [6, 21, 54], 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 [59] show that it has applications in the problem of refutation of random k-CNF formulas [31–34].

If the objective polynomial function in (P_B) is homogeneous, likewise, the homogeneous quadratic case has been studied extensively, e.g., [5, 39, 87, 89].

Homogeneous cubic polynomial case is also discussed by Khot and Naor [59]. 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.,

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

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 the optimal value of the following problem:

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

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 (PO) extend polynomial integer programming problems to the mixed integer programming problems, which is also an important subclass of (PO) studied in this brief.

1.1.2 Algorithms

Polynomial optimization problems are typically non-convex and highly nonlinear. In most cases, (PO) is NP-hard, even for very special instances, such as maximizing a cubic polynomial over a sphere (see Nesterov [89]), maximizing a quadratic form in binary variables (see, e.g., Goemans and Williamson [39]), etc. The reader is referred to De Klerk [61] 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. [63]), 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 most immediate attempt for solving polynomial optimization problems is to simply regard 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., [30,38,119]). In [38], the authors develop special algorithms for that purpose, such as subdivision methods proposed by Mourrain and Pavone [83], and generalized normal forms algorithms designed by Mourrain and Trébuchet [84]. 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 studied by Qi et al. [103]. Moreover, Parpas and Rustem [92] and Maringer and Parpas [79] 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, 104]), by using the concept of a conflict graph of a 0-1 polynomial function.

The so-called SOS method has been one major systematic approach for solving general polynomial optimization problems. The method was introduced by Lasserre [67, 68] and Parrilo [93, 94], and a significant amount of research on the SOS method has been conducted in the past 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 (PO) to any given accuracy. For univariate polynomial optimization, Nesterov [88] 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 SOS. In general, however, the SDP problems required to be solved by the SOS method may grow very large, and are not practical when the program dimension goes high. At any rate, thanks to the recently developed efficient SDP solvers (e.g., SeDuMi of Sturm [109], SDPT3 of Toh et al. [112]), the SOS method appears to be attractive. Henrion and Lasserre [49] developed a specialized tool known as GloptiPoly (the latest version, GloptiPoly 3, can be found in Henrion et al. [50]) for finding a global optimal solution of the polynomial optimization problems arising from the SOS method, based on Matlab and SeDuMi. For an overview on the recent theoretical developments, we refer to the excellent survey by Laurent [69].

Along a different line, the intractability of general polynomial optimization also 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 SDP relaxation and randomization approach for the max-cut problem due to Goemans and Williamson [39], 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 [39] has been generalized subsequently by many authors, including Nesterov [87], Ye [115, 116], Nemirovski et al. [86], Zhang [117], Charikar and Wirth [23], Alon and Naor [5], Zhang and Huang [118], Luo et al. [74], and He et al. [48]. In particular, when the matrix $\boldsymbol{F}$ is only known to be positive semidefinite, Nesterov [87] 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 [23] derived an $\Omega(1/\ln n)$-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 optimization, Nemirovski et al. [86] proposed an $\Omega(1/\ln m)$-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. [74] and He et al. [48].

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. [63] 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 Euclidean sphere. Later, Barvinok [14] showed that optimizing a certain class of polynomials over the Euclidean sphere also admits a randomized PTAS. Note that the results in [14, 63] apply only when the objective polynomial has some special structure. A quite general result is due to Khot and Naor [59], where they showed how to estimate the optimal value of the problem $\max_{\boldsymbol{x}\in\{1,-1\}^n}\sum_{1\le i,j,k\le n}F_{ijk}x_ix_jx_k$ 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 $\Omega(\sqrt{\frac{\ln n}{n}})$ times the optimal value. Two recent papers (Luo and Zhang [76] and Ling et al. [72]) 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 optimization, which are essentially the main subject in this brief. Luo and Zhang [76] considered quartic optimization, and showed that optimizing a homogenous quartic form over the intersection of some co-centered ellipsoids can be relaxed 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. [72] considered a special quartic optimization model. Basically, the problem is to minimize a biquadratic function over two spherical constraints. In [72], approximate solutions as well as exact solutions using the SOS method are considered. The approximation bounds in [72] are indeed comparable to the bound in [76], although they are dealing with two different models. Very recently, Zhang et al. [120] and Ling al. [73] 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. Zhang et al. [121] also studied homogeneous cubic polynomial optimization over spherical constraints, and derived some approximation bound.

However, for (PO) with an arbitrary degree polynomial objective, the approximation results remained nonexistent until recently. He et al. [46] proposed a first polynomial-time approximation algorithm for optimizing any fixed degree homogeneous polynomial with quadratic constraints. This has set off a flurry of research activities. In a subsequent paper, He et al. [45] generalized the approximation methods and proposed first polynomial-time approximation algorithm for optimizing any fixed degree inhomogeneous polynomial function over a general convex set. Note that this is the only approximation result for optimizing any degree *inhomogeneous* polynomial function. So [106] improved some of the approximation bounds in [45], for the case of optimizing any fixed degree homogeneous polynomial with spherical constraints. Along a different line, He et al. [47] studied any degree polynomial integer programming and mixed integer programming. In particular, they proposed polynomial-time approximation algorithms for polynomial optimization with binary constraints and polynomial optimization with spherical and binary constraints. The results of He, Li, and Zhang were summarized in the recent Ph.D. thesis of Li [70], which forms a basis for this brief.

1.2 Contributions

This brief presents a systematic study on approximation methods for optimizing any fixed degree polynomial function over some important and widely used constraint sets, e.g., the Euclidean ball, the Euclidean sphere, hypercube, binary hypercube, intersection of co-centered ellipsoids, a general convex compact set, and even a mixture of them. The objective polynomial function ranges from multilinear tensor function, homogeneous polynomial, and generic inhomogeneous polynomial function. With combination of the constraint sets, the models constitute most of the subclasses of (PO) in real applications. The detailed description of the models studied is listed in Sect. 1.3.3, or specifically Table 1.1. 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. The application of these polynomial optimization models will be discussed. Specifically, our contributions are highlighted as follows:

1. We propose approximation algorithms for optimization of any fixed degree homogeneous polynomial over the Euclidean ball, 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 optimization, our approximation ratios improve the previous ones, when specialized to their particular degrees.
2. We establish key linkages between multilinear functions and homogeneous polynomials, and thus establish the same approximation ratios for homogeneous polynomial optimization with their multilinear form relaxation problems.
3. We propose a general scheme to handle inhomogeneous polynomial optimization through the method of homogenization, and thus establish the same approximation ratios (in the sense of relative approximation ratio) for inhomogeneous polynomial optimization with their homogeneous polynomial relaxation problems. It is the first approximation bound of approximation algorithms for general inhomogeneous polynomial optimization with a high degree.
4. We propose several decomposition routines for polynomial optimization over different types of constraint sets, and derive approximation bounds for multilinear function optimization with their lower degree relaxation problems, based on which we derive approximation algorithms for polynomial optimization over various constraint sets.
5. With the availability of our proposed approximation methods, we illustrate some potential modeling opportunities with the new optimization models.

The whole brief is organized as follows. In the remainder of this chapter (Sects. 1.3 and 1.4), we introduce the notations and various polynomial optimization models studied in this brief, followed by some necessary preparations, e.g., definitions of approximation algorithms and approximation ratios, various tensor operations, etc. The main part of the brief is the dealing with approximation methods for the polynomial optimization models concerned, which will be the contents of Chaps. 2 and 3. In particular, in Chap. 2, we elaborate on polynomial optimization over the Euclidean ball: we propose various techniques step by step in handling different types of objective polynomial functions, and discuss how these steps leads to the final approximation algorithm for optimizing any fixed degree inhomogeneous polynomial function over the Euclidean ball. Chapter 3 deals with various technical extensions of the approximation methods proposed in Chap. 2, armed with which we propose approximation algorithms for solving many other important polynomial optimization models. Sample applications of the polynomial optimization models and their approximation algorithms will be the topic for Chap. 4. Finally, in Chap. 5 we conclude this brief by tabulating the approximation ratios developed in the brief so as to provide an overview of the approximation results and the context; other methods related to approximation algorithms of polynomial optimization models are also commented on, including a discussion of the recent developments and possible future research topics.

1.3 Notations and Models

Throughout this brief, 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 $\boldsymbol{Q}$).

1.3.1 Objective Functions

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

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

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. In other words, the notions of multilinear form and tensor are exchangeable. The meaning of multilinearity is that if one fixes $(\boldsymbol{x}^2,\boldsymbol{x}^3,\ldots,\boldsymbol{x}^d)$ in the function F, then it is a linear function in $\boldsymbol{x}^1$, 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 $\boldsymbol{F} = (F_{i_1 i_2\cdots i_d})$ *supersymmetric* (see [64]), if any of its components $F_{i_1 i_2\cdots i_d}$ is invariant under all permutations of $\{i_1, i_2, \dots, i_d\}$. 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 supersymmetric 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 supersymmetric tensor can be written as $\boldsymbol{F} = (F_{i_1 i_2\cdots i_d}) \in \mathbb{R}^{n^d}$, with $F_{i_1 i_2\cdots i_d} \equiv F'_{i_1 i_2\cdots i_d}/|\Pi(i_1, i_2, \dots, i_d)|$, where $|\Pi(i_1, i_2, \dots, i_d)|$ is the number of distinctive permutations of the indices $\{i_1, i_2, \dots, i_d\}$. This supersymmetric tensor representation is indeed unique. Let F be its corresponding multilinear form defined by the supersymmetric tensor $\boldsymbol{F}$, then we have $f(\boldsymbol{x}) = F(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \dots, \boldsymbol{x}}_{d})$. The letter "H" here is used to emphasize that the polynomial function in question is *homogeneous*.

We shall also consider in this brief the following mixed form:

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

where $d_1 + d_2 + \cdots + d_s = d$, $\boldsymbol{x}^k \in \mathbb{R}^{n_k}$ for $k = 1, 2, \dots, s$, 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 $(\boldsymbol{x}^2, \boldsymbol{x}^3, \dots, \boldsymbol{x}^s)$, $F(\underbrace{\cdot, \cdot, \dots, \cdot}_{d_1}, \underbrace{\boldsymbol{x}^2, \boldsymbol{x}^2, \dots, \boldsymbol{x}^2}_{d_2}, \dots, \underbrace{\boldsymbol{x}^s, \boldsymbol{x}^s, \dots, \boldsymbol{x}^s}_{d_s})$ is a supersymmetric d_1th order tensor form, and so on.

Beyond the homogeneous polynomial functions (multilinear form, homogeneous form, and mixed forms) described above, we also study in this brief the generic multivariate inhomogeneous polynomial function. An n-dimensional d-th degree polynomial function can be explicitly written as a summation of homogenous forms in decreasing degrees as follows:

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

where $\boldsymbol{x} \in \mathbb{R}^n$, $f_0 \in \mathbb{R}$, and $f_k(\boldsymbol{x}) = F_k(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \dots, \boldsymbol{x}}_{k})$ is a homogenous form of degree k for $k = 1, 2, \dots, d$; 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 brief, 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 brief, 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 $\bar{\boldsymbol{x}} = (x_1, x_2, \ldots, x_n, x_{n+1})^{\mathrm{T}} \in \mathbb{R}^{n+1}$, then $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}} \in \mathbb{R}^n$ 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 form (Function H) or a mixed form (Function M), and p for a generic inhomogeneous polynomial (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.

1.3.2 Constraint Sets

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

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

The notion "B" signifies the *binary* variables or *binary* constraints, and "S" signifies the Euclidean *spherical* constraint, with "B̄" (hypercube) and "S̄" (the Euclidean ball) signifying their *convex hulls*, respectively. The norm notation "$\|\cdot\|$" in this brief 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

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

The notion "Q" signifies the *quadratic* constraints, and we focus on convex quadratic constraints in this brief, 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 brief, which is denoted by the notion "G". Constraints B̄, S̄, 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 $\bar{\text{S}}$ and Constraint $\bar{\text{B}}$ as well.

1.3.3 Models and Organization

All the polynomial optimization models discussed in this brief 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 Sect. 1.3.1 and the names of the constraint sets described in Sect. 1.3.2, with the names of the constraints in the subscription. For example, model (T_S) is to maximize a multilinear form (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.

Chapter 2 is concerned with the approximation methods for optimizing a multilinear form, a homogenous form, a mixed form, and an inhomogeneous polynomial over the Euclidean ball, i.e., $(T_{\bar{S}})$, $(H_{\bar{S}})$, $(M_{\bar{S}})$, and $(P_{\bar{S}})$. Chapter 3 deals with various polynomial optimization over other constraint sets. In particular, Sect. 3.1 deals with polynomial optimization over hypercube or binary hypercube, i.e., (T_B), (H_B), (M_B), (P_B), $(T_{\bar{B}})$, $(H_{\bar{B}})$, $(M_{\bar{B}})$, and $(P_{\bar{B}})$; Sect. 3.2 deals with homogeneous polynomial optimization over the Euclidean sphere, i.e., (T_S), (H_S), and (M_S); Sect. 3.3 deals with polynomial optimization over intersection of co-centered ellipsoids, i.e., (T_Q), (H_Q), (M_Q), and (P_Q); Sect. 3.4 deals with polynomial optimization over a general convex compact set, i.e., (P_G); and Sect. 3.5 deals with homogeneous polynomial optimization over binary hypercube and the Euclidean sphere, i.e., (T_{BS}), (H_{BS}), and (P_{BS}). The details of the models are listed in Table 1.1 for a quick reference.

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}). For all the polynomial optimization models in Table 1.1, we discuss its computational complexity, and focus on polynomial-time approximation algorithms with worst-case performance ratios. 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 models concerned.

Table 1.1 Description of polynomial optimization models

Section	Model	Objective to be maximized	Constraint set
2.1	$(T_{\bar{S}})$	$F(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^d)$	$\boldsymbol{x}^k \in \bar{\mathbb{S}}^{n_k}, k=1,2,\ldots,d$
2.2	$(H_{\bar{S}})$	$f(\boldsymbol{x})$	$\boldsymbol{x} \in \bar{\mathbb{S}}^n$
2.3	$(M_{\bar{S}})$	$f(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^s)$	$\boldsymbol{x}^k \in \bar{\mathbb{S}}^{n_k}, k=1,2,\ldots,s$
2.4	$(P_{\bar{S}})$	$p(\boldsymbol{x})$	$\boldsymbol{x} \in \bar{\mathbb{S}}^n$
3.1	(T_B)	$F(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^d)$	$\boldsymbol{x}^k \in \mathbb{B}^{n_k}, k=1,2,\ldots,d$
	(H_B)	$f(\boldsymbol{x})$	$\boldsymbol{x} \in \mathbb{B}^n$
	(M_B)	$f(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^s)$	$\boldsymbol{x}^k \in \mathbb{B}^{n_k}, k=1,2,\ldots,s$
	(P_B)	$p(\boldsymbol{x})$	$\boldsymbol{x} \in \mathbb{B}^n$
	$(T_{\bar{B}})$	$F(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^d)$	$\boldsymbol{x}^k \in \bar{\mathbb{B}}^{n_k}, k=1,2,\ldots,d$
	$(H_{\bar{B}})$	$f(\boldsymbol{x})$	$\boldsymbol{x} \in \bar{\mathbb{B}}^n$
	$(M_{\bar{B}})$	$f(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^s)$	$\boldsymbol{x}^k \in \bar{\mathbb{B}}^{n_k}, k=1,2,\ldots,s$
	$(P_{\bar{B}})$	$p(\boldsymbol{x})$	$\boldsymbol{x} \in \bar{\mathbb{B}}^n$
3.2	(T_S)	$F(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^d)$	$\boldsymbol{x}^k \in \mathbb{S}^{n_k}, k=1,2,\ldots,d$
	(H_S)	$f(\boldsymbol{x})$	$\boldsymbol{x} \in \mathbb{S}^n$
	(M_S)	$f(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^s)$	$\boldsymbol{x}^k \in \mathbb{S}^{n_k}, k=1,2,\ldots,s$
3.3	(T_Q)	$F(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^d)$	$(\boldsymbol{x}^k)^{\mathrm{T}}\boldsymbol{Q}^k_{i_k}\boldsymbol{x}^k \le 1, k=1,2,\ldots,d,$ $i_k=1,2,\ldots,m_k$ $\boldsymbol{x}^k \in \mathbb{R}^{n_k}, k=1,2,\ldots,d$
	(H_Q)	$f(\boldsymbol{x})$	$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x} \le 1, i=1,2,\ldots,m$ $\boldsymbol{x} \in \mathbb{R}^n$
	(M_Q)	$f(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^s)$	$(\boldsymbol{x}^k)^{\mathrm{T}}\boldsymbol{Q}^k_{i_k}\boldsymbol{x}^k \le 1, k=1,2,\ldots,s,$ $i_k=1,2,\ldots,m_k$ $\boldsymbol{x}^k \in \mathbb{R}^{n_k}, k=1,2,\ldots,s$
	(P_Q)	$p(\boldsymbol{x})$	$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x} \le 1, i=1,2,\ldots,m$ $\boldsymbol{x} \in \mathbb{R}^n$
3.4	(P_G)	$p(\boldsymbol{x})$	$\boldsymbol{x} \in G$
3.5	(T_{BS})	$F(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^d,\boldsymbol{y}^1,\boldsymbol{y}^2,\ldots,\boldsymbol{y}^{d'})$	$\boldsymbol{x}^k \in \mathbb{B}^{n_k}, k=1,2,\ldots,d$ $\boldsymbol{y}^\ell \in \mathbb{S}^{m_\ell}, \ell=1,2,\ldots,d'$
	(H_{BS})	$f(\boldsymbol{x},\boldsymbol{y})$	$\boldsymbol{x} \in \mathbb{B}^n$ $\boldsymbol{y} \in \mathbb{S}^m$
	(M_{BS})	$f(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^s,\boldsymbol{y}^1,\boldsymbol{y}^2,\ldots,\boldsymbol{y}^t)$	$\boldsymbol{x}^k \in \mathbb{B}^{n_k}, k=1,2,\ldots,s$ $\boldsymbol{y}^\ell \in \mathbb{S}^{m_\ell}, \ell=1,2,\ldots,t$

1.4 Preliminary

In this last section, we shall try to get necessary preparations for the main contents to come. The topics include some basics of tensor operations, approximation algorithms, and randomized algorithms. We shall also present the SDP relaxation and randomization techniques, which are helpful to understand the main ideas underlying the approximation methods in this brief.

1.4.1 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 subsection describes a few tensor operations commonly used in this brief. For a general review of other tensor operations, the reader is referred to [65]. 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

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

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 call rows and columns. 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 brief. 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 essentially vectorization of a tensor.

The other commonly used operation of tensor is modes switch, which 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 $\boldsymbol{G}'' \in \mathbb{R}^{3\times2\times2\times2}$, 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 brief, we assume their modes have been switched (in fact reordered), so that their dimensions are in a nondecreasing 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 turns $\boldsymbol{G}$ into $\boldsymbol{G}''' \in \mathbb{R}^{2\times2\times3}$, 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

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

Its corresponding multilinear function is in fact $G(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\hat{\boldsymbol{w}})$, with the underling variables $\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}$. Whenever applicable, we often use $G(\cdot,\cdot,\cdot,\hat{\boldsymbol{w}})$ 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 $\hat{\boldsymbol{Z}} \in \mathbb{R}^{3\times2}$ in modes 3 and 4, then we get a second order tensor (matrix) in $\mathbb{R}^{2\times2}$, 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 is multiplied by a d'th order tensor $(d' \le d)$ 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.

1.4.2 Approximation Algorithms

Approximation algorithms are the algorithms designed to find approximate solutions for an optimization problem. In practice, the concept of approximation algorithm is attractive for NP-hard problem, since it is unlikely that there exist polynomial-time exact algorithms for solving such problems, and therefore one is forced to settle with polynomial-time suboptimal 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.

We now define formally the approximation algorithms and approximation ratios. Throughout this brief, 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 1.4.1. 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, so much so that there is no polynomial-time approximation algorithm which approximates the optimal value within any positive factor. In those unfortunate cases, an alternative would be to resort to approximation algorithms with *relative* approximation ratios.

Definition 1.4.2. 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., [11, 57]).

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 1, 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 Sect. 1.3.3, all optimization models considered in this brief are maximization problems. Thus we reserve the greek letter τ, specialized to indicate the approximation ratio, which is a key ingredient throughout this brief. All the approximation ratios presented in this brief 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 $n \geq n_0$.

1.4.3 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.

In solving NP-hard optimization problems, randomized algorithms are often utilized to ensure performance ratios, in terms of expectation, or with high probability. The randomized version of approximation algorithms (the deterministic counterpart is to be found in Definition 1.4.1) below; similarly for Definition 1.4.2.

Definition 1.4.3. 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 $v(P) \geq 0$ 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 $\mathrm{E}[p(\hat{\boldsymbol{x}})] \geq \tau\, v(P)$.
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 $1-\varepsilon$ for all $\varepsilon \in (0,1)$.

1.4.4 *Semidefinite Programming Relaxation and Randomization*

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 subspace. It can be viewed as an extension of the well-known linear programming model, 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

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

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 Sect. 1.4.1, and "$\boldsymbol{X} \succeq 0$" means matrix $\boldsymbol{X}$ is positive semidefinite.

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 $\{\boldsymbol{v}^1, \boldsymbol{v}^2, \ldots, \boldsymbol{v}^n\}$ 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 $\{\boldsymbol{v}^1, \boldsymbol{v}^2, \ldots, \boldsymbol{v}^n\}$ 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 $\ln(1/\varepsilon)$. Interior point methods are the most popular and widely used ones. A lot of efficient SDP solvers based on interior point methods have been developed, including SeDuMi of Sturm [109], SDPT3 of Toh et al. [112], SDPA of Fujisawa et al. [36], CSDP of Borchers [19], DSDP of Benson and Ye [16], and so on.

SDP is of great importance in convex optimization 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 the design of approximate solutions to combinatorial optimization problems, starting from the seminal work by Goemans and Williamson [39], 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 [5, 23, 48, 74, 86, 87, 115–118] and even quartic polynomial optimization [72, 76]. We now elaborate the max-cut algorithm of Goemans and Williamson.

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 $\{1,2,\dots,n\}$ 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 $(i = 1,2,\dots,n)$ 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:

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

The problem is NP-hard (see, e.g., Garey and Johnson [37]). Now by introducing a matrix $\boldsymbol{X}$ with X_{ij} replacing $x_i x_j$, the constraint is then equivalent to $\operatorname{diag}(\boldsymbol{X}) = \boldsymbol{e}$, $\boldsymbol{X} \succeq 0$, $\operatorname{rank}(\boldsymbol{X}) = 1$. A straightforward SDP relaxation is dropping the rank-one constraint, which yields

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

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

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

and lets $\hat{x}_i = \text{sign}(\xi_i)$ for $i = 1,2,\dots,n$. Note that generating a zero-mean normal random vector with covariance 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 [18] that

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

which further leads to

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

This yields a 0.878-approximation ratio for the max-cut problem. The ratio significantly improves the previous best known one.

We conclude this subsection 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. [86]. The problem is

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

where $\boldsymbol{Q}_i \succeq 0$ for $i = 1,2,\dots,m$ and $\sum_{i=1}^m \boldsymbol{Q}_i \succ 0$. Remark that this is exactly the model (H_Q) when $d = 2$, which is a special case of general polynomial optimization discussed in this brief. By using the same relaxation approach where x_ix_j is replaced by X_{ij} and the rank-one constraint is dropped, we then obtain a standard SDP relaxation for (SQP)

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

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

$$\hat{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{F}\hat{\boldsymbol{x}} \ge \Omega(1/\ln m)\, v(\text{SQP}) \ge \Omega(1/\ln m)\, v(\text{QP})$$

holds with probability bigger than a constant. Therefore, running this algorithm $O(\ln(1/\varepsilon))$ times and picking the best solution shall hit the approximation bound of $\Omega(1/\ln m)$ with probability at least $1-\varepsilon$. For details, one is referred to Nemirovski et al. [86] or He et al. [48].

Chapter 2
Polynomial Optimization Over the Euclidean Ball

In this chapter, we shall present approximation methods for polynomial optimization. The focus will be placed on optimizing several classes of polynomial functions over the Euclidean ball. The models include maximizing a multilinear form over Cartesian product of the Euclidean balls, a homogeneous form over the Euclidean ball, a mixed form over Cartesian product of the Euclidean balls, and a general inhomogeneous polynomial over the Euclidean ball:

$$\boxed{\begin{array}{ll} (T_{\bar{\mathbb{S}}}) & \max\ F(\boldsymbol{x}^1,\boldsymbol{x}^2,\dots,\boldsymbol{x}^d) \\ & \text{s.t.}\quad \boldsymbol{x}^k \in \bar{\mathbb{S}}^{n_k},\ k=1,2,\dots,d \end{array}}$$

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

$$\boxed{\begin{array}{ll} (M_{\bar{\mathbb{S}}}) & \max\ f(\boldsymbol{x}^1,\boldsymbol{x}^2,\dots,\boldsymbol{x}^s) \\ & \text{s.t.}\quad \boldsymbol{x}^k \in \bar{\mathbb{S}}^{n_k},\ k=1,2,\dots,s \end{array}}$$

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

Among the above four polynomial optimization models, the degree of generality increases in the sequential order. Our focus is the design of polynomial-time approximation algorithms with guaranteed worst case performance ratios. There are two reasons for us to choose the Euclidean ball as a typical constraint set. The first is its simplicity, notwithstanding the wide applications. The second and more important reason is that, through this relatively simple case-study, we hope to clearly demonstrate how the new techniques work; much of the analysis can be adapted to other forms of constraints.

2.1 Multilinear Form

The first subclass of polynomial optimization models studied in this brief is the following multilinear form optimization over the Euclidean ball, i.e.,

$$\boxed{\begin{array}{ll}(T_{\bar{S}}) & \max\ F(\boldsymbol{x}^1,\boldsymbol{x}^2,\dots,\boldsymbol{x}^d)\\ & \text{s.t.}\quad \boldsymbol{x}^k \in \bar{\mathbb{S}}^{n_k},\ k=1,2,\dots,d\end{array}}$$

where $n_1 \le n_2 \le \cdots \le n_d$.

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

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

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. Therefore in this section, for the ease of presentation, we use $\mathbb{S}^{n_k}$ and $\bar{\mathbb{S}}^{n_k}$ interchangeably in the analysis.

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., [66,113]). Multilinear form is a special class of homogeneous polynomials. In fact, one of the main reasons for us to study multilinear form optimization is its strong connection to homogenous polynomial optimization in deriving approximation bounds, whose details will be discussed in Sect. 2.2. This connection enables a new approach to solve polynomial optimization problems, and the fundamental issue is how to optimize a multilinear form over a set. Chen et al. [24] establish the tightness result of multilinear form relaxation for maximizing a homogeneous form over the Euclidean ball. The study of multilinear form optimization has become centrally important.

The low degree cases of $(T_{\bar{S}})$ are immediately recognizable. For $d = 1$, its optimal solution is $\boldsymbol{F}/\|\boldsymbol{F}\|$ due to the Cauchy–Schwartz inequality; for $d = 2$, $(T_{\bar{S}})$ is to compute the spectrum norm of the matrix $\boldsymbol{F}$ with efficient algorithms readily available. As we shall prove later that $(T_{\bar{S}})$ is already NP-hard when $d = 3$, the focus of this section is to design polynomial-time approximation algorithms with worst-case performance ratios for any fixed degree d. Our basic approach to deal with a high degree multilinear form is to bring its order down step by step, finally leading to a multilinear form optimization in a very low order, hence solvable. Like any matrix can be treated as a long vector, any tensor can also be regarded as a reformed lower order tensor, e.g., by rewriting its corresponding multilinear form by one degree lower. (See the tensor operation in Sect. 1.4.1). After we solve the problem at a lower order, we need to decompose the solution to make it feasible for the original order. Then, specific decomposition methods are required, which will be the topic of this section.

2.1.1 Computational Complexity

To start, a special case of $(T_{\bar{S}})$ is worth noting, which plays an important role in our algorithms.

Proposition 2.1.1 *If $d = 2$, then $(T_{\bar{S}})$ can be solved in polynomial time, with $v(T_{\bar{S}}) \geq \|\boldsymbol{F}\|/\sqrt{n_1}$.*

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

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

which implies $v(T_{\bar{S}}) = \sqrt{\lambda_{\max}(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F})} \geq \|\boldsymbol{F}\|/\sqrt{n_1}$. □

However, for any degree $d \geq 3$, $(T_{\bar{S}})$ becomes NP-hard. Before engaging in a formal proof, let us first quote a complexity result for a polynomial optimization over the Euclidean sphere due to Nesterov [89].

Lemma 2.1.2 *Suppose $\boldsymbol{A}_k \in \mathbb{R}^{n\times n}$ is symmetric for $k = 1,2,\ldots,m$, and $f(\boldsymbol{x})$ is a homogeneous cubic form, then*

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

and

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

are both NP-hard.

The proof is based on the reduction to the Motzkin–Straus formulation [82] of the stability number of the graph; for details, the reader is referred to Theorem 4 of [89].

Proposition 2.1.3 *If $d = 3$, then $(T_{\bar{S}})$ is NP-hard.*

Proof. 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 $1 \leq i,j,k \leq n$, the objective function of $(T_{\bar{S}})$ 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, $(T_{\bar{\mathbb{S}}})$ is equivalent to

$$\begin{array}{ll} \max & \sum_{k=1}^n (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_k\boldsymbol{y})^2 \\ \text{s.t.} & \boldsymbol{x},\boldsymbol{y} \in \mathbb{S}^n. \end{array} \tag{2.1}$$

We shall first 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^*. 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 $\lambda = \mu = v^*$. Summing up the two equations, pre-multiplying $\hat{\boldsymbol{w}}^{\mathrm{T}}$, and then scaling, lead us to

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

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

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

which implies that $(\hat{\boldsymbol{w}},\hat{\boldsymbol{w}})$ is also an optimal solution. Problem (2.1) is then reduced to Nesterov's quartic model in Lemma 2.1.2, and its NP-hardness thus follows. □

2.1.2 Cubic Case

In the remainder of this section, we focus on approximation algorithms for $(T_{\bar{\mathbb{S}}})$ with general degree d. To illustrate the main idea of the algorithms, we first work with the case $d = 3$ in this subsection

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

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

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

Model $(\hat{T}_{\bar{S}})$ can now be relaxed to

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

Notice that the above problem is exactly $(T_{\bar{S}})$ with $d=2$, which can be solved in polynomial-time by Proposition 2.1.1. Denote its optimal solution to be $(\hat{\boldsymbol{W}},\hat{\boldsymbol{z}})$. Clearly $F(\hat{\boldsymbol{W}},\hat{\boldsymbol{z}}) \ge v(\hat{T}_{\bar{S}})$. 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 2.1.1

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

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 $\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{\eta} \ge \boldsymbol{M}\bullet \boldsymbol{W}$ *and* $\|\boldsymbol{\xi}\|\|\boldsymbol{\eta}\| \le O(\sqrt{n_1})$.

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. [48] 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 $\theta \ge 0.03$ such that

$$\Pr\{f(\boldsymbol{x}) \ge \mathrm{E}[f(\boldsymbol{x})]\} \ge \theta.$$

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

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

Moreover, by using a property of normal random vectors (see Lemma 3.1 of [76]), we have

$$\begin{aligned}\mathrm{E}\left[\|\boldsymbol{\xi}\|^2\|\boldsymbol{\eta}\|^2\right] &= \mathrm{E}\left[\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\xi_i{}^2\eta_j{}^2\right] = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\left(\mathrm{E}[\xi_i{}^2]\mathrm{E}[\eta_j{}^2]+2(\mathrm{E}[\xi_i\eta_j])^2\right)\\ &= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\left[(\hat{W}^{\mathrm{T}}\hat{W})_{jj}+2\hat{W}_{ij}{}^2\right] = (n_1+2)\,\mathrm{tr}\,(\hat{\boldsymbol{W}}^{\mathrm{T}}\hat{\boldsymbol{W}}) \le n_1+2.\end{aligned}$$

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

$$\Pr\{\|\boldsymbol{\xi}\|^2\|\boldsymbol{\eta}\|^2 \ge t\} \le \mathrm{E}\left[\|\boldsymbol{\xi}\|^2\|\boldsymbol{\eta}\|^2\right]/t \le (n_1+2)/t.$$

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

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

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

$$F(\boldsymbol{x},\boldsymbol{y},\hat{\boldsymbol{z}}) \ge \frac{F(\hat{\boldsymbol{W}},\hat{\boldsymbol{z}})}{\sqrt{t}} \ge v(\hat{T}_{\bar{S}})\sqrt{\frac{\theta}{2(n_1+2)}},$$

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

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

Decomposition Routine 2.1.2

```
• 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 2.1.1, which tells us that $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{y} \ge \|\boldsymbol{M}\|/\sqrt{n_1}$. Thus we have

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

The complexity for DR 2.1.1 is $O(n_1 n_2 \ln(1/\varepsilon))$ with probability $1-\varepsilon$, and for DR 2.1.2 it is $O\left(\max\{{n_1}^3, n_1 n_2\}\right)$. However, DR 2.1.2 is indeed very easy to implement, and is deterministic. Both DR 2.1.1 and DR 2.1.2 lead to the following approximation result in terms of the order of the approximation ratio.

Theorem 2.1.4 *If $d=3$, then $(T_{\bar{S}})$ admits a polynomial-time approximation algorithm with approximation ratio $1/\sqrt{n_1}$.*

2.1.3 General Fixed Degree

Now we are ready to proceed to the general case of fixed degree d. Let $\boldsymbol{X} = \boldsymbol{x}^1(\boldsymbol{x}^d)^{\mathrm{T}}$, and $(T_{\bar{S}})$ can be relaxed to

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

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

$$F(\hat{\boldsymbol{X}}, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \ldots, \hat{\boldsymbol{x}}^{d-1}) \ge \tau v(\tilde{T}_{\bar{S}}) \ge \tau v(T_{\bar{S}}).$$

Observing that $F(\cdot, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \ldots, \hat{\boldsymbol{x}}^{d-1}, \cdot)$ is an $n_1 \times n_d$ matrix, using DR 2.1.2 we shall find $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^d)$ such that

$$F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \ldots, \hat{\boldsymbol{x}}^d) \ge F(\hat{\boldsymbol{X}}, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \ldots, \hat{\boldsymbol{x}}^{d-1})/\sqrt{n_1} \ge {n_1}^{-\frac{1}{2}} \tau v(T_{\bar{S}}).$$

By induction this leads to the following.

Theorem 2.1.5 *$(T_{\bar{S}})$ 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 $(T_{\bar{S}})$ as in Theorem 2.1.5.

Algorithm 2.1.3

```
• INPUT: a d-th order tensor $\boldsymbol{F} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ with $n_1 \le n_2 \le \cdots \le n_d$.
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.,
  $F_{i_1,i_2,\ldots,i_d} = F'_{i_2,i_3,\ldots,i_{d-1},(i_1-1)n_d+i_d} \quad \forall\, 1 \le i_1 \le n_1, 1 \le i_2 \le n_2, \ldots, 1 \le i_d \le n_d.$
2 For $(T_{\bar{S}})$ with the $(d-1)$-th order tensor $\boldsymbol{F}'$: if $d-1=2$, then apply DR 2.1.2,
  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
  $(\hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \ldots, \hat{\boldsymbol{x}}^{d-1}, \hat{\boldsymbol{x}}^{1,d})$ by recursion.
3 Compute a matrix $\boldsymbol{M}' = F(\cdot, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \ldots, \hat{\boldsymbol{x}}^{d-1}, \cdot)$ and rewrite the vector $\hat{\boldsymbol{x}}^{1,d}$ as a
  matrix $\boldsymbol{X} \in \bar{\mathbb{S}}^{n_1 \times n_d}$.
4 Apply either DR 2.1.1 or DR 2.1.2, 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 $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \ldots, \hat{\boldsymbol{x}}^d)$.
```

2.2 Homogeneous Form

This section focuses on optimization of homogeneous polynomials (or forms) over the Euclidean ball:

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

When the degree of the polynomial objective, d, is odd, $(H_{\bar{S}})$ is equivalent to

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

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 $\mathbb{S}^n$. However, if d is even, then this equivalence may not hold. For example, the optimal value of (H_S) may be negative, if the tensor $\boldsymbol{F}$ is negative definite, i.e., $f(\boldsymbol{x}) < 0$ for all $\boldsymbol{x} \neq \boldsymbol{0}$, while the optimal value of $(H_{\bar{S}})$ is always nonnegative, since $\boldsymbol{0}$ is always a feasible solution.

The model $(H_{\bar{S}})$ is in general NP-hard. In fact, when $d=1$, $(H_{\bar{S}})$ has a close-form solution, due to the Cauchy–Schwartz inequality; when $d=2$, $(H_{\bar{S}})$ is related to the largest eigenvalue of the symmetric matrix $\boldsymbol{F}$; when $n \geq 3$, $(H_{\bar{S}})$ becomes NP-hard, which was proven by Nesterov [89] (see Lemma 2.1.2). Interestingly, when $d \geq 3$, the model $(H_{\bar{S}})$ is also regarded as computing the largest eigenvalue of the supersymmetric tensor $\boldsymbol{F}$, like the case $d = 2$ (see, e.g., Qi [99]). Luo and Zhang [76] 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.

Here in this section, we are going to present polynomial-time approximation algorithms with guaranteed worse-case performance ratios for the models concerned. Our algorithms are designed to solve polynomial optimization with any given degree d, and the approximation ratios improve the previous works specialized to their particular degrees. The major novelty in our approach here is the multilinear tensor relaxation, instead of quadratic SDP relaxation methods in [72, 76]. The relaxed multilinear form optimization problems admit polynomial-time approximation algorithms discussed in Sect. 2.1. After we solve the relaxed problem approximately, the solutions for the tensor model will then be used to produce a feasible solution for the original polynomial optimization model. The remaining task of the section is to illustrate how this can be done.

2.2.1 Link Between Multilinear Form and Homogeneous Form

Let $\boldsymbol{F}$ be the supersymmetric tensor satisfying $F(\underbrace{\boldsymbol{x},\boldsymbol{x},\ldots,\boldsymbol{x}}_{d}) = f(\boldsymbol{x})$. Then $(H_{\bar{S}})$ can be *relaxed* to multilinear form optimization model $(T_{\bar{S}})$ discussed in Sect. 2.1, as follows:

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

Theorem 2.1.5 asserts that $(\hat{H}_{\bar{S}})$ 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 $(H_{\bar{S}})$ from the approximate solution of $(\hat{H}_{\bar{S}})$. For this purpose, we establish the following link between $(H_{\bar{S}})$ and $(\hat{H}_{\bar{S}})$.

Lemma 2.2.1 *Suppose* $\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^d \in \mathbb{R}^n$, *and* $\xi_1,\xi_2,\ldots,\xi_d$ *are i.i.d. random variables, each taking values* 1 *and* -1 *with equal probability* $1/2$. *For any supersymmetric d-th order tensor* $\boldsymbol{F}$ *and function* $f(\boldsymbol{x}) = F(\boldsymbol{x},\boldsymbol{x},\ldots,\boldsymbol{x})$, *it holds that*

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

Proof. First we observe that

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

If $\{k_1,k_2,\ldots,k_d\}$ 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 $k_0\ne k_j$ for all $1\le j\le d$. In the latter case,

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

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

Note that the coefficients of the link identity in Lemma 2.2.1, $\prod_{i=1}^{d}\xi_i$, are not always positive. Therefore, whether the degree of the polynomial objective d is even or odd makes a difference.

2.2.2 The Odd Degree Case

When d is odd, the identity in Lemma 2.2.1 can be rewritten as

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

Since $\xi_1,\xi_2,\ldots,\xi_d$ 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

$$f\left(\sum_{k=1}^{d}\left(\prod_{i\ne k}\beta_i\right)\boldsymbol{x}^k\right) \ge d!F(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^d). \tag{2.2}$$

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

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

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

Combining with Theorem 2.1.5, we have

Theorem 2.2.2 *When $d \ge 3$ is odd, $(H_{\bar{S}})$ 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 $(H_{\bar{S}})$ with odd d is highlighted below.

Algorithm 2.2.1

```
• INPUT: a d-th order supersymmetric tensor $\boldsymbol{F} \in \mathbb{R}^{n^d}$
1 Apply Algorithm 2.1.3 to solve the problem
      $\max\ F(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^d)$
      $\text{s.t.}\ \ \boldsymbol{x}^k \in \bar{\mathbb{S}}^n,\ k=1,2,\ldots,d$
  approximately, with input $\boldsymbol{F}$ and output $(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\ldots,\hat{\boldsymbol{x}}^d)$.
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 $f\left(\sum_{k=1}^d \beta_k\hat{\boldsymbol{x}}^k\right) \ge d!F(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\ldots,\hat{\boldsymbol{x}}^d)$.
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 2.2.1, as (2.2) 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 2.2.1 is deterministic and runs in polynomial time for fixed d.

2.2.3 *The Even Degree Case*

When d is even, the only easy case of $(H_{\bar{S}})$ appears to be $d = 2$, and even worse, we have the following.

Proposition 2.2.3 *If $d = 4$, then there is no polynomial-time approximation algorithm with a positive approximation ratio for $(H_{\bar{\mathbb{S}}})$ unless $P = NP$.*

Proof. Let $f(\boldsymbol{x}) = F(\boldsymbol{x},\boldsymbol{x},\boldsymbol{x},\boldsymbol{x})$ with $\boldsymbol{F}$ being supersymmetric. 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 $v^* = \max_{\boldsymbol{x}\in\bar{\mathbb{S}}^n} -F(\boldsymbol{x},\boldsymbol{x},\boldsymbol{x},\boldsymbol{x})$, 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 $(H_{\bar{\mathbb{S}}})$ when $d \geq 4$ is even. Thus we can only speak of *relative* approximation ratio. The following algorithm slightly modifies Algorithm 2.2.1, and works for $(H_{\bar{\mathbb{S}}})$ when d is even.

Algorithm 2.2.2

```
• INPUT: a d-th order supersymmetric tensor $\boldsymbol{F} \in \mathbb{R}^{n^d}$
1 Apply Algorithm 2.1.3 to solve the problem
      $\max\ F(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^d)$
      $\text{s.t.}\ \ \boldsymbol{x}^k \in \bar{\mathbb{S}}^n,\ k = 1,2,\ldots,d$
  approximately, with input $\boldsymbol{F}$ and output $(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\ldots,\hat{\boldsymbol{x}}^d)$.
2 Compute $\boldsymbol{\beta} = \arg\max_{\boldsymbol{\xi}\in\mathbb{B}^d,\prod_{i=1}^d \xi_i=1} \left\{ f\left(\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right) \right\}$.
3 Compute $\hat{\boldsymbol{x}} = \sum_{k=1}^d \beta_k \hat{\boldsymbol{x}}^k / d$.
• OUTPUT: a feasible solution $\hat{\boldsymbol{x}} \in \bar{\mathbb{S}}^n$.
```

Theorem 2.2.4 *When $d \geq 4$ is even, $(H_{\bar{\mathbb{S}}})$ admits a polynomial-time approximation algorithm with relative approximation ratio $\tau(H_S)$.*

Proof. Like in the proof of Theorem 2.2.2, by relaxing $(H_{\bar{\mathbb{S}}})$ to $(\hat{H}_{\bar{\mathbb{S}}})$, we are able to find a set of vectors $(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\ldots,\hat{\boldsymbol{x}}^d)$ in the Euclidean ball, such that

$$F(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\ldots,\hat{\boldsymbol{x}}^d) \geq n^{-\frac{d-2}{2}} v(\hat{H}_{\bar{\mathbb{S}}}).$$

Besides, we observe that $v(H_{\bar{S}}) \le v(\hat{H}_{\bar{S}})$ and $\underline{v}(H_{\bar{S}}) \ge \underline{v}(\hat{H}_{\bar{S}}) = -v(\hat{H}_{\bar{S}})$. Therefore

$$2\,v(\hat{H}_{\bar{S}}) \ge v(H_{\bar{S}}) - \underline{v}(H_{\bar{S}}). \tag{2.3}$$

Let $\xi_1, \xi_2, \dots, \xi_d$ be i.i.d. random variables, each taking values 1 and -1 with equal probability $1/2$. Obviously, $\Pr\left\{\prod_{i=1}^d \xi_i = 1\right\} = \Pr\left\{\prod_{i=1}^d \xi_i = -1\right\} = 1/2$. By the triangle inequality, it follows that $\frac{1}{d}\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k \in \bar{\mathbb{S}}^n$, and so $f(\frac{1}{d}\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k) \ge \underline{v}(H_{\bar{S}})$. Applying Lemma 2.2.1 and we have

$$\begin{aligned}
&\frac{1}{2}\,\mathrm{E}\left[f\left(\frac{1}{d}\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right) - \underline{v}(H_{\bar{S}}) \,\middle|\, \prod_{i=1}^d \xi_i = 1 \right] \\
&\ge \mathrm{E}\left[f\left(\frac{1}{d}\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right) - \underline{v}(H_{\bar{S}}) \,\middle|\, \prod_{i=1}^d \xi_i = 1 \right] \Pr\left\{\prod_{i=1}^d \xi_i = 1\right\} \\
&\quad - \mathrm{E}\left[f\left(\frac{1}{d}\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right) - \underline{v}(H_{\bar{S}}) \,\middle|\, \prod_{i=1}^d \xi_i = -1 \right] \Pr\left\{\prod_{i=1}^d \xi_i = -1\right\} \\
&= \mathrm{E}\left[\prod_{i=1}^d \xi_i \left(f\left(\frac{1}{d}\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right) - \underline{v}(H_{\bar{S}}) \right) \right] \\
&= d^{-d}\,\mathrm{E}\left[\prod_{i=1}^d \xi_i f\left(\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k\right) \right] - \underline{v}(H_{\bar{S}})\,\mathrm{E}\left[\prod_{i=1}^d \xi_i \right] \\
&= d^{-d} d!\, F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \dots, \hat{\boldsymbol{x}}^d) \ge \tau(H_S)\, v(\hat{H}_{\bar{S}}) \ge (\tau(H_S)/2)\left(v(H_{\bar{S}}) - \underline{v}(H_{\bar{S}})\right).
\end{aligned}$$

Thus we may find a binary vector $\boldsymbol{\beta} \in \mathbb{B}^d$ with $\prod_{i=1}^d \beta_i = 1$, such that

$$f\left(\frac{1}{d}\sum_{k=1}^d \beta_k \hat{\boldsymbol{x}}^k\right) - \underline{v}(H_{\bar{S}}) \ge \tau(H_S)\left(v(H_{\bar{S}}) - \underline{v}(H_{\bar{S}})\right).$$

□

2.3 Mixed Form

In this section, we extend the study on the multilinear form and the homogeneous form to a general mixed form, i.e.,

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

where $d = d_1 + d_2 + \cdots + d_s$ is deemed 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. Here we assume that $n_1 \le n_2 \le \cdots \le n_s$. The mixed-form optimization model considered here is

$$\boxed{\begin{array}{ll} (M_{\bar{S}}) & \max\ f(\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^s) \\ & \text{s.t.}\quad \boldsymbol{x}^k \in \bar{\mathbb{S}}^{n_k},\ k = 1, 2, \ldots, s \end{array}}$$

The model $(M_{\bar{S}})$ is a generalization of $(T_{\bar{S}})$ in Sect. 2.1 and $(H_{\bar{S}})$ in Sect. 2.2. For the computational complexity, it is similar to its special cases $(T_{\bar{S}})$ and $(H_{\bar{S}})$. It is solvable in polynomial time when $d \le 2$, and is NP-hard when $d \ge 3$, which will be shown shortly later. Moreover, when $d \ge 4$ and all $d_i\,(1 \le k \le s)$ are even, there is no polynomial-time approximation algorithm with a positive approximation ratio unless $P = NP$. This can be verified in its simplest case of $d = 4$ and $d_1 = d_2 = 2$ by using a similar argument as in Ling et al. [72]. In fact, the biquadratic optimization model considered in Ling et al. [72] is slightly different from $(M_{\bar{S}})$, and is exactly the model (M_S) when $d = 4$ and $d_1 = d_2 = 2$, i.e., the Euclidean sphere is considered instead of the Euclidean ball. 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 $\Omega\left(1/{n_2}^2\right)$.

Like we did earlier, below we are going to present polynomial-time approximation algorithms with guaranteed worse-case performance ratios. Our algorithms work for any fixed degree d, and the approximation ratios improve that of Ling et al. [72] specialized to the quartic case. Instead of using the quadratic SDP relaxation methods in [72], we resort to the multilinear form relaxation, similar as for $(H_{\bar{S}})$. However, one has to adjust Lemma 2.2.1 carefully, and a more general link from the multilinear form to the mixed form need be established, which is the objective of this section.

2.3.1 Complexity and a Step-by-Step Adjustment

First, let us settle the following hardness issue.

Proposition 2.3.1 *If $d = 3$, then $(M_{\bar{S}})$ is NP-hard.*

Proof. We need to verify the NP-hardness for three cases under $d = 3$: $(d_1, d_2, d_3) = (3,0,0)$, $(d_1, d_2, d_3) = (2,1,0)$ and $(d_1, d_2, d_3) = (1,1,1)$. The first case is exactly $(H_{\bar{S}})$ with $d = 3$, whose NP-hardness was claimed in Lemma 2.1.2, and the last case is exactly $(T_{\bar{S}})$ with $d = 3$, whose NP-hardness was shown in Proposition 2.1.3.

It remains to consider the second case $(d_1, d_2, d_3) = (2,1,0)$. As a special case, we focus on $n_1 = n_2 = n$ and $\boldsymbol{F} \in \mathbb{R}^{n^3}$ satisfying $F_{ijk} = F_{jik}$ for all $1 \le i, j, k \le n$. We notice that the following form of $(T_{\bar{S}})$ is NP-hard (cf. the proof of Proposition 2.1.3):

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

We shall show that the optimal value of $(\check{T}_{\bar{S}})$ is equal to the optimal value of this special case

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

It is obvious that $v(\check{T}_{\bar{S}})\ge v(\check{M}_{\bar{S}})$. Now choose any optimal solution $(\boldsymbol{x}^*,\boldsymbol{y}^*,\boldsymbol{z}^*)$ of $(\check{T}_{\bar{S}})$ 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

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

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 $(\check{T}_{\bar{S}})$. Therefore $v(\check{T}_{\bar{S}})\le v(\check{M}_{\bar{S}})$, and this proves $v(\check{T}_{\bar{S}})=v(\check{M}_{\bar{S}})$. If $(\check{M}_{\bar{S}})$ can be solved in polynomial time, then its optimal solution is also an optimal solution for $(\check{T}_{\bar{S}})$, which would solve $(\check{T}_{\bar{S}})$ in polynomial time, a contradiction to its NP-hardness. □

Thus we shall focus on polynomial-time approximation algorithms. Similar to the relaxation in Sect. 2.2, we relax $(M_{\bar{S}})$ to the multilinear form optimization $(T_{\bar{S}})$ as follows:

$$\begin{array}{ll}\max & F(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^d) \\ \text{s.t.} & \boldsymbol{x}^k\in\bar{\mathbb{S}}^{n_1},\ 1\le k\le d_1, \\ & \boldsymbol{x}^k\in\bar{\mathbb{S}}^{n_2},\ d_1+1\le k\le d_1+d_2, \\ & \quad\vdots \\ & \boldsymbol{x}^k\in\bar{\mathbb{S}}^{n_s},\ d_1+d_2+\cdots+d_{s-1}+1\le k\le d,\end{array}$$

then by Theorem 2.1.5 we are able to find $(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\ldots,\hat{\boldsymbol{x}}^d)$ with $\|\boldsymbol{x}^k\|\le 1$ for all $1\le k\le d$ in polynomial time, such that

$$F(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\ldots,\hat{\boldsymbol{x}}^d)\ge\tilde{\tau}(M_S)v(M_{\bar{S}}), \tag{2.4}$$

where

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

In order to draw a feasible solution for $(M_{\bar{S}})$ from $(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\ldots,\hat{\boldsymbol{x}}^d)$, we need to apply the identity stipulated in Lemma 2.2.1 in a careful manner. Approximation results for $(H_{\bar{S}})$ can be similarly derived for the odd case.

Theorem 2.3.2 *If $d \geq 3$ and one of $d_k\,(k = 1,2,\ldots,s)$ is odd, then $(M_{\bar{S}})$ admits a polynomial-time approximation algorithm with approximation ratio $\hat{\tau}(M_S)$, where*

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

To prevent the notations from getting out of hand, here we shall only consider a special case $(\hat{M}_{\bar{S}})$, which is easily extended to general $(M_{\bar{S}})$:

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

By (2.4), we are able to find $\boldsymbol{x}^1,\boldsymbol{x}^2,\boldsymbol{x}^3,\boldsymbol{x}^4 \in \bar{\mathbb{S}}^{n_1}, \boldsymbol{y}^1,\boldsymbol{y}^2 \in \bar{\mathbb{S}}^{n_2}$, and $\boldsymbol{z}^1,\boldsymbol{z}^2,\boldsymbol{z}^3 \in \bar{\mathbb{S}}^{n_3}$ in polynomial time, such that

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

Let us first fix $(\boldsymbol{y}^1,\boldsymbol{y}^2,\boldsymbol{z}^1,\boldsymbol{z}^2,\boldsymbol{z}^3)$ and try to get a solution for the problem

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

Using the same argument as in the proof of Theorem 2.2.2, we are able to find $\hat{\boldsymbol{x}} \in \bar{\mathbb{S}}^{n_1}$, 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 we use $-\boldsymbol{z}^1$ to update $\boldsymbol{z}^1$. In this context the even degree ($d_1 = 4$) of $\boldsymbol{x}$ does not raise any issue, as we can always move the negative sign to $\boldsymbol{z}^1$. This process may be considered variable adjustment, and the approximation bound is $4!4^{-4}\tilde{\tau}(M_S)$.

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

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

The problem is equivalent to finding the largest absolute eigenvalue of a matrix, which can be solved in polynomial time. Denote its optimal solution to be $\hat{\boldsymbol{y}}$, and update $\boldsymbol{z}^1$ with $-\boldsymbol{z}^1$ if necessary. This process leads to an 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 on $\boldsymbol{x}$, now we work with

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

and we can find $\hat{\boldsymbol{z}} \in \bar{\mathbb{S}}^{n_3}$ 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 need one of d_k's to be odd, which allows us to ensure that the last variable for adjustment has an odd degree.

2.3.2 Extended Link Between Multilinear Form and Mixed Form

If all d_k's $(k = 1,2,\dots,s)$ are even, then we can only hope for a *relative* approximation ratio. For the simplest case where $d = 4$ and $d_1 = d_2 = 2$, the biquadratic optimization model $\max_{\boldsymbol{x}\in\bar{\mathbb{S}}^{n_1},\boldsymbol{y}\in\bar{\mathbb{S}}^{n_2}} F(\boldsymbol{x},\boldsymbol{x},\boldsymbol{y},\boldsymbol{y})$ does not admit any polynomial-time approximation algorithm with a positive approximation ratio. Before working out this case, let us first introduce the following link between the multilinear form and the mixed form, extended from Lemma 2.2.1.

Lemma 2.3.3 *Suppose that* $\boldsymbol{x}^k \in \mathbb{R}^{n_1}\,(1 \le k \le d_1)$, $\boldsymbol{x}^k \in \mathbb{R}^{n_2}\,(d_1+1 \le k \le d_1+d_2)$, $\dots$, $\boldsymbol{x}^k \in \mathbb{R}^{n_s}\,(d_1+d_2+\cdots+d_{s-1}+1 \le k \le d_1+d_2+\cdots+d_s = d)$, *and* $\xi_1,\xi_2,\dots,\xi_d$ *are i.i.d. random variables, each taking values* 1 *and* -1 *with equal probability* $1/2$. *Denote*

$$\boldsymbol{x}^1_\xi = \sum_{k=1}^{d_1} \xi_k \boldsymbol{x}^k,\ \boldsymbol{x}^2_\xi = \sum_{k=d_1+1}^{d_1+d_2} \xi_k \boldsymbol{x}^k,\ \dots,\ \boldsymbol{x}^s_\xi = \sum_{k=d_1+d_2+\cdots+d_{s-1}+1}^{d} \xi_k \boldsymbol{x}^k. \tag{2.5}$$

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*

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

it holds that

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

This lemma is easy to prove by invoking Lemma 2.2.1 repeatedly s times. Now, with this extended link in hand, we can then apply a similar argument as in the proof of Theorem 2.2.4.

Theorem 2.3.4 *If $d \geq 4$ and all d_k $(k = 1, 2, \ldots, s)$ are even, then $(M_{\bar{S}})$ admits a polynomial-time approximation algorithm with relative approximation ratio $\tau(M_S)$, where*

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

Remark that the case $d_s = 1$ is theoretically not relevant for Theorem 2.3.4 since it assumes all d_k to be even. However, we shall keep this definition of $\tau(M_S)$ for the interest of Sect. 3.2 where this definition will be used.

2.4 Inhomogeneous Polynomial

The last section of this chapter tackles an important and useful extension of the models studied in the previous sections: a generic inhomogeneous polynomial objective function. As is evident, many important applications of polynomial optimization involve an objective that is intrinsically inhomogeneous. Specifically, we consider the following model:

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

The above model 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 show later. Therefore, the whole section is focused on *relative* approximation algorithms. The inapproximability of $(P_{\bar{S}})$ differs greatly from that of the homogeneous model $(H_{\bar{S}})$ discussed in Sect. 2.2, since when d is odd, $(H_{\bar{S}})$ admits a polynomial-time approximation algorithm with a positive approximation ratio by Theorem 2.2.2. Consequently, the optimization of an inhomogeneous polynomial is much harder than a homogeneous one.

Extending the solution methods and the corresponding analysis from *homogeneous* polynomial optimization 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 [48, 74, 86, 87, 117] and the quartic models considered in [72, 76], 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 section 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 section 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.

Let us now focus on the approximation methods for $(P_{\bar{\mathbb{S}}})$. As this section is concerned with the relative approximation ratios, we may without loss of generality assume $p(\boldsymbol{x})$ to have no constant term, i.e., $p(\boldsymbol{0}) = 0$. Thus the optimal value of this problem is obviously nonnegative, i.e., $v(P_{\bar{\mathbb{S}}}) \geq 0$. The complexity to solve $(P_{\bar{\mathbb{S}}})$ is summarized in the following proposition.

Proposition 2.4.1 *If $d \leq 2$, then $(P_{\bar{\mathbb{S}}})$ can be solved in polynomial time. If $d \geq 3$, then $(P_{\bar{\mathbb{S}}})$ is NP-hard, and there is no polynomial-time approximation algorithm with a positive approximation ratio unless $P = NP$.*

Proof. For $d \leq 2$, $(P_{\bar{\mathbb{S}}})$ is a standard trust region subproblem. As such it is well known to be solvable in polynomial time (see, e.g., [110, 111] and the references therein). For $d \geq 3$, in a special case where $p(\boldsymbol{x})$ is a homogeneous cubic form, $(P_{\bar{\mathbb{S}}})$ is equivalent to $\max_{\boldsymbol{x}\in\mathbb{S}^n} p(\boldsymbol{x})$, which is shown to be NP-hard by Nesterov [89]; see also Lemma 2.1.2.

Let us now consider a special class of $(P_{\bar{\mathbb{S}}})$ when $d = 3$:

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

where $\alpha \geq 0$, and $f(\boldsymbol{x})$ is a homogeneous cubic form associated with a nonzero supersymmetric tensor $\boldsymbol{F} \in \mathbb{R}^{n\times n\times n}$. 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 $\alpha = \|\boldsymbol{F}\| \geq \max_{\boldsymbol{x}\in\bar{\mathbb{S}}^n} f(\boldsymbol{x})$, then $v(\alpha) = 0$. Since otherwise we must have $v(\alpha) > 0$ and $\|\boldsymbol{x}^*\| = 1$, with

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

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 nonincreasing 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 $(P_{\bar{S}})$ when $d \geq 3$. Then for every $\alpha \geq 0$, we can find $\boldsymbol{z} \in \bar{\mathbb{S}}^n$ 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 value 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 $P = NP$. $\square$

The negative result in Proposition 2.4.1 rules out any polynomial-time approximation algorithm with a positive approximation ratio for $(P_{\bar{S}})$ when $d \geq 3$. 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 $(P_{\bar{S}})$, which admits a (relative) worst-case performance ratio. In fact, here we present a general scheme aiming at solving the polynomial optimization $(P_{\bar{S}})$. 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 $(P_{\bar{S}})$, 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 2.4.1

```
• INPUT: an n-dimensional d-th degree polynomial function $p(\boldsymbol{x})$.
1 Rewrite $p(\boldsymbol{x}) - p(\boldsymbol{0}) = F(\underbrace{\bar{\boldsymbol{x}},\bar{\boldsymbol{x}},\dots,\bar{\boldsymbol{x}}}_{d})$ when $x_h = 1$ as in (2.7), with $\boldsymbol{F}$ being an
  $(n+1)$-dimensional d-th order supersymmetric tensor.
2 Apply Algorithm 2.1.3 to solve the problem
      $\max\ F(\bar{\boldsymbol{x}}^1,\bar{\boldsymbol{x}}^2,\dots,\bar{\boldsymbol{x}}^d)$
      $\text{s.t.}\ \ \bar{\boldsymbol{x}}^k \in \bar{\mathbb{S}}^{n+1},\ k = 1,2,\dots,d$
  approximately, with input $\boldsymbol{F}$ and output $(\bar{\boldsymbol{y}}^1,\bar{\boldsymbol{y}}^2,\dots,\bar{\boldsymbol{y}}^d)$.
3 Compute $(\bar{\boldsymbol{z}}^1,\bar{\boldsymbol{z}}^2,\dots,\bar{\boldsymbol{z}}^d) = \arg\max\left\{F\left(\binom{\xi_1\boldsymbol{y}^1/d}{1},\binom{\xi_2\boldsymbol{y}^2/d}{1},\dots,\binom{\xi_d\boldsymbol{y}^d/d}{1}\right),\boldsymbol{\xi}\in\mathbb{B}^d\right\}$.
4 Compute $\boldsymbol{z} = \arg\max\left\{p(\boldsymbol{0});p\left(\boldsymbol{z}(\beta)/z_h(\beta)\right),\boldsymbol{\beta}\in\mathbb{B}^d \text{ and } \beta_1 = \prod_{k=2}^d \beta_k = 1\right\}$,
  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 $\boldsymbol{z} \in \bar{\mathbb{S}}^n$.
```

In Step 2 of Algorithm 2.4.1, Algorithm 2.1.3 is called to approximately solve multilinear form optimization over the Euclidean ball, which is a deterministic polynomial-time algorithm. Notice the degree of the polynomial $p(\boldsymbol{x})$ is deemed a fixed parameter in this brief, and thus Algorithm 2.4.1 runs in polynomial time, and is deterministic too. Our main result in this section is the following.

Theorem 2.4.2 *$(P_{\bar{S}})$ admits a polynomial-time approximation algorithm with relative approximation ratio $\tau(P_S)$, where*

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

Below we study in detail how a particular implementation of these four steps of the scheme (which becomes Algorithm 2.4.1) leads to the promised worst-case relative performance ratio in Theorem 2.4.2.

2.4.1 Homogenization

The method of homogenization depends on the form of the polynomial $p(\boldsymbol{x})$. Without losing generality henceforth we assume $p(\boldsymbol{x})$ to have no constant term, although Algorithm 2.4.1 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

$$f_k(\boldsymbol{x}) = F_k(\underbrace{\boldsymbol{x},\boldsymbol{x},\ldots,\boldsymbol{x}}_{k}) \tag{2.6}$$

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

$$\begin{aligned} p(\boldsymbol{x}) &= \sum_{k=1}^{d} f_k(\boldsymbol{x}) = \sum_{k=1}^{d} f_k(\boldsymbol{x}) {x_h}^{d-k} = \sum_{k=1}^{d} F_k(\underbrace{\boldsymbol{x},\boldsymbol{x},\ldots,\boldsymbol{x}}_{k}){x_h}^{d-k} \\ &= F\Bigg(\underbrace{\begin{pmatrix}\boldsymbol{x}\\ x_h\end{pmatrix},\begin{pmatrix}\boldsymbol{x}\\ x_h\end{pmatrix},\ldots,\begin{pmatrix}\boldsymbol{x}\\ x_h\end{pmatrix}}_{d}\Bigg) = F(\underbrace{\bar{\boldsymbol{x}},\bar{\boldsymbol{x}},\ldots,\bar{\boldsymbol{x}}}_{d}) = f(\bar{\boldsymbol{x}}), \end{aligned} \tag{2.7}$$

where $\boldsymbol{F}$ is an $(n+1)$-dimensional d-th order supersymmetric 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, then 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 proceed according to (2.6) and (2.7) to obtain the tensor form $\boldsymbol{F}$, as required.

Finally, we may equivalently reformulate $(P_{\bar{S}})$ as

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

Obviously, we have $v(P_{\bar{S}}) = v(\bar{P}_{\bar{S}})$ and $\underline{v}(P_{\bar{S}}) = \underline{v}(\bar{P}_{\bar{S}})$.

2.4.2 Multilinear Form Relaxation

Multilinear form relaxation has proven to be effective, as discussed in Sects. 2.2 and 2.3. Specifically, Lemmas 2.2.1 and 2.3.3 are the key link formulae. Now we relax $(\bar{P}_{\bar{S}})$ to an inhomogeneous multilinear form optimization model

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

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

Proposition 2.4.3 *$(TP_{\bar{S}})$ is NP-hard whenever $d \geq 3$.*

Proof. Notice that in Proposition 2.1.3, we proved the following problem is NP-hard:

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

For $d = 3$ and a special case where $\boldsymbol{F}$ satisfies $F_{n+1,j,k} = F_{i,n+1,k} = F_{i,j,n+1} = 0$ for all $1 \leq i,j,k \leq n+1$, $(TP_{\bar{S}})$ is equivalent to the above model, and so it is NP-hard in general. □

$(TP_{\bar{S}})$ 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:

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

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

Both the objective and the constraints are now homogeneous, and it is obvious that for all $t > 0$, $(TP_{\bar{S}}(t))$ is equivalent (in fact scalable) to each other. Moreover, $(TP_{\bar{S}}(1))$ is

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

which is exactly $(T_{\bar{S}})$ as we discussed in Sect. 2.1. By using Algorithm 2.1.3 and applying Theorem 2.1.5, $(TP_{\bar{S}}(1))$ admits a polynomial-time approximation algorithm with approximation ratio $(n+1)^{-\frac{d-2}{2}}$. Therefore, for all $t > 0$, $(TP_{\bar{S}}(t))$ also admits a polynomial-time approximation algorithm with approximation ratio $(n+1)^{-\frac{d-2}{2}}$, and $v(TP_{\bar{S}}(t)) = t^d\, v(TP_{\bar{S}}(1))$. After this relaxation step (Step 2 in Algorithm 2.4.1), we are able to find a feasible solution $(\bar{\boldsymbol{y}}^1,\bar{\boldsymbol{y}}^2,\dots,\bar{\boldsymbol{y}}^d)$ of $(TP_{\bar{S}}(1))$ in polynomial time, such that

$$\begin{aligned} F(\bar{\boldsymbol{y}}^1,\bar{\boldsymbol{y}}^2,\dots,\bar{\boldsymbol{y}}^d) &\geq (n+1)^{-\frac{d-2}{2}}\, v(TP_{\bar{S}}(1)) \\ &= 2^{-\frac{d}{2}}(n+1)^{-\frac{d-2}{2}}\, v(TP_{\bar{S}}(\sqrt{2})) \\ &\geq 2^{-\frac{d}{2}}(n+1)^{-\frac{d-2}{2}}\, v(TP_{\bar{S}}). \end{aligned} \tag{2.8}$$

Algorithm 2.1.3 is the engine which enables the second step of our scheme. In fact, any polynomial-time approximation algorithm of $(TP_{\bar{S}}(1))$ 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 $(TP_{\bar{S}}(1))$ leads to the improvement of relative approximation ratio in Theorem 2.4.2. For example, recently So [106] improved the approximation bound of $(TP_{\bar{S}}(1))$ to $\Omega\left(\left(\frac{\ln n}{n}\right)^{-\frac{d-2}{2}}\right)$ (though the algorithm is mainly of theoretical interest), and consequently the relative approximation ratio under our scheme is improved to $\Omega\left(\left(\frac{\ln n}{n}\right)^{\frac{d-2}{2}}\right)$ too. Of course, one may apply any other favorite algorithm to solve the relaxation $(TP_{\bar{S}}(1))$. For instance, the alternating least square (ALS) algorithm (see, e.g., [65] and the references therein) and the maximum block improvement (MBI) method of Chen et al. [24] can be the other alternatives for the second step.

2.4.3 Adjusting the Homogenizing Components

The approximate solution $(\bar{\boldsymbol{y}}^1,\bar{\boldsymbol{y}}^2,\ldots,\bar{\boldsymbol{y}}^d)$ of $(TP_{\bar{S}}(1))$ satisfies $\|\bar{\boldsymbol{y}}^k\|\le 1$ for all $1\le k\le d$, which implies $\|\boldsymbol{y}^k\|\le 1$. Other from that, we do not have any control on the size of y^k_h, and thus $(\bar{\boldsymbol{y}}^1,\bar{\boldsymbol{y}}^2,\ldots,\bar{\boldsymbol{y}}^d)$ may not be a feasible solution for $(TP_{\bar{S}})$. The following lemma plays a link role in our analysis to ensure that the construction of a feasible solution for the inhomogeneous model $(TP_{\bar{S}})$ is possible.

Lemma 2.4.4 *Suppose $\bar{\boldsymbol{x}}^k\in\mathbb{R}^{n+1}$ with $|x^k_h|\le 1$ for all $1\le k\le d$. Let $\eta_1,\eta_2,\ldots,\eta_d$ be independent random variables, each taking values 1 and -1 with $\mathrm{E}[\eta_k]=x^k_h$ for all $1\le k\le d$, and let $\xi_1,\xi_2,\ldots,\xi_d$ 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*

$$\mathrm{E}\left[\prod_{k=1}^d \eta_k F\left(\begin{pmatrix}\eta_1\boldsymbol{x}^1\\1\end{pmatrix},\begin{pmatrix}\eta_2\boldsymbol{x}^2\\1\end{pmatrix},\ldots,\begin{pmatrix}\eta_d\boldsymbol{x}^d\\1\end{pmatrix}\right)\right]=F(\bar{\boldsymbol{x}}^1,\bar{\boldsymbol{x}}^2,\ldots,\bar{\boldsymbol{x}}^d), \tag{2.9}$$

and

$$\mathrm{E}\left[F\left(\begin{pmatrix}\xi_1\boldsymbol{x}^1\\1\end{pmatrix},\begin{pmatrix}\xi_2\boldsymbol{x}^2\\1\end{pmatrix},\ldots,\begin{pmatrix}\xi_d\boldsymbol{x}^d\\1\end{pmatrix}\right)\right]=0. \tag{2.10}$$

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

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

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

and

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

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

Lemma 2.4.4 suggests that one may enumerate the 2^d possible combinations of $\left(\binom{\xi_1\boldsymbol{y}^1}{1}, \binom{\xi_2\boldsymbol{y}^2}{1}, \ldots, \binom{\xi_d\boldsymbol{y}^d}{1}\right)$ 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 $(TP_{\bar{S}})$ from a feasible solution for the homogeneous multilinear form optimization $(TP_{\bar{S}}(1))$, with a controlled possible quality deterioration. This fact plays a key role in proving the approximation ratio for $(TP_{\bar{S}})$.

Theorem 2.4.5 *$(TP_{\bar{S}})$ admits a polynomial-time approximation algorithm with approximation ratio $2^{-\frac{3d}{2}}(n+1)^{-\frac{d-2}{2}}$.*

Proof. Let $(\bar{\boldsymbol{y}}^1, \bar{\boldsymbol{y}}^2, \ldots, \bar{\boldsymbol{y}}^d)$ be the feasible solution found in Step 2 of Algorithm 2.4.1 satisfying (2.8), and let $\boldsymbol{\eta} = (\eta_1, \eta_2, \ldots, \eta_d)^{\mathrm{T}}$ with all η_k's being independent and taking values 1 and -1 such that $\mathrm{E}[\eta_k] = y_h^k$. Applying Lemma 2.4.4, we have (2.9) which implies

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

and (2.10) which implies

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

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

$$\begin{aligned} &F(\bar{\boldsymbol{y}}^1,\bar{\boldsymbol{y}}^2,\dots,\bar{\boldsymbol{y}}^d)\\ &\quad=\sum_{\boldsymbol{\beta}\in\mathbb{B}^d,\,\prod_{k=1}^d\beta_k=-1}(c-\Pr\{\boldsymbol{\eta}=\boldsymbol{\beta}\})\,F\left(\binom{\beta_1\boldsymbol{y}^1}{1},\binom{\beta_2\boldsymbol{y}^2}{1},\dots,\binom{\beta_d\boldsymbol{y}^d}{1}\right)\\ &\qquad+\sum_{\boldsymbol{\beta}\in\mathbb{B}^d,\,\prod_{k=1}^d\beta_k=1}(c+\Pr\{\boldsymbol{\eta}=\boldsymbol{\beta}\})\,F\left(\binom{\beta_1\boldsymbol{y}^1}{1},\binom{\beta_2\boldsymbol{y}^2}{1},\dots,\binom{\beta_d\boldsymbol{y}^d}{1}\right). \end{aligned} \tag{2.11}$$

If we let

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

then the coefficients of each term in (2.11) will be nonnegative. Therefore we are able to find $\boldsymbol{\beta}'\in\mathbb{B}^d$, such that

$$F\left(\binom{\beta_1'\boldsymbol{y}^1}{1},\binom{\beta_2'\boldsymbol{y}^2}{1},\dots,\binom{\beta_d'\boldsymbol{y}^d}{1}\right)\ge\tau_0 F(\bar{\boldsymbol{y}}^1,\bar{\boldsymbol{y}}^2,\dots,\bar{\boldsymbol{y}}^d), \tag{2.12}$$

where

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

Let us denote $\bar{\boldsymbol{z}}^k:=\binom{\beta_k'\boldsymbol{y}^k}{1}$ for $k=1,2,\dots,d$. Since $\|\boldsymbol{z}^k\|=\|\beta_k'\boldsymbol{y}^k\|\le 1$, we know that $(\bar{\boldsymbol{z}}^1,\bar{\boldsymbol{z}}^2,\dots,\bar{\boldsymbol{z}}^k)$ is a feasible solution for $(TP_{\bar{S}})$. By combing with (2.8), we have

$$\begin{aligned} F(\bar{\boldsymbol{z}}^1,\bar{\boldsymbol{z}}^2,\dots,\bar{\boldsymbol{z}}^d)&\ge\tau_0F(\bar{\boldsymbol{y}}^1,\bar{\boldsymbol{y}}^2,\dots,\bar{\boldsymbol{y}}^d)\\ &\ge 2^{-d}2^{-\frac{d}{2}}(n+1)^{-\frac{d-2}{2}}\,v(TP_{\bar{S}})\\ &=2^{-\frac{3d}{2}}(n+1)^{-\frac{d-2}{2}}\,v(TP_{\bar{S}}). \qquad\square \end{aligned}$$

One may notice that our proposed algorithm for solving $(TP_{\bar{S}})$ is very similar to Steps 2 and 3 of Algorithm 2.4.1, with only a minor modification at Step 3, namely we choose a solution in $\arg\max\left\{F\left(\binom{\beta_1 \boldsymbol{y}^1}{1}, \binom{\beta_2 \boldsymbol{y}^2}{1}, \ldots, \binom{\beta_d \boldsymbol{y}^d}{1}\right), \boldsymbol{\beta} \in \mathbb{B}^d\right\}$, instead of choosing a solution in $\arg\max\left\{F\left(\binom{\beta_1 \boldsymbol{y}^1/d}{1}, \binom{\beta_2 \boldsymbol{y}^2/d}{1}, \ldots, \binom{\beta_d \boldsymbol{y}^d/d}{1}\right), \boldsymbol{\beta} \in \mathbb{B}^d\right\}$. The reason to divide d at Step 3 in Algorithm 2.4.1 (to solve $(P_{\bar{S}})$) will become clear later. Finally, we remark again that it is unnecessary to enumerate all possible 2^d combinations in this step, as (2.11) 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.

2.4.4 Feasible Solution Assembling

Finally we come to the last step of the scheme. In Step 4 of Algorithm 2.4.1, 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 $\beta_\ell(d+1)\bar{\boldsymbol{z}}^\ell + \sum_{1\le k\le d, k\ne\ell} \beta_k \bar{\boldsymbol{z}}^k$ over all $\boldsymbol{\beta} \in \mathbb{B}^d$ and $1 \le \ell \le d$, and take the best possible solution; again, this will not change the theoretical performance ratio. The polarization formula at Step 4 of Algorithm 2.4.1 works for any fixed degree d, and we shall complete the final stage of the proof of Theorem 2.4.2. Specifically, we shall prove that by letting

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

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

$$p(\boldsymbol{z}) - \underline{v}(P_{\bar{S}}) \ge \tau(P_S)\,(v(P_{\bar{S}}) - \underline{v}(P_{\bar{S}}))\,. \tag{2.13}$$

First, the solution $(\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \ldots, \bar{\boldsymbol{z}}^d)$ as established at Step 3 of Algorithm 2.4.1 satisfies $\|\boldsymbol{z}^k\| \le 1/d$ (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 2.4.5 can show that

$$F(\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \ldots, \bar{\boldsymbol{z}}^d) \ge d^{-d} 2^{-\frac{3d}{2}} (n+1)^{-\frac{d-2}{2}} v(TP_{\bar{S}}) \ge 2^{-\frac{3d}{2}} d^{-d} (n+1)^{-\frac{d-2}{2}} v(P_{\bar{S}}). \tag{2.14}$$

It is easy to see that

$$2 \le |z_h(\beta)| \le 2d \text{ and } \|\boldsymbol{z}(\beta)\| \le (d+1)/d + (d-1)/d = 2. \tag{2.15}$$

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

$$\beta_1 = 1 \Longrightarrow f(\bar{\boldsymbol{z}}(\beta)) \geq (2d)^d \, \underline{v}(P_{\bar{S}}). \tag{2.16}$$

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

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

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

Suppose $\xi_1, \xi_2, \ldots, \xi_d$ are i.i.d. random variables, each taking values 1 and -1 with equal probability 1/2. By the link Lemma 2.2.1, noticing that $f\left(\bar{\boldsymbol{z}}(-\xi)\right) = f\left(-\bar{\boldsymbol{z}}(\xi)\right) = (-1)^d f\left(\bar{\boldsymbol{z}}(\xi)\right)$, we have

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

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

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

$$
\begin{aligned}
&-\frac{1}{4}\mathrm{E}\left[\left(f\left(\bar{\boldsymbol{z}}(\xi)\right)-(2d)^d\,\underline{v}(P_{\bar{S}})\right)\,\middle|\,\xi_1=1,\prod_{k=2}^{d}\xi_k=-1\right]\\
&+\frac{(-1)^{d-1}}{4}\mathrm{E}\left[\left(f\left(\bar{\boldsymbol{z}}(\xi)\right)-(2d)^d\,\underline{v}(P_{\bar{S}})\right)\,\middle|\,\xi_1=1,\prod_{k=2}^{d}\xi_k=(-1)^{d-1}\right]\\
&+\frac{(-1)^{d}}{4}\mathrm{E}\left[\left(f\left(\bar{\boldsymbol{z}}(\xi)\right)-(2d)^d\,\underline{v}(P_{\bar{S}})\right)\,\middle|\,\xi_1=1,\prod_{k=2}^{d}\xi_k=(-1)^{d}\right]\\
\le\ &\frac{1}{2}\mathrm{E}\left[\left(f\left(\bar{\boldsymbol{z}}(\xi)\right)-(2d)^d\,\underline{v}(P_{\bar{S}})\right)\,\middle|\,\xi_1=1,\prod_{k=2}^{d}\xi_k=1\right], \qquad (2.17)
\end{aligned}
$$

where the last inequality is due to (2.16). Therefore, there is a binary vector $\boldsymbol{\beta}'\in\mathbb{B}^d$ with $\beta_1'=\prod_{k=2}^{d}\beta_k'=1$, such that

$$
\begin{aligned}
f(\bar{\boldsymbol{z}}(\beta'))-(2d)^d\underline{v}(P_{\bar{S}}) &\ge 2d!F((d+1)\bar{\boldsymbol{z}}^1,\bar{\boldsymbol{z}}^2,\ldots,\bar{\boldsymbol{z}}^d)\\
&\ge 2^{-\frac{3d}{2}+1}(d+1)!d^{-d}(n+1)^{-\frac{d-2}{2}}v(P_{\bar{S}}),
\end{aligned}
$$

where the last step is due to (2.14).

Below we argue $\boldsymbol{z}=\arg\max\left\{p(\mathbf{0});p\left(\frac{\boldsymbol{z}(\beta)}{z_h(\beta)}\right),\boldsymbol{\beta}\in\mathbb{B}^d \text{ and } \beta_1=\prod_{k=2}^{d}\beta_k=1\right\}$ satisfies (2.13). In fact, if $-\underline{v}(P_{\bar{S}})\ge\tau(P_S)\left(v(P_{\bar{S}})-\underline{v}(P_{\bar{S}})\right)$, then $\mathbf{0}$ trivially satisfies (2.13), and so does $\boldsymbol{z}$ in this case. Otherwise, if $-\underline{v}(P_{\bar{S}})<\tau(P_S)\left(v(P_{\bar{S}})-\underline{v}(P_{\bar{S}})\right)$, then we have

$$
v(P_{\bar{S}})>(1-\tau(P_S))\left(v(P_{\bar{S}})-\underline{v}(P_{\bar{S}})\right)\ge\frac{v(P_{\bar{S}})-\underline{v}(P_{\bar{S}})}{2},
$$

which implies

$$
\begin{aligned}
f\left(\frac{\bar{\boldsymbol{z}}(\beta')}{2d}\right)-\underline{v}(P_{\bar{S}}) &\ge (2d)^{-d}2^{-\frac{3d}{2}+1}(d+1)!d^{-d}(n+1)^{-\frac{d-2}{2}}v(P_{\bar{S}})\\
&\ge \tau(P_S)\left(v(P_{\bar{S}})-\underline{v}(P_{\bar{S}})\right).
\end{aligned}
$$

The above inequality also implies that $f\left(\bar{\boldsymbol{z}}(\beta')/(2d)\right)>0$. Recall that $\beta_1'=1$ implies $z_h(\beta')>0$, and thus $2d/z_h(\beta')\ge 1$ by (2.15). Therefore, we have

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

This shows that $\boldsymbol{z}$ satisfies (2.13) in both cases, which concludes the whole proof.

Chapter 3
Extensions of the Constraint Sets

In this chapter, we shall extend the approximation methods for polynomial optimization discussed in Chap. 2. The extensions are focused on the constraint sets of the polynomial optimization models, including binary hypercube, hypercube, the Euclidean sphere, intersection of co-centered ellipsoids, a general convex compact set, and even a mixture of binary hypercube and the Euclidean sphere. These extensions are not straightforward generalizations of the approximation methods proposed before. Rather, they entail specifications to account for the different structures of the constraint sets at hand. The most noticeable novelty is in the decomposition routines, which play an instrumental role in designing approximation algorithms for multilinear form optimization models, as Sect. 2.1 already shows. Along with the approximation methods discussed in Chap. 2, we hope these extended techniques will be helpful in designing approximation methods when new models are encountered.

3.1 Hypercube and Binary Hypercube

The approximation methods proposed in Chap. 2 will be first extended to discrete models. In fact, discrete polynomial optimization models are commonly encountered, e.g., the graph partition problems and the satisfiability problems. This section will be concerned with the models where a polynomial function is optimized over the binary hypercube $\mathbb{B}^n$, with the objective being the four types of polynomial functions mentioned in Sect. 1.3.1. Specifically, the models are

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

$$\boxed{\begin{array}{ll}(H_B)\ \max & f(\boldsymbol{x})\\ \quad\ \text{s.t.} & \boldsymbol{x}\in\mathbb{B}^n\end{array}}$$

$$\boxed{\begin{array}{ll}(M_B) & \max\ f(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^s)\\ & \text{s.t.}\quad \boldsymbol{x}^k\in\mathbb{B}^{n_k},\ k=1,2,\ldots,s\end{array}}$$

$$\boxed{\begin{array}{ll}(P_B) & \max\ p(\boldsymbol{x})\\ & \text{s.t.}\quad \boldsymbol{x}\in\mathbb{B}^{n}\end{array}}$$

These four models will be studied in this section in the above order, each in one section. The latter model generalizes the former one, and each generalization follows a similar extension of the approximation methods discussed in Chap. 2. We shall also discuss polynomial optimization over hypercube as a byproduct. They are models $(T_{\bar{B}})$, $(H_{\bar{B}})$, $(M_{\bar{B}})$, and $(P_{\bar{B}})$, i.e., the respective models (T_B), (H_B), (M_B), and (P_B) with $\mathbb{B}$ being replaced by $\bar{\mathbb{B}}$. Remark the 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:

$$\begin{array}{ll}\max & p(\boldsymbol{x})\\ \text{s.t.} & x_i\in\{a^i_1,a^i_2,\cdots,a^i_{m_i}\},\ i=1,2,\ldots,n.\end{array}$$

All these models are NP-hard in general when the degree of the objective polynomial $d\geq 2$, though 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.,

$$\|\boldsymbol{F}\|_{\infty\mapsto 1}=\begin{array}{ll}\max & (\boldsymbol{x}^1)^{\mathrm{T}}\boldsymbol{F}\boldsymbol{x}^2\\ \text{s.t.} & \boldsymbol{x}^1\in\mathbb{B}^{n_1},\boldsymbol{x}^2\in\mathbb{B}^{n_2},\end{array}$$

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 [55] 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 [59], 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 [59] is $\Omega\left(\sqrt{\frac{\ln n_1}{n_1}}\right)$. However, no polynomial-time procedure is provided to find a corresponding approximate solution.

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 [39], 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 [87] 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 [23], where an $\Omega\left(1/\ln n\right)$-approximate ratio for (H_B) is proposed when the matrix $\boldsymbol{F}$ is diagonal-free. If the degree of the objective polynomial gets higher, the only approximation result in the literature is due to Khot and Naor [59] in considering homogeneous cubic polynomial, where an $\Omega\left(\sqrt{\frac{\ln n}{n}}\right)$-approximation bound is provided when the tensor $\boldsymbol{F}$ is square-free. In fact, being square-free (or in the matrix case diagonal-free) is somewhat necessary for 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 $P = NP$.

We shall 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 3.1.1 discusses the model (T_B). Essentially, we apply a similar approach as in Sect. 2.1, by relaxing the multilinear objective to a lower order multilinear form recursively. Notwithstanding the similarity to the continuous case, the discrete models need to be dealt with carefully in the design of the decomposition routine. Sections 3.1.2 and 3.1.3 discuss models (H_B) and (M_B), respectively. Both will rely on the application of multilinear form relaxations. After we have dealt with the models in multilinear objective function, we are in the position to solve the models in homogeneous form objective using two different versions of certain linkage identities, under the square-free assumption. General model (P_B) is discussed in Sect. 3.1.4, where the homogenization technique in Sect. 2.4 is modified and applied again. All these approximation algorithms can be applied to polynomial optimization over hypercube, which we will briefly discuss in Sect. 3.1.5.

3.1.1 Multilinear Form

The first discrete model in our discussion is to maximize a multilinear objective 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 $n_1 \le n_2 \le \cdots \le n_d$. Essentially, the approximation method follows a similar flow as we solve the model $(T_{\bar{S}})$ in Sect. 2.1: we first propose a base algorithm for the case $d = 2$, and then design a decomposition routine in order to enable a recursive scheme. Unlike $(T_{\bar{S}})$, the case $d = 2$ for (T_B) is already NP-hard. Fortunately, there is a readily available randomized approximation algorithm with constant approximation ratio, due to Alon and Naor [5], which is the following.

Theorem 3.1.1 *When $d = 2$, (T_B) admits a polynomial-time randomized approximation algorithm with approximation ratio at least* $\frac{2\ln(1+\sqrt{2})}{\pi} > 0.56$.

The algorithm is based on SDP relaxation and randomization, which is different from the approach of Goemans and Williamson [39] for the max-cut problem as described in Sect. 1.4.4. In fact, Alon and Naor's rounding technique uses the so-called Grothendieck's inequality. The approximation ratio is indeed the inverse of the so-called Grothendieck's constant, which is upper bounded by $\frac{\pi}{2\ln(1+\sqrt{2})}$ while the precise value is still unknown. For a complete proof of Theorem 3.1.1, one is referred to [5].

Let us now focus on the decomposition routine. 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

$$\begin{aligned} &\max\ F(\boldsymbol{X}, \boldsymbol{x}^3) \\ &\text{s.t.}\quad \boldsymbol{X} \in \mathbb{B}^{n_1 n_2}, \boldsymbol{x}^3 \in \mathbb{B}^{n_3}. \end{aligned}$$

This problem is exactly in the form of (T_B) when $d = 2$, which can be solved approximately with approximation ratio $\frac{2\ln(1+\sqrt{2})}{\pi}$ as stipulated in Theorem 3.1.1. 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 2.1.1 and 2.1.2 in Sect. 2.1.

Decomposition Routine 3.1.1

- *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 $\boldsymbol{x}^1 = \mathrm{sign}(\boldsymbol{\xi})$ *and* $\boldsymbol{x}^2 = \mathrm{sign}(\boldsymbol{\eta})$, *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 3.1.1 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 3.1.1, 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 bound for $d = 3$. By a recursive procedure, this approximation algorithm is readily extended to solve (T_B) with any fixed degree d.

Theorem 3.1.2 *(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 in Theorem 3.1.1 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

$$\begin{aligned} F\left(\hat{\boldsymbol{X}},\hat{\boldsymbol{x}}^2,\hat{\boldsymbol{x}}^3,\ldots,\hat{\boldsymbol{x}}^{d-1}\right) &\geq (2/\pi)^{d-2}\ln\left(1+\sqrt{2}\right)\left(\textstyle\prod_{k=2}^{d-2} n_k\right)^{-\frac{1}{2}} v(\tilde{T}_B) \\ &\geq (2/\pi)^{d-2}\ln\left(1+\sqrt{2}\right)\left(\textstyle\prod_{k=2}^{d-2} n_k\right)^{-\frac{1}{2}} v(T_B). \end{aligned}$$

Rewrite $\hat{\boldsymbol{X}}$ as an $n_1\times n_d$ matrix, construct $\tilde{\boldsymbol{X}} = \begin{bmatrix} \boldsymbol{I}_{n_1\times n_1} & \hat{\boldsymbol{X}}/\sqrt{n_1} \\ \hat{\boldsymbol{X}}^{\mathrm{T}}/\sqrt{n_1} & \hat{\boldsymbol{X}}^{\mathrm{T}}\hat{\boldsymbol{X}}/n_1 \end{bmatrix}$ as in DR 3.1.1, 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, it follows from [18] 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

$$\begin{aligned}
\mathrm{E}\left[F\left(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d\right)\right] &= \mathrm{E}\left[\sum_{1\le i\le n_1, 1\le j\le n_d} \hat{x}_i^1 \hat{Q}_{ij} \hat{x}_j^d\right] \\
&= \sum_{1\le i\le n_1, 1\le j\le n_d} \hat{Q}_{ij}\, \mathrm{E}\left[\hat{x}_i^1 \hat{x}_j^d\right] \\
&= \sum_{1\le i\le n_1, 1\le j\le n_d} \hat{Q}_{ij} \frac{2}{\pi} \hat{X}_{ij} \arcsin \frac{1}{\sqrt{n_1}} \\
&= \frac{2}{\pi} \arcsin \frac{1}{\sqrt{n_1}} \sum_{1\le i\le n_1, 1\le j\le n_d} \hat{Q}_{ij} \hat{X}_{ij} \\
&= \frac{2}{\pi} \arcsin \frac{1}{\sqrt{n_1}} F\left(\hat{\boldsymbol{X}}, \hat{\boldsymbol{x}}^2, \hat{\boldsymbol{x}}^3, \cdots, \hat{\boldsymbol{x}}^{d-1}\right) \\
&\ge \frac{2}{\pi\sqrt{n_1}} \left(\frac{2}{\pi}\right)^{d-2} \ln\left(1+\sqrt{2}\right) \left(\prod_{k=2}^{d-2} n_k\right)^{-\frac{1}{2}} v(T_B) \\
&= \left(\frac{2}{\pi}\right)^{d-1} \ln\left(1+\sqrt{2}\right) \left(\prod_{k=1}^{d-2} n_k\right)^{-\frac{1}{2}} v(T_B). \qquad (3.1)
\end{aligned}$$

Thus $\hat{\boldsymbol{x}}^1$ and $\hat{\boldsymbol{x}}^d$ can be found by a randomization process. This paves the way to solve the problem recursively. □

The algorithm for solving general model (T_B) is summarized below. This algorithm is similar to Algorithm 2.1.3, with a major difference being the different decomposition routines used, and also the procedure to solve the initial step of $d = 2$.

Algorithm 3.1.2

- *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 in Theorem 3.1.1 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 3.1.1, 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* $(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\cdots,\hat{\boldsymbol{x}}^d)$.

3.1.2 Homogeneous Form

We now consider the model of maximizing a homogeneous form over binary hypercube

$$\boxed{\begin{array}{ll}(H_B) & \max\ f(\boldsymbol{x}) \\ & \text{s.t.}\quad \boldsymbol{x} \in \mathbb{B}^n\end{array}}$$

As before, we propose polynomial-time randomized approximation algorithms of (H_B) for any fixed degree d. First, 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]). Like the model $(H_{\bar{S}})$, the key link from multilinear form $F(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^d)$ to the homogeneous form $f(\boldsymbol{x})$ is Lemma 2.2.1. 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.

Lemma 3.1.3 *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* $p(\hat{\boldsymbol{x}}) \le p(\boldsymbol{z}) \le p(\tilde{\boldsymbol{x}})$.

Proof. Since $p(\boldsymbol{x})$ is square-free, by fixing $x_2,x_3,\ldots,x_n$ as constants and taking x_1 as an independent variable, we may write

$$p(\boldsymbol{x}) = g_1(x_2,x_3,\ldots,x_n) + x_1 g_2(x_2,x_3,\ldots,x_n).$$

Let

$$\hat{x}_1 = \begin{cases} -1 & g_2(z_2,z_3,\cdots,z_n) \ge 0, \\ 1 & g_2(z_2,z_3,\cdots,z_n) < 0. \end{cases}$$

Then

$$p\left((\hat{x}_1,z_2,z_3,\cdots,z_n)^{\mathrm{T}}\right) \le p(\boldsymbol{z}).$$

Repeat the same procedures for $z_2,z_3,\cdots,z_n$, and let them be replaced by binary scales $\hat{x}_2,\hat{x}_3,\ldots,\hat{x}_n$, respectively. Then $\hat{\boldsymbol{x}} = (\hat{x}_1,\hat{x}_2,\ldots,\hat{x}_n)^{\mathrm{T}} \in \mathbb{B}^n$ satisfies $p(\hat{\boldsymbol{x}}) \le p(\boldsymbol{z})$. Using a similar procedure, we may find $\tilde{\boldsymbol{x}} \in \mathbb{B}^n$ with $p(\tilde{\boldsymbol{x}}) \ge p(\boldsymbol{z})$. □

Lemma 3.1.3 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, from the approximation result for (T_B) in Theorem 3.1.2, together with Lemma 3.1.3 and the link Lemma 2.2.1, we present the main results in this section.

Theorem 3.1.4 *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 supersymmetric, and (H_B) can be relaxed to

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

By Theorem 3.1.2 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

$$\begin{aligned} F(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\cdots,\hat{\boldsymbol{x}}^d) &\geq \left(\frac{2}{\pi}\right)^{d-1} \ln(1+\sqrt{2})\, n^{-\frac{d-2}{2}} v(\tilde{H}_B) \\ &\geq \left(\frac{2}{\pi}\right)^{d-1} \ln(1+\sqrt{2})\, n^{-\frac{d-2}{2}} v(H_B). \end{aligned}$$

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 2.2.1 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 $1 \leq j \leq n$,

$$\left|\left(\frac{1}{d}\sum_{k=1}^d \left(\prod_{i\neq k}\beta_i\right)\hat{\boldsymbol{x}}^k\right)_j\right| = \frac{1}{d}\left|\sum_{k=1}^d \left(\prod_{i\neq k}\beta_i\right)\hat{x}^k_j\right| \leq \frac{1}{d}\sum_{k=1}^d \left|\left(\prod_{i\neq k}\beta_i\right)\hat{x}^k_j\right| = 1. \quad (3.2)$$

Since $f(\boldsymbol{x})$ is square-free, by Lemma 3.1.3 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 3.1.5 *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 3.1.4, 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 (3.2), we have $\frac{1}{d}\sum_{k=1}^{d}\xi_k\hat{\boldsymbol{x}}^k \in \bar{\mathbb{B}}^n$. Then by Lemma 3.1.3, 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 2.2.1 and we have

$$\begin{aligned}
&\frac{1}{2}\mathrm{E}\left[f\left(\frac{1}{d}\sum_{k=1}^{d}\xi_k\hat{\boldsymbol{x}}^k\right) - \underline{v}(H_B) \,\middle|\, \prod_{i=1}^{d}\xi_i = 1\right] \\
&\geq \frac{1}{2}\mathrm{E}\left[f\left(\frac{1}{d}\sum_{k=1}^{d}\xi_k\hat{\boldsymbol{x}}^k\right) - \underline{v}(H_B) \,\middle|\, \prod_{i=1}^{d}\xi_i = 1\right] \\
&\quad -\frac{1}{2}\mathrm{E}\left[f\left(\frac{1}{d}\sum_{k=1}^{d}\xi_k\hat{\boldsymbol{x}}^k\right) - \underline{v}(H_B) \,\middle|\, \prod_{i=1}^{d}\xi_i = -1\right] \\
&= \mathrm{E}\left[\prod_{i=1}^{d}\xi_i \left(f\left(\frac{1}{d}\sum_{k=1}^{d}\xi_k\hat{\boldsymbol{x}}^k\right) - \underline{v}(H_B)\right)\right]
\end{aligned}$$

$$= d^{-d}\mathrm{E}\left[\prod_{i=1}^{d}\xi_i f\left(\sum_{k=1}^{d}\xi_k\hat{\boldsymbol{x}}^k\right)\right] - \underline{v}(H_B)\,\mathrm{E}\left[\prod_{i=1}^{d}\xi_i\right]$$
$$= d^{-d}d!F(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\ldots,\hat{\boldsymbol{x}}^d) \geq \tau(H_B)\,v(\tilde{H}_B) \geq (\tau(H_B)/2)\,(v(H_B)-\underline{v}(H_B)),$$

where the last inequality is due to $2v(\tilde{H}_B) \geq v(H_B) - \underline{v}(H_B)$. 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 3.1.3, by the square-free property of $f(\boldsymbol{x})$, we are able to find $\tilde{\boldsymbol{x}} \in \mathbb{B}^n$ with

$$f(\tilde{\boldsymbol{x}}) - \underline{v}(H_B) \geq f\left(\frac{1}{d}\sum_{k=1}^{d}\beta_k\hat{\boldsymbol{x}}^k\right) - \underline{v}(H_B) \geq \tau(H_B)\,(v(H_B)-\underline{v}(H_B)). \qquad \square$$

To conclude this section, we summarize the approximation algorithm for (H_B) below (independent of d being odd or even).

Algorithm 3.1.3

```
• INPUT: a d-th order supersymmetric square-free tensor $\boldsymbol{F} \in \mathbb{R}^{n^d}$.
1 Apply Algorithm 3.1.2 to solve the problem
      max $F(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^d)$
      s.t. $\boldsymbol{x}^k \in \mathbb{B}^n, k=1,2,\ldots,d$
  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 3.1.3, with input $\hat{\boldsymbol{x}} \in \bar{\mathbb{B}}^n$ 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 $\tilde{\boldsymbol{x}} \in \mathbb{B}^n$.
```

3.1.3 Mixed Form

We further move on to consider the mixed-form optimization model

$$\boxed{\begin{array}{ll}(M_B) & \max\ f(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^s)\\ & \text{s.t.}\quad \boldsymbol{x}^k \in \mathbb{B}^{n_k},\ k=1,2,\ldots,s\end{array}}$$

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, $n_1 \le n_2 \le \cdots \le n_s$, and $d = d_1 + d_2 + \cdots + d_s$ is deemed as a fixed constant. This model is a generalization of (T_B) in Sect. 3.1.1 and (H_B) in Sect. 3.1.2, 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 form, which leads to (T_B). After solving (T_B) approximately by Theorem 3.1.2, we are able to adjust the solutions step by step, or use Lemma 2.3.3. The following approximation results are presented, which are comparable to the ones in Sect. 3.1.2.

Theorem 3.1.6 *If $f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s)$ is square-free in each $\boldsymbol{x}^k$ $(k = 1, 2, \ldots, s)$, $d \ge 3$ 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*

$$\tau(M_B) := \tilde{\tau}(M_S) \left(\frac{2}{\pi}\right)^{d-1} \ln\left(1+\sqrt{2}\right) \prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}} = \Omega\left(\tilde{\tau}(M_S)\right)$$

$$= \begin{cases} \left(\frac{2}{\pi}\right)^{d-1} \ln\left(1+\sqrt{2}\right) \left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}}\right) \left(\frac{\prod_{k=1}^{s-1} {n_k}^{d_k}}{n_{s-1}}\right)^{-\frac{1}{2}} & d_s = 1, \\ \left(\frac{2}{\pi}\right)^{d-1} \ln\left(1+\sqrt{2}\right) \left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}}\right) \left(\frac{\prod_{k=1}^{s} {n_k}^{d_k}}{{n_s}^2}\right)^{-\frac{1}{2}} & d_s \ge 2. \end{cases}$$

Proof. Like in the proof of Theorem 3.1.4, 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

$$F(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d) \ge \tau(M_B) \left(\prod_{k=1}^{s} \frac{{d_k}^{d_k}}{d_k!}\right) v(M_B).$$

Let $\boldsymbol{\xi} = (\xi_1, \xi_2, \cdots, \xi_d)^{\mathrm{T}}$, whose components are i.i.d. random variables, taking values 1 and -1 with equal probability $1/2$. Similar as (2.5), we denote

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

Without loss of generality, we assume d_1 to be odd. Applying Lemma 2.3.3 we have

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

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 (3.2), 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 3.1.3, 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

$$f(\tilde{\boldsymbol{x}}^1,\tilde{\boldsymbol{x}}^2,\cdots,\tilde{\boldsymbol{x}}^s) \geq f\left(\prod_{i=1}^{d}\beta_i\frac{\hat{\boldsymbol{x}}_\beta^1}{d_1},\frac{\hat{\boldsymbol{x}}_\beta^2}{d_2},\cdots,\frac{\hat{\boldsymbol{x}}_\beta^s}{d_s}\right) \geq \tau(M_B)\,v(M_B). \qquad \square$$

Theorem 3.1.7 *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 $\tau(M_B)$.*

The proof is analogous to that of Theorem 3.1.5. The main differences are: (1) we use Lemma 2.3.3 instead of invoking Lemma 2.2.1 directly; and (2) 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.

3.1.4 Inhomogeneous Polynomial

Finally, we consider binary integer programming model to the optimization on a generic (inhomogeneous) polynomial function, i.e.,

$$\boxed{\begin{array}{ll}(P_B) & \max\ p(\boldsymbol{x})\\ & \text{s.t.}\quad \boldsymbol{x} \in \mathbb{B}^n\end{array}}$$

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 optimization discussed in previous sections. The analysis here is similar to Sect. 2.4, which deals with *homogenization* directly. 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 $i = 1,2,\ldots,n$, allowing us to reduce the power of x_i to 0 or 1 in each monomial of $p(\boldsymbol{x})$. We now propose the following approximation algorithm for solving (P_B).

Algorithm 3.1.4

```
• 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
  $p_0(\boldsymbol{x}) - p_0(\boldsymbol{0}) = F(\underbrace{\bar{\boldsymbol{x}},\bar{\boldsymbol{x}},\cdots,\bar{\boldsymbol{x}}}_{d})$ when $x_h = 1$ as in (2.7), with $\boldsymbol{F}$ being an $(n+1)$-
  dimensional d-th order supersymmetric tensor.
2 Apply Algorithm 3.1.2 to solve the problem
      $\max\ F(\bar{\boldsymbol{x}}^1,\bar{\boldsymbol{x}}^2,\cdots,\bar{\boldsymbol{x}}^d)$
      $\text{s.t.}\ \ \bar{\boldsymbol{x}}^k \in \mathbb{B}^{n+1},\ k=1,2,\ldots,d$
  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 $\boldsymbol{z} = \arg\max\left\{p_0(\boldsymbol{0});\ p_0\left(\boldsymbol{z}(\beta)/z_h(\beta)\right),\boldsymbol{\beta}\in\mathbb{B}^d \text{ and } \beta_1 = \prod_{k=2}^d \beta_k = 1\right\}$,
  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 3.1.3, 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 $p_0(\boldsymbol{y}) \ge p_0(\boldsymbol{z})$.
• OUTPUT: a feasible solution $\boldsymbol{y}\in\mathbb{B}^n$.
```

Before presenting the main result to analyze Algorithm 3.1.4, 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, which plays an important role in the analysis of the algorithm for (P_B).

Lemma 3.1.8 *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 $\underline{v}(P_B) \le 0 \le v(P_B)$, and a binary vector $\tilde{\boldsymbol{x}}\in\mathbb{B}^n$ can be found in polynomial time with $p(\tilde{\boldsymbol{x}}) \ge 0$.*

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 $F_{i_1i_2\ldots i_k}x_{i_1}x_{i_2}\cdots x_{i_k}$ with degree $k\,(1\le k\le d)$ of $p(\boldsymbol{x})$, by the square-free property, it follows that

$$\mathrm{E}[F_{i_1i_2\ldots i_k}\xi_{i_1}\xi_{i_2}\cdots\xi_{i_k}] = F_{i_1i_2\cdots i_k}\mathrm{E}[\xi_{i_1}]\mathrm{E}[\xi_{i_2}]\cdots\mathrm{E}[\xi_{i_k}] = 0.$$

This implies $\mathrm{E}[p(\boldsymbol{\xi})] = 0$, and consequently $\underline{v}(P_B) \le 0 \le v(P_B)$. By a randomization process, a binary vector $\tilde{\boldsymbol{x}}\in\mathbb{B}^n$ can be found in polynomial time with $p(\tilde{\boldsymbol{x}}) \ge 0$. □

We remark that the second part of Lemma 3.1.8 can also be proven by conducting the procedure in Lemma 3.1.3 with the input vector $\boldsymbol{0}\in\bar{\mathbb{B}}^n$, since $p(\boldsymbol{0}) = 0$. Therefore, finding a binary vector $\tilde{\boldsymbol{x}}\in\mathbb{B}^n$ with $p(\tilde{\boldsymbol{x}}) \ge 0$ can be done by either a randomized process (Lemma 3.1.8) or a deterministic process (Lemma 3.1.3). We now present the main result in this section.

Theorem 3.1.9 (P_B) *admits a polynomial-time randomized approximation algorithm with relative approximation ratio* $\tau(P_B)$*, where*

$$\tau(P_B) := \frac{\ln\left(1+\sqrt{2}\right)}{2(1+\mathrm{e})\pi^{d-1}}(d+1)!\,d^{-2d}(n+1)^{-\frac{d-2}{2}} = \Omega\left(n^{-\frac{d-2}{2}}\right).$$

The main idea of the proof is quite similar to that of Theorem 2.4.2. However, the discrete nature of the problem, as well as the non-convexity of the feasible region, require an extra dose of caution when dealing with the specific details. The entire proof is presented below.

Proof. Since we are working with relative approximation ratio, by Step 1 of Algorithm 3.1.4, we may assume that $p(\boldsymbol{x})$ is square-free and has no constant term. Then by homogenization (see (2.7)), we have

$$p(\boldsymbol{x}) = F\Bigg(\underbrace{\begin{pmatrix}\boldsymbol{x}\\ x_h\end{pmatrix},\begin{pmatrix}\boldsymbol{x}\\ x_h\end{pmatrix},\cdots,\begin{pmatrix}\boldsymbol{x}\\ x_h\end{pmatrix}}_{d}\Bigg) = F(\underbrace{\bar{\boldsymbol{x}},\bar{\boldsymbol{x}},\cdots,\bar{\boldsymbol{x}}}_{d}) = f(\bar{\boldsymbol{x}}),$$

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 supersymmetric 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 3.1.2 with

$$\begin{aligned} F(\bar{\boldsymbol{u}}^1,\bar{\boldsymbol{u}}^2,\cdots,\bar{\boldsymbol{u}}^d) &\ge (2/\pi)^{d-1}\ln(1+\sqrt{2})(n+1)^{-\frac{d-2}{2}}v(\tilde{P}_B) \\ &\ge (2/\pi)^{d-1}\ln(1+\sqrt{2})(n+1)^{-\frac{d-2}{2}}v(P_B). \end{aligned}$$

Denote $\bar{\boldsymbol{v}}^k = \bar{\boldsymbol{u}}^k/d$ for $k=1,2,\ldots,d$, and consequently

$$\begin{aligned} F(\bar{\boldsymbol{v}}^1,\bar{\boldsymbol{v}}^2,\cdots,\bar{\boldsymbol{v}}^d) &= d^{-d}F(\bar{\boldsymbol{u}}^1,\bar{\boldsymbol{u}}^2,\cdots,\bar{\boldsymbol{u}}^d) \\ &\ge (2/\pi)^{d-1}\ln(1+\sqrt{2})d^{-d}(n+1)^{-\frac{d-2}{2}}v(P_B). \end{aligned}$$

Notice that for all $1 \le k \le d$, $|v_h^k| = |u_h^k/d| = 1/d \le 1$ and the last component of tensor $\boldsymbol{F}$ is 0. By applying Lemma 2.4.4, it follows that

$$\mathrm{E}\left[\prod_{k=1}^{d} \eta_k F\left(\binom{\eta_1 \boldsymbol{v}^1}{1}, \binom{\eta_2 \boldsymbol{v}^2}{1}, \cdots, \binom{\eta_d \boldsymbol{v}^d}{1}\right)\right] = F(\bar{\boldsymbol{v}}^1, \bar{\boldsymbol{v}}^2, \cdots, \bar{\boldsymbol{v}}^d)$$

and

$$\mathrm{E}\left[F\left(\binom{\xi_1 \boldsymbol{v}^1}{1}, \binom{\xi_2 \boldsymbol{v}^2}{1}, \cdots, \binom{\xi_d \boldsymbol{v}^d}{1}\right)\right] = 0,$$

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:

$$\begin{aligned}
&F(\bar{\boldsymbol{v}}^1, \bar{\boldsymbol{v}}^2, \cdots, \bar{\boldsymbol{v}}^d) \\
&= \sum_{\boldsymbol{\beta} \in \mathbb{B}^d, \prod_{k=1}^d \beta_k = -1} (c - \Pr\{\boldsymbol{\eta} = \boldsymbol{\beta}\}) F\left(\binom{\beta_1 \boldsymbol{v}^1}{1}, \binom{\beta_2 \boldsymbol{v}^2}{1}, \cdots, \binom{\beta_d \boldsymbol{v}^d}{1}\right) \\
&\quad + \sum_{\boldsymbol{\beta} \in \mathbb{B}^d, \prod_{k=1}^d \beta_k = 1} (c + \Pr\{\boldsymbol{\eta} = \boldsymbol{\beta}\}) F\left(\binom{\beta_1 \boldsymbol{v}^1}{1}, \binom{\beta_2 \boldsymbol{v}^2}{1}, \cdots, \binom{\beta_d \boldsymbol{v}^d}{1}\right).
\end{aligned}$$

If we let $c = \max_{\boldsymbol{\beta} \in \mathbb{B}^d, \prod_{k=1}^d \beta_k = -1} \Pr\{\boldsymbol{\eta} = \boldsymbol{\beta}\}$, 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

$$F\left(\binom{\beta_1' \boldsymbol{v}^1}{1}, \binom{\beta_2' \boldsymbol{v}^2}{1}, \cdots, \binom{\beta_d' \boldsymbol{v}^d}{1}\right) \ge \tau_1 F(\bar{\boldsymbol{v}}^1, \bar{\boldsymbol{v}}^2, \cdots, \bar{\boldsymbol{v}}^d),$$

with

$$\begin{aligned}
\tau_1 &= \left(\sum_{\boldsymbol{\beta} \in \mathbb{B}^d, \prod_{k=1}^d \beta_k = 1} (c + \Pr\{\boldsymbol{\eta} = \boldsymbol{\beta}\}) + \sum_{\boldsymbol{\beta} \in \mathbb{B}^d, \prod_{k=1}^d \beta_k = -1} (c - \Pr\{\boldsymbol{\eta} = \boldsymbol{\beta}\})\right)^{-1} \\
&\ge \left(2^d c + 1\right)^{-1} \ge \left(2^d \left(\frac{1}{2} + \frac{1}{2d}\right)^d + 1\right)^{-1} \ge \frac{1}{1+e},
\end{aligned}$$

where $c \le \left(\frac{1}{2} + \frac{1}{2d}\right)^d$ 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

$$F(\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \cdots, \bar{\boldsymbol{z}}^d) \ge \tau_1 F(\bar{\boldsymbol{v}}^1, \bar{\boldsymbol{v}}^2, \cdots, \bar{\boldsymbol{v}}^d) \ge \left(\frac{2}{\pi}\right)^{d-1} \frac{\ln\left(1+\sqrt{2}\right)}{1+e} d^{-d} (n+1)^{-\frac{d-2}{2}} v(P_B).$$

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 $1 \le k \le d$ and $1 \le i \le n$, it follows that

$$2 \le |z_h(\beta)| \le 2d \text{ and } |z_i(\beta)| \le (d+1)/d + (d-1)/d = 2 \quad \forall 1 \le i \le n.$$

Thus $\boldsymbol{z}(\beta)/z_h(\beta) \in \bar{\mathbb{B}}^n$. By Lemma 3.1.3, there exists $\boldsymbol{x}' \in \mathbb{B}^n$, such that

$$\underline{v}(P_B) \le p(\boldsymbol{x}') \le p\left(\boldsymbol{z}(\beta)/z_h(\beta)\right) = f\left(\bar{\boldsymbol{z}}(\beta)/z_h(\beta)\right).$$

Moreover, we shall argue below that

$$\beta_1 = 1 \Longrightarrow f(\bar{\boldsymbol{z}}(\beta)) \ge (2d)^d\, \underline{v}(P_B). \tag{3.3}$$

If this were not the case, then by Lemma 3.1.8 $f\left(\bar{\boldsymbol{z}}(\beta)/(2d)\right) < \underline{v}(P_B) \le 0$. Notice that $\beta_1 = 1$ implies $z_h(\beta) > 0$, and thus we have

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

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 (3.3) holds and using the same argument as (2.17), we get

$$\frac{1}{2}\mathrm{E}\left[\left(f\left(\bar{\boldsymbol{z}}(\xi)\right) - (2d)^d\, \underline{v}(P_B)\right) \middle| \xi_1 = 1, \prod_{k=2}^d \xi_k = 1\right] \ge d!F\left((d+1)\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \cdots, \bar{\boldsymbol{z}}^d\right).$$

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

$$\begin{aligned} f(\bar{\boldsymbol{z}}(\beta'')) - (2d)^d\, \underline{v}(P_B) &\ge 2d!F((d+1)\bar{\boldsymbol{z}}^1, \bar{\boldsymbol{z}}^2, \cdots, \bar{\boldsymbol{z}}^d) \\ &\ge \left(\frac{2}{\pi}\right)^{d-1} \frac{2\ln\left(1+\sqrt{2}\right)}{1+\mathrm{e}} (d+1)!\, d^{-d} (n+1)^{-\frac{d-2}{2}}\, v(P_B). \end{aligned}$$

By Lemma 3.1.8, a binary vector $\boldsymbol{x}' \in \mathbb{B}^n$ can be found in polynomial time with $p(\boldsymbol{x}') \ge 0$. Moreover, as $\boldsymbol{z}(\beta'')/z_h(\beta'') \in \bar{\mathbb{B}}^n$, by Lemma 3.1.3, another binary vector $\boldsymbol{x}'' \in \mathbb{B}^n$ can be found in polynomial time with $p(\boldsymbol{x}'') \ge p(\boldsymbol{z}(\beta'')/z_h(\beta''))$. Below we shall prove at least one of $\boldsymbol{x}'$ and $\boldsymbol{x}''$ satisfies

$$p(\boldsymbol{x}) - \underline{v}(P_B) \ge \tau(P_B)\left(v(P_B) - \underline{v}(P_B)\right). \tag{3.4}$$

Indeed, if $-\underline{v}(P_B) \ge \tau(P_B)\left(v(P_B) - \underline{v}(P_B)\right)$, then $\boldsymbol{x}'$ satisfies (3.4) in this case. Otherwise we shall have $-\underline{v}(P_B) < \tau(P_B)\left(v(P_B) - \underline{v}(P_B)\right)$, then

$$v(P_B) > \left(1 - \tau(P_B)\right)\left(v(P_B) - \underline{v}(P_B)\right) \ge \left(v(P_B) - \underline{v}(P_B)\right)/2,$$

which implies

$$\begin{aligned}
&f\left(\frac{\bar{z}(\beta'')}{2d}\right) - \underline{v}(P_B) \\
&\geq (2d)^{-d}\left(\frac{2}{\pi}\right)^{d-1}\frac{2\ln\left(1+\sqrt{2}\right)}{1+\mathrm{e}}(d+1)!\,d^{-d}(n+1)^{-\frac{d-2}{2}}v(P_B) \\
&\geq \tau(P_B)\left(v(P_B) - \underline{v}(P_B)\right).
\end{aligned}$$

The above inequality also implies that $f(\bar{z}(\beta'')/(2d)) > 0$. Recall that $\beta_1'' = 1$ implies $z_h(\beta'') > 0$. Therefore,

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

which implies that $\boldsymbol{x}''$ satisfies (3.4). Finally, $\arg\max\{p(\boldsymbol{x}'), p(\boldsymbol{x}'')\}$ satisfies (3.4) 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 in discrete values:

$$\begin{array}{lll}
(PD) & \max & p(\boldsymbol{x}) \\
& \text{s.t.} & x_i \in \{a_1^i, a_2^i, \cdots, a_{m_i}^i\},\ i = 1,2,\ldots,n.
\end{array}$$

To see this, we observe that by adopting the Lagrange interpolation technique and letting

$$x_i = \sum_{j=1}^{m_i} a_j^i \prod_{1\leq k\leq m_i, k\neq j} \frac{u_i - k}{j-k} \quad \forall\, 1 \leq i \leq n,$$

the original decision variables can be equivalently transformed to

$$u_i = j \Longrightarrow x_i = a_j^i \quad \forall\, 1 \leq i \leq n,\ 1 \leq j \leq m_i,$$

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 $\max_{1\leq i\leq n}\{d(m_i - 1)\}$, 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, and the dimension of its decision variables is $\sum_{i=1}^n \lceil \log_2 m_i \rceil$.

3.1.5 Hypercube

Let us turn back to the continuous polynomial optimization models. The optimization of polynomial function over hypercube ($\bar{\mathbb{B}}^n$) also has versatile applications. Basically, it includes finding the maximum (or minimum) value of a polynomial function with the variables having lower and upper bounds. As a byproduct of the analysis in this section, we remark that all the approximation algorithms proposed in previous sections are also applicable for polynomial optimization over hypercube, which are models $(T_{\bar{B}}), (H_{\bar{B}}), (M_{\bar{B}})$, and $(P_{\bar{B}})$, 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 objective and mixed-form objective, and consequently Algorithms 3.1.3 and 3.1.4 can be made simpler without going through the process in Lemma 3.1.3. We now conclude this section, by the following theorem without proof. Interested readers can take it as a good exercise.

Theorem 3.1.10 *The following approximation results hold for polynomial optimization over hypercube:*

1. *$(T_{\bar{B}})$ admits a polynomial-time randomized approximation algorithm with approximation ratio $\tau(T_B)$.*
2. *If $d \geq 3$ is odd, then $(H_{\bar{B}})$ admits a polynomial-time randomized approximation algorithm with approximation ratio $\tau(H_B)$; otherwise $d \geq 4$ 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 $(M_{\bar{B}})$ 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 $(M_{\bar{B}})$ admits a polynomial-time randomized approximation algorithm with relative approximation ratio $\tau(M_B)$.*
4. *$(P_{\bar{B}})$ admits a polynomial-time randomized approximation algorithm with relative approximation ratio $\tau(P_B)$.*

3.2 The Euclidean Sphere

The Euclidean sphere ($\mathbb{S}^n$) is the boundary of the Euclidean ball ($\bar{\mathbb{S}}^n$), or conversely, the Euclidean ball is the convex hull of the Euclidean sphere. In Chap. 2, we discussed various polynomial optimization models with the constraint set being the Euclidean ball. In fact, the spherically constrained polynomial optimization might be encountered more frequently than the ball-constrained polynomial optimization (see the discussion in Chap. 4), albeit in many cases they are equivalent. Here in this section, we shall discuss the approximation methods for homogeneous polynomial optimization over the Euclidean sphere, i.e.,

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

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

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

In Sect. 2.1, we mentioned the equivalence relationship between (T_S) and $(T_{\bar{S}})$. This is because the optimal value of $(T_{\bar{S}})$ is always positive unless the data $\boldsymbol{F}$ is a zero tensor. Therefore, the corresponding optimal solution must have the Euclidean norm to be one (otherwise one may simply scale the solution along its direction to make itself in the Euclidean sphere), and this makes itself being optimal to (T_S). In general, for the homogeneous polynomial optimization models proposed above, we have the following equivalence.

Proposition 3.2.1 *If $v(T_S)>0$ (respectively $v(H_S)>0$, and $v(M_S)>0$), then (T_S) (respectively (H_S), and (M_S)) is equivalent to $(T_{\bar{S}})$ (respectively $(H_{\bar{S}})$, and $(M_{\bar{S}})$). In particular, the equivalence holds for (T_S), (H_S) with odd d, and (M_S) with one of $d_k\,(k=1,2,\ldots,s)$ being odd.*

Now, the only remaining cases left are the even cases of (H_S) and (M_S), or specifically, (H_S) with even d, and (M_S) with all $d_k\,(k=1,2,\ldots,s)$ being even. In fact, the even case of (H_S) is a special one of the even case of (M_S). Unfortunately, when $d\geq 4$, it is NP-hard to check whether $v(M_S)>0$ (or $v(H_S)>0$) for its even case, otherwise we may at least reserve the first part of Proposition 3.2.1 for help. In order to express the main ideas in handling these even cases, let us first focus on (H_S) with even d.

When d is even, the only case of (H_S) that can be solved in polynomial time is the quadratic one, which is exactly the largest eigenvalue problem for a symmetric matrix. Even worse, we have the following inapproximate result, whose proof is almost same as that of Proposition 2.2.3

Proposition 3.2.2 *If $d=4$, then there is no polynomial-time approximation algorithm with a positive approximation ratio for (H_S) unless $P=NP$.*

Therefore, the only hope is a polynomial-time approximation algorithm with a relative approximation ratio, similar as the even case of $(H_{\bar{S}})$. However, unlike the model $(H_{\bar{S}})$, whose optimal value is always nonnegative, since $\mathbf{0}$ is always a feasible solution, the spherically constrained optimization does not have such sign properties. Specifically, inequality (2.3) in the proof of Theorem 2.2.4 no longer holds here. In order to overcome this difficulty, a feasible solution should be generated as a bench mark. Below we introduce the algorithm for (H_S) for even d.

Algorithm 3.2.1

```
• INPUT: a d-th order supersymmetric 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 supersymmetric 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 2.1.3 to solve the problem
      $\max\ F(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^d) - f(\boldsymbol{x}^0)H(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^d)$
      $\text{s.t.}\ \ \boldsymbol{x}^k \in \mathbb{S}^n,\ k = 1,2,\ldots,d$
  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 $\hat{\boldsymbol{x}} \in \mathbb{S}^n$.
```

A similar approximation result as of Theorem 2.2.4 can be derived, which preserves the same approximation ratio for (H_S) and $(H_{\bar{S}})$.

Theorem 3.2.3 *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 supersymmetric 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:

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

Applying Algorithm 2.1.3 (notice in this case that the ball constraint and the spherical constraint are equivalent), we obtain a solution $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$ in polynomial time, and by Theorem 2.1.5 we have

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

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

Let us first work on the case that

$$f(\boldsymbol{x}^0) - \underline{v}(H_S) \leq (\tilde{\tau}(T_S)/4)\,(v(H_S) - \underline{v}(H_S)). \tag{3.5}$$

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

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

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 $1/2$. Applying Lemma 2.2.1 we know

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

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

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

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

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

Recall that the above inequality is derived under the condition that (3.5) holds. In case (3.5) does not hold, then

$$f(\boldsymbol{x}^0) - \underline{v}(H_S) > (\tilde{\tau}(T_S)/4)\,(v(H_S) - \underline{v}(H_S)) \geq \tau(H_S)\,(v(H_S) - \underline{v}(H_S))\,. \qquad (3.6)$$

By picking $\tilde{\boldsymbol{x}} = \arg\max\{f(\hat{\boldsymbol{x}}), f(\boldsymbol{x}^0)\}$, regardless whether (3.5) or (3.6) holds, we shall uniformly have $f(\tilde{\boldsymbol{x}}) - \underline{v}(H_S) \geq \tau(H_S)\,(v(H_S) - \underline{v}(H_S))$. □

Algorithm 3.2.1 can be modified and applied for more general case of (H_S), i.e., the even case of (M_S). Essentially, the extended link from multilinear form to mixed form (Lemma 2.3.3) is required, instead of the one from multilinear form to homogenous form (Lemma 2.2.1). The following approximation result can be proven by using a similar argument as of Theorem 3.2.3. The approximation ratio is same as that for $(M_{\bar{S}})$ in Theorem 2.3.4.

Theorem 3.2.4 *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 $\tau(M_S)$.*

3.3 Intersection of Co-centered Ellipsoids

In this section, we consider a generalization of the polynomial optimization models discussed in Chap. 2, to include ellipsoidal constraints. Specifically, the models include optimization of the four types of polynomial functions described in Sect. 1.3.1, with the constraint sets being extended to intersection of finite number of co-centered ellipsoids, i.e.,

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

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

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

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

where $\boldsymbol{Q}^k_{i_k} \succeq 0$ and $\sum_{i_k=1}^{m_k} \boldsymbol{Q}^k_{i_k} \succ 0$, and $\boldsymbol{Q}_i \succeq 0$ and $\sum_{i=1}^{m} \boldsymbol{Q}_i \succ 0$ whenever appropriate.

When $d=1$, all the models can be formulated as standard second order cone programs (SOCP), and therefore can be solved in polynomial time. However, they become NP-hard when $d \ge 2$. Nemirovski et al. [86] proposed a polynomial-time randomized approximation algorithm with approximation ratio $\Omega(1/\ln m)$ for (H_Q) when $d=2$, based on SDP relaxation and randomization techniques, and the method can also be applied to (T_Q), (M_Q), and (P_Q) as well. When $d=4$, Luo and Zhang [76] established the relationship of (H_Q) with its quadratic SDP relaxation, and proposed polynomial-time randomized approximation algorithm when the number of constraints being one, or essentially the model $(H_{\bar{S}})$.

We shall propose polynomial-time randomized approximation algorithms for these models with provable worst-case performance ratios, provided the degree of the objective polynomial is fixed. The extension from the Euclidean ball constraint to co-centered ellipsoidal constraint is not straightforward, as the complexity for the quadratic polynomial objective is already different, i.e., NP-hard for (P_Q) while polynomial-time solvable for $(P_{\bar{S}})$. In fact, the extension from multilinear form optimization (T_Q) to homogeneous form optimization (H_Q) and mixed-form optimization (M_Q) are quite similar as that of the Euclidean ball constraint, as well as the extension to general inhomogeneous polynomial optimization (P_Q). However, the decomposition routine, which plays an indispensable role in solving (T_Q), is quite technically involved.

3.3.1 Multilinear Form

Let us start by discussing the approximation algorithm for

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

where $\boldsymbol{Q}^k_{i_k} \succeq 0$ and $\sum_{i_k=1}^{m_k}\boldsymbol{Q}^k_{i_k} \succ 0$ for $k=1,2,\ldots,d$, $i_k=1,2,\ldots,m_k$. As before, we first investigate the base: $d=2$. Suppose $F(\boldsymbol{x}^1,\boldsymbol{x}^2)=(\boldsymbol{x}^1)^{\mathrm{T}}\boldsymbol{F}\boldsymbol{x}^2$. Denote $\boldsymbol{y}=\begin{pmatrix}\boldsymbol{x}^1\\ \boldsymbol{x}^2\end{pmatrix}$, $\boldsymbol{F}'=\begin{bmatrix}\boldsymbol{0}_{n_1\times n_1} & \boldsymbol{F}/2\\ \boldsymbol{F}^{\mathrm{T}}/2 & \boldsymbol{0}_{n_2\times n_2}\end{bmatrix}$, $\boldsymbol{Q}_i=\begin{bmatrix}\boldsymbol{Q}^1_i & \boldsymbol{0}_{n_1\times n_2}\\ \boldsymbol{0}_{n_2\times n_1} & \boldsymbol{0}_{n_2\times n_2}\end{bmatrix}$ 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

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

The above is a special case of (H_Q) for $d=2$: a standard quadratically constrained quadratic program (QCQP). Such problems can be solved approximately by a polynomial-time randomized algorithm with approximation ratio $\Omega\left(\frac{1}{\ln(m_1+m_2)}\right)$. Specifically, it is the following approximation result by Nemirovski et al. [86].

Theorem 3.3.1 *If $d=2$, (H_Q) admits a polynomial-time randomized approximation algorithm with approximation ratio $\Omega\left(\frac{1}{\ln m}\right)$.*

The underlying algorithm is based on the SDP relaxation and randomization as discussed in Sect. 1.4.4; for more details, one is referred to Nemirovski et al. [86] or He et al. [48].

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

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

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

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

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

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

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

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

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

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

Indeed, it is easy to verify that $\operatorname{tr}(\boldsymbol{Q}_i \boldsymbol{W} \boldsymbol{P}_j \boldsymbol{W}^{\mathrm{T}}) = (\operatorname{vec}(\boldsymbol{W}))^{\mathrm{T}} (\boldsymbol{Q}_i \otimes \boldsymbol{P}_j) \operatorname{vec}(\boldsymbol{W})$, which implies that $\operatorname{tr}(\boldsymbol{Q}_i \boldsymbol{W} \boldsymbol{P}_j \boldsymbol{W}^{\mathrm{T}}) \le 1$ 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

$$F(\hat{\boldsymbol{W}}, \hat{\boldsymbol{z}}) \ge \Omega\left(\frac{1}{\ln(m_1 m_2 + m_3)} \right) v(\check{T}_Q) \ge \Omega\left(\frac{1}{\ln m} \right) v(\hat{T}_Q), \tag{3.7}$$

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 $\hat{\boldsymbol{W}}$, and a new decomposition routine is needed. For this purpose, we first introduce the following lemma.

Lemma 3.3.2 *Suppose $\boldsymbol{Q}_i \in \mathbb{R}^{n \times n}$, $\boldsymbol{Q}_i \succeq 0$ for all $1 \le i \le m$, and $\sum_{i=1}^{m} \boldsymbol{Q}_i \succ 0$, the following SDP problem:*

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

has an optimal solution with optimal value equal to n.

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

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

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., $v(\mathrm{PS}) = v(\mathrm{DS})$. 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,

$$
v(\mathrm{PS}) = \sum_{i=1}^{m} t_i^* \ge \sum_{i=1}^{m} t_i^* \mathrm{tr}\,(\boldsymbol{U}^* \boldsymbol{Q}_i) \ge \mathrm{tr}\,(\boldsymbol{U}^* (\boldsymbol{U}^*)^{-1}) = n. \tag{3.8}
$$

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

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

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 $d_i Y'_{ii} - (Z'_{ii})^2 \ge 0$ for $i = 1, 2, \ldots, n$. Therefore,

$$
\begin{aligned}
-\mathrm{tr}\,(\boldsymbol{X}\boldsymbol{Y}) - 2\,\mathrm{tr}\,(\boldsymbol{Z}) &= -\mathrm{tr}\,(\boldsymbol{D}\boldsymbol{Y}') - 2\,\mathrm{tr}\,(\boldsymbol{Z}') = -\sum_{i=1}^{n} d_i Y'_{ii} - 2\sum_{i=1}^{n} Z'_{ii} \\
&\le -\sum_{i=1}^{n} (Z'_{ii})^2 - 2\sum_{i=1}^{n} Z'_{ii} \le -\sum_{i=1}^{n} (Z'_{ii} + 1)^2 + n \le n.
\end{aligned}
$$

This implies that $v(\mathrm{DS}) \le v(RS) \le n$. By combining this with (3.8), and noticing the strong duality relationship, it follows that $v(\mathrm{PS}) = v(\mathrm{DS}) = n$. □

We then have the following decomposition method, to be called DR 3.3.1, as a further extension of DR 2.1.1, 2.1.2, and 3.1.1.

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

1 *Solve the SDP problem*

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

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*

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

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

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

- *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 $O({n_1}^2)$ variables and $O(m_1)$ constraints. Moreover, it requires $O(n_2(n_1 m_1 + n_2 m_2)\ln(1/\varepsilon))$ other operations to get the quality assured solution with probability $1-\varepsilon$.

Lemma 3.3.3 *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* $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x} \le 1$ *for all* $1 \le i \le m_1$ *and* $\boldsymbol{y}^{\mathrm{T}}\boldsymbol{P}_j\boldsymbol{y} \le 1$ *for all* $1 \le j \le m_2$, *such that*

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

Proof. Following the randomization procedure in Step 3 of DR 3.3.1, by Lemma 3.3.2 we have for any $1 \le i \le m_1$ and $1 \le j \le m_2$,

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

So et al. [107] have established that if $\boldsymbol{\xi}$ is a normal random vector and $\boldsymbol{Q} \succeq 0$, then for any $\alpha > 0$,

$$\Pr\left\{\boldsymbol{\xi}^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{\xi} \ge \alpha \mathrm{E}[\boldsymbol{\xi}^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{\xi}]\right\} \le 2\mathrm{e}^{-\frac{\alpha}{2}}.$$

Applying this result we have

$$\Pr\left\{\boldsymbol{\xi}^{\mathrm{T}} \boldsymbol{Q}_i \boldsymbol{\xi} \ge \alpha_1\right\} \le \Pr\left\{\boldsymbol{\xi}^{\mathrm{T}} \boldsymbol{Q}_i \boldsymbol{\xi} \ge \alpha_1 \mathrm{E}[\boldsymbol{\xi}^{\mathrm{T}} \boldsymbol{Q}_i \boldsymbol{\xi}]\right\} \le 2\mathrm{e}^{-\frac{\alpha_1}{2}},$$
$$\Pr\left\{\boldsymbol{\eta}^{\mathrm{T}} \boldsymbol{P}_j \boldsymbol{\eta} \ge \alpha_2 n_1\right\} \le \Pr\left\{\boldsymbol{\eta}^{\mathrm{T}} \boldsymbol{P}_j \boldsymbol{\eta} \ge \alpha_2 \mathrm{E}[\boldsymbol{\eta}^{\mathrm{T}} \boldsymbol{P}_j \boldsymbol{\eta}]\right\} \le 2\mathrm{e}^{-\frac{\alpha_2}{2}}.$$

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

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

where $\alpha_1 := 2\ln(8m_1/\theta)$ and $\alpha_2 := 2\ln(8m_2/\theta)$. Since $\alpha_1\alpha_2 = O(\ln m_1 \ln m_2)$, 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 3.3.3, then in polynomial time we can find $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$, satisfying the constraints of $(\hat{T}_Q)$, such that

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

Combined with (3.7), we thus got a $\frac{1}{\sqrt{n_1}} \Omega\left(\frac{1}{\ln^2 m}\right)$-approximate solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ for the model (T_Q) when $d = 3$.

This result can be generalized to the model (T_Q) of any fixed degree d by induction, as the following theorem asserts, whose proof is omitted.

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

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

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

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 $n_1 \le n_2 \le \cdots \le n_d$, matrices $\boldsymbol{Q}^k_{i_k} \in \mathbb{R}^{n_k\times n_k}$, $\boldsymbol{Q}^k_{i_k} \succeq 0$ and $\sum_{i_k=1}^{m_k} \boldsymbol{Q}^k_{i_k} \succ 0$ for all $1 \le k \le d$ and $1 \le i_k \le m_k$.
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.,
      $F_{i_1,i_2,\cdots,i_d} = F'_{i_2,i_3,\cdots,i_{d-1},(i_1-1)n_d+i_d} \quad \forall\, 1 \le i_1 \le n_1, 1 \le i_2 \le n_2, \cdots, 1 \le i_d \le n_d.$
2 Compute matrices $\boldsymbol{P}_{i_1,i_d} = \boldsymbol{Q}^1_{i_1} \otimes \boldsymbol{Q}^d_{i_d}$ for all $1 \le i_1 \le m_1$ and $1 \le i_d \le m_d$.
3 For $(T_Q)$ with the $(d-1)$-th order tensor $\boldsymbol{F}'$, matrices $\boldsymbol{Q}^k_{i_k}$ $(2 \le k \le d-1, 1 \le i_k \le m_k)$ and $\boldsymbol{P}_{i_1,i_d}$ $(1 \le i_1 \le m_1, 1 \le i_d \le m_d)$: if $d-1=2$, then apply SDP relaxation and randomization procedure in Theorem 3.3.1 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 $(\boldsymbol{Q}^1_i, \boldsymbol{Q}^d_j, \boldsymbol{X}, \boldsymbol{M}') = (\boldsymbol{Q}_i, \boldsymbol{P}_j, \boldsymbol{W}, \boldsymbol{M})$ 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 $(\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^d)$.
```

3.3.2 *Homogeneous Form*

Next we proceed to homogenous polynomial optimization over intersection of co-centered ellipsoids

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

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

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

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

Theorem 3.3.4 asserts an approximate solution for $(\tilde{H}_Q)$, together with Lemma 2.2.1 we are led to the following approximation algorithm for solving (H_Q).

Algorithm 3.3.3

```
• INPUT: a d-th order supersymmetric 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 $1 \le i \le m$ with $\sum_{i=1}^m \boldsymbol{Q}_i \succ 0$.
1 Apply Algorithm 3.3.2 to solve the problem
      $\max\ F(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^d)$
      s.t. $(\boldsymbol{x}^k)^{\mathrm{T}}\boldsymbol{Q}_i\boldsymbol{x}^k \le 1,\ k=1,2,\ldots,d,\ i=1,2,\ldots,m,$
           $\boldsymbol{x}^k \in \mathbb{R}^n,\ k=1,2,\ldots,d$
  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 3.3.3 applies for both odd and even d for (H_Q), the approximation results are different, as the following theorems attest. Essentially, we can only speak of a relative approximation ratio when d is even, which is the same as its special case $(H_{\bar{S}})$.

Theorem 3.3.5 *When $d \ge 3$ is odd, (H_Q) admits a polynomial-time randomized approximation algorithm with approximation ratio $\tau(H_Q)$, where*

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

Theorem 3.3.6 *When $d \ge 4$ is even, (H_Q) admits a polynomial-time randomized approximation algorithm with relative approximation ratio $\tau(H_Q)$.*

The proofs of the above theorems are by-and-large similar to those of Theorems 2.2.2 and 2.2.4 for the Euclidean ball-constrained case, with one caveat being the feasibility of the solution generated in the second step of Algorithm 3.3.3. However, it can be dealt with by noticing that for any $1 \le k \le m$

$$\begin{aligned}\left(\sum_{i=1}^{d}\beta_i\hat{\boldsymbol{x}}^i\right)^{\mathrm{T}}\boldsymbol{Q}_k\left(\sum_{j=1}^{d}\beta_j\hat{\boldsymbol{x}}^j\right) &= \sum_{i,j=1}^{d}\beta_i(\hat{\boldsymbol{x}}^i)^{\mathrm{T}}\boldsymbol{Q}_k\beta_j\hat{\boldsymbol{x}}^j\\ &= \sum_{i,j=1}^{d}\left(\beta_i\boldsymbol{Q}_k{}^{\frac{1}{2}}\hat{\boldsymbol{x}}^i\right)^{\mathrm{T}}\left(\beta_j\boldsymbol{Q}_k{}^{\frac{1}{2}}\hat{\boldsymbol{x}}^j\right) \le \sum_{i,j=1}^{d}\left\|\beta_i\boldsymbol{Q}_k{}^{\frac{1}{2}}\hat{\boldsymbol{x}}^i\right\|\left\|\beta_j\boldsymbol{Q}_k{}^{\frac{1}{2}}\hat{\boldsymbol{x}}^j\right\|\\ &= \sum_{i,j=1}^{d}\sqrt{(\hat{\boldsymbol{x}}^i)^{\mathrm{T}}\boldsymbol{Q}_k\hat{\boldsymbol{x}}^i}\sqrt{(\hat{\boldsymbol{x}}^j)^{\mathrm{T}}\boldsymbol{Q}_k\hat{\boldsymbol{x}}^j} \le \sum_{i,j=1}^{d}1\cdot 1 = d^2. \qquad (3.9)\end{aligned}$$

3.3.3 Mixed Form

For the general mixed-form optimization over intersection of co-centered ellipsoids

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

where $\boldsymbol{Q}^k_{i_k}\succeq 0$ and $\sum_{i_k=1}^{m_k}\boldsymbol{Q}^k_{i_k}\succ 0$ for $k=1,2,\ldots,s$, $i_k=1,2,\ldots,m_k$, we have the following results (cf. $(M_{\bar{S}})$ in Sect. 2.3).

Theorem 3.3.7 *If $d\ge 3$ 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*

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

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

The proof of Theorem 3.3.7 is quite similar to that of Theorem 2.3.2, 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 approximation ratio $\tilde{\tau}(M_S)\Omega\left(\ln^{-(d-1)}m\right)$. Then, Lemma 2.3.3 servers as a bridge from that solution to a feasible solution for (M_Q). Specifically, we may adjust the solution of (T_Q) step by step. In each step, we apply Lemma 2.2.1 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 (3.9).

Theorem 3.3.8 *If $d \geq 4$ and all $d_k\,(k=1,2,\dots,s)$ are even, then (M_Q) admits a polynomial-time randomized approximation algorithm with relative approximation ratio $\tau(M_Q)$.*

The proof is analogous to that of Theorems 2.2.4, 2.3.4, and 3.3.6, which is left to the interested reader.

3.3.4 Inhomogeneous Polynomial

Finally in this section, we consider general polynomial optimization over quadratic constraints, namely

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

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

We remark here that (P_Q) includes as a special case the optimization of a general polynomial function over a central-symmetric polytope:

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

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

As general extensions of $(P_{\bar{S}})$ and (H_Q), (P_Q) will at most only admit a polynomial-time approximation algorithm with a relative approximation ratio unless $P=NP$. The main algorithm and the approximation result are as follows.

Algorithm 3.3.4

- *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 (2.7), with $\boldsymbol{F}$ being an $(n+1)$-dimensional d-th order supersymmetric tensor.*

2 *Apply Algorithm 3.3.2 to solve the problem*

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

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

Theorem 3.3.9 (P_Q) *admits a polynomial-time randomized approximation algorithm with relative approximation ratio* $\tau(P_Q)$, *where*

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

Our scheme for solving general polynomial optimization model (P_Q) is similar to that for solving $(P_{\bar{S}})$ in Sect. 2.4. 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 3.3.9 is similar to that of Theorem 2.4.2. 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

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

which can be relaxed to the inhomogeneous multilinear form optimization model

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

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

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

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 $(n+1)^{-\frac{d-2}{2}}\Omega\left(\ln^{-(d-1)} m\right)$, 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 3.3.4 satisfies

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

Noticing that $(y_h^k)^2 \le (\bar{\boldsymbol{y}}^k)^{\mathrm{T}}\bar{\boldsymbol{Q}}_1\bar{\boldsymbol{y}}^k \le 1$ for $k=1,2,\ldots,d$, we again apply Lemma 2.4.4 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 2.4.5.

Let $c = \max_{\boldsymbol{\beta}\in\mathbb{B}^d,\prod_{k=1}^d \beta_k=-1}\Pr\{\boldsymbol{\eta}=\boldsymbol{\beta}\}$, 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 $k=1,2,\ldots,d$. Similar as in (2.12), we are able to find a binary vector $\boldsymbol{\beta}'\in\mathbb{B}^d$, with

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

This proves the following theorem as a byproduct.

Theorem 3.3.10 *(TP_Q) admits a polynomial-time randomized approximation algorithm with approximation ratio $2^{-\frac{3d}{2}}(n+1)^{-\frac{d-2}{2}}\Omega\left(\ln^{-(d-1)} m\right)$.*

To prove the main theorem in this section (Theorem 3.3.9), we only need to check the feasibility of $\boldsymbol{z}$ generated by Algorithm 3.3.4, while the worst-case performance ratio can be proven by the similar argument in Sect. 2.4.4. Indeed, $(\bar{\boldsymbol{z}}^1,\bar{\boldsymbol{z}}^2,\cdots,\bar{\boldsymbol{z}}^d)$ at

Step 3 of Algorithm 3.3.4 satisfies

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

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 $2 \le |z_h(\beta)| \le 2d$. Noticing by the Cauchy–Schwartz inequality,

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

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 that $\boldsymbol{z}$ is also feasible.

3.4 Convex Compact Set

In this section, we study polynomial optimization model in a very general framework

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

where $G \subset \mathbb{R}^n$ is a given convex compact set. (P_G) actually includes many of the precious polynomial optimization models as its subclasses, including polynomial optimization over the Euclidean ball discussed in Chap. 2, over the hypercube discussed in Sect. 3.1, and over intersection of co-centered ellipsoids discussed in Sect. 3.3.

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, 85]), which is the following.

Theorem 3.4.1 *Given a convex compact set $G \subset \mathbb{R}^n$ with nonempty 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 $\{\boldsymbol{B}\boldsymbol{x}/\sqrt{n} + \boldsymbol{b} \,|\, \boldsymbol{x} \in \bar{\mathbb{S}}^n\} \subset G$.*

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

Algorithm 3.4.1

```
• 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 $\operatorname{rank}(\boldsymbol{A}) = m \le n$, such that two co-centered ellipsoids $E_1 = \{\boldsymbol{Au}+\boldsymbol{b} \,|\, \boldsymbol{u} \in \bar{\mathbb{S}}^m\}$ and $E_2 = \{t\boldsymbol{Au}+\boldsymbol{b} \,|\, \boldsymbol{u} \in \bar{\mathbb{S}}^m\}$ satisfy $E_1 \subset G \subset E_2$.
2 Compute a polynomial function $p_0(\boldsymbol{u}) = p(\boldsymbol{Au}+\boldsymbol{b})$ of variable $\boldsymbol{u} \in \mathbb{R}^m$.
3 Apply Algorithm 2.4.1 with input $p_0(\boldsymbol{x})$ and output $\boldsymbol{y} \in \bar{\mathbb{S}}^m$.
4 Compute $\boldsymbol{z} = \boldsymbol{Ay}+\boldsymbol{b}$.
• OUTPUT: a feasible solution $\boldsymbol{z} \in G$.
```

The key result in this section is the following theorem.

Theorem 3.4.2 *If $\bar{\mathbb{S}}^n \subset G \subset t\bar{\mathbb{S}}^n := \{\boldsymbol{x} \in \mathbb{R}^n \,|\, \|\boldsymbol{x}\| \le t\}$ for some $t \ge 1$, 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 supersymmetric, then $(\bar{P}_G)$ can be relaxed to the inhomogeneous multilinear form problem

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

Recall that in Sect. 2.4.2, we have defined

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

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, $(TP_{\bar{S}}(\sqrt{t^2+1}))$ is a relaxation of (TP_G), and $v(TP_{\bar{S}}(\sqrt{t^2+1})) \ge v(TP_G) \ge v(\bar{P}_G) = v(P_G)$. The rest of the proof follows similarly as that in Sect. 2.4.4. 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 $\tau(P_G)(t)$. $\square$

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 3.4.3 *Given a bounded set $G \subset \mathbb{R}^n$, if two co-centered ellipsoids $E_1 = \{A\boldsymbol{u}+\boldsymbol{b} \mid \boldsymbol{u} \in \bar{\mathbb{S}}^n\}$ and $E_2 = \{tA\boldsymbol{u}+\boldsymbol{b} \mid \boldsymbol{u} \in \bar{\mathbb{S}}^n\}$ 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 3.4.2 and Corollary 3.4.3 does not need to be convex, as long as the two required ellipsoids are in place. However, the Löwner–John theorem guarantees the existence of such inner and outer ellipsoids required in Corollary 3.4.3 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, 85]:

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

By Corollary 3.4.3 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 $(P_Q): \max_{\boldsymbol{x}^{\mathrm{T}}Q_i\boldsymbol{x} \le 1, i=1,2,\ldots,m} p(\boldsymbol{x})$ discussed in Sect. 3.3.4, may in principle be solved by directly applying Corollary 3.4.3 as well. If we adopt that approach (Algorithm 3.4.1), 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 of the two, the quality approximation ratio in Theorem 3.3.9 can be improved to $\Omega\left(\max\left\{n^{-\frac{d-2}{2}} \ln^{-(d-1)} m,\, n^{-\frac{2d-2}{2}}\right\}\right)$.

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 $\mathscr{E}_{m,n}$ to be the collection of all such structures, or more specifically

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

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

$$t(G;m,n) := \inf \{ t \,|\, E_{m,n} \in \mathscr{E}_{m,n} \text{ such that } \boldsymbol{b} + E_{m,n} \subset G \subset \boldsymbol{b} + tE_{m,n} \},$$
$$\theta(m,n) := \sup \{ t(G;m,n) \,|\, G \subset \mathbb{R}^n \text{ is convex compact and central-symmetric} \}.$$

The 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 3.3.9 suggests a polynomial-time randomized approximation algorithm of (P_G) with relative approximation ratio $(\theta(m,n))^{-d}\tau(P_Q) = \Omega\left((\theta(m,n))^{-d} n^{-\frac{d-2}{2}} \ln^{-(d-1)} m \right)$.

Finally, to conclude this section, we remark that homogeneous polynomial (including multilinear form, homogeneous form, and mixed form) optimization over a general convex compact set also admits a polynomial-time approximation algorithm with worst-case performance ratio, by resorting to the Löwner–John theorem.

3.5 Mixture of Binary Hypercube and the Euclidean Sphere

Our last part of the approximation methods brings most of the results in previous works together, to yield solution methods for solving 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 Euclidean spherical constraint and the binary constraint. In particular, the models considered are:

$$\boxed{\begin{array}{lll} (T_{BS}) & \max & F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d, \boldsymbol{y}^1, \boldsymbol{y}^2, \ldots, \boldsymbol{y}^{d'}) \\ & \text{s.t.} & \boldsymbol{x}^k \in \mathbb{B}^{n_k}, k = 1,2,\ldots,d, \\ & & \boldsymbol{y}^\ell \in \mathbb{S}^{m_\ell}, \ell = 1,2,\ldots,d' \end{array}}$$

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

$$\boxed{\begin{array}{ll}(M_{BS}) \ \max & f(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^s,\boldsymbol{y}^1,\boldsymbol{y}^2,\cdots,\boldsymbol{y}^t)\\ \quad\text{s.t.} & \boldsymbol{x}^k\in\mathbb{B}^{n_k},\ k=1,2,\ldots,s,\\ & \boldsymbol{y}^\ell\in\mathbb{S}^{m_\ell},\ \ell=1,2,\ldots,t\end{array}}$$

The model (M_{BS}) is a generalization of the models (T_{BS}) and (H_{BS}). In fact, it can also be taken as a generalization of most of the homogenous polynomial optimization models discussed in previous sections, namely (T_B), (H_B), and (M_B) of Sect. 3.1, and (T_S), (H_S), and (M_S) of Sect. 3.2 as well.

These mixed models have versatile applications, e.g., matrix combinatorial problem, and the vector-valued max-cut problem; we will discuss the details in Sect. 4.4. 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, including the simplest constraint set of one Euclidean sphere and one binary hypercube, 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 quadratic form over binary hypercube, 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 Sect. 2.1, and two types of decomposition routines are called, one for decomposition of the Euclidean spherical (ball) constraint 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 form relaxation method is again applied. As the key lemmas (Lemma 2.2.1 and 2.3.3) are still available, we are able to design approximation algorithms under some mild square-free conditions. For the easy of reading, in this section 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, $d+d'$, as a constant.

3.5.1 Multilinear Form

Our first mixed model is to maximize a multilinear form over binary hypercube and the Euclidean sphere, namely,

$$\boxed{\begin{array}{ll}(T_{BS}) \ \max & F(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^d,\boldsymbol{y}^1,\boldsymbol{y}^2,\ldots,\boldsymbol{y}^{d'})\\ \quad\text{s.t.} & \boldsymbol{x}^k\in\mathbb{B}^{n_k},\ k=1,2,\ldots,d,\\ & \boldsymbol{y}^\ell\in\mathbb{S}^{m_\ell},\ \ell=1,2,\ldots,d'\end{array}}$$

where $n_1 \le n_2 \le \cdots \le n_d$ and $m_1 \le m_2 \le \cdots \le m_{d'}$. The simplest case of (T_{BS}), $d = d' = 1$, is worth mention, as it plays an essential role in the whole section. Based on this case, we shall derive polynomial-time approximation algorithm with worst-case performance ratio for (T_{BS}) with any fixed degree $d + d'$. Towards this end, let us first introduce an approximation result of Nesterov [87] concerning quadratic maximization over the binary hypercube—a special case of (H_B) for $d = 2$.

Theorem 3.5.1 *If $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is positive semidefinite, then the problem $\max_{\boldsymbol{x}\in\mathbb{B}^n} \boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}$ admits a polynomial-time randomized approximation algorithm with approximation ratio $2/\pi$.*

This result generalizes that of Goemans and Williamson [39], where matrix $\boldsymbol{A}$ is the Laplacian of a given graph. The proof of Theorem 3.5.1 is based on SDP relaxation and the randomization method like the one we presented in Sect. 1.4.4, which can be found in [87]. Let us now present the method for solving (T_{BS}) when $d = d' = 1$.

Proposition 3.5.2 *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

$$(\hat{T}_{BS}) \ \max \ \boldsymbol{x}^{\mathrm{T}}\boldsymbol{F}\boldsymbol{y} \quad \text{s.t.} \quad \boldsymbol{x} \in \mathbb{B}^{n_1}, \boldsymbol{y} \in \mathbb{S}^{m_1}.$$

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

$$\max \ \boldsymbol{x}^{\mathrm{T}}\boldsymbol{F}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{x} \quad \text{s.t.} \quad \boldsymbol{x} \in \mathbb{B}^{n_1}.$$

Noticing that matrix $\boldsymbol{F}\boldsymbol{F}^{\mathrm{T}}$ is positive semidefinite, the above problem includes the max-cut problem (see, e.g., [39]) as a special case. Therefore it is NP-hard. According to Theorem 3.5.2, it admits a polynomial-time randomized approximation algorithm 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 3.5.2 serves as the base for solving (T_{BS}) recursively, for the case of general degrees d and d'. The induction with regard to the discrete variables and the continuous variables is conducted separately. For the recursion on d with discrete variables $\boldsymbol{x}^k$ $(k = 1,2,\ldots,d)$, DR 3.1.1 is applied in each recursive step; for the recursion on d' with continuous variables $\boldsymbol{y}^\ell$ $(\ell = 1,2,\ldots,d')$, two decomposition routines in Sect. 2.1 are used: the eigenvalue decomposition approach DR 2.1.2 and the randomized decomposition approach DR 2.1.1. The main result in this section is the following.

Theorem 3.5.3 (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 3.5.2 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 2.1.5, (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 2.1.1 (used in DR 2.1.2), $\max_{\boldsymbol{y}^1 \in \mathbb{S}^{m_1}, \boldsymbol{y}^{d'} \in \mathbb{S}^{m_{d'}}} (\boldsymbol{y}^1)^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{y}^{d'}$ can be solved in polynomia 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 conclude 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.,

$$\begin{aligned} \max\ & F(\boldsymbol{X},\boldsymbol{x}^2,\boldsymbol{x}^3,\cdots,\boldsymbol{x}^{d-1},\boldsymbol{y}^1,\boldsymbol{y}^2,\cdots,\boldsymbol{y}^{d'}) \\ \text{s.t.}\ & \boldsymbol{X} \in \mathbb{B}^{n_1 n_d}, \boldsymbol{x}^k \in \mathbb{B}^{n_k}, k=2,3,\ldots,d-1, \\ & \boldsymbol{y}^\ell \in \mathbb{S}^{m_\ell}, \ell = 1,2,\ldots,d'. \end{aligned}$$

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 3.1.1, which will further deteriorate by an additional factor of $\frac{2}{\pi\sqrt{n_1}}$ in expectation, as shown in (3.1). Combining these two factors together, we are led to the ratio $\tau(T_{BS})$. □

The algorithm for solving (T_{BS}) is summarized below.

Algorithm 3.5.1

- *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 3.5.2, 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 2.1.3, 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 3.1.2, with input $\boldsymbol{F}''$ and output $(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\cdots,\hat{\boldsymbol{x}}^d)$.*

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

3.5.2 Homogeneous Form

We further extend the mixed model in previous section to more general case of homogeneous form, namely,

$$\boxed{\begin{aligned} (H_{BS})\ \max\ & f(\boldsymbol{x},\boldsymbol{y}) \\ \text{s.t.}\ & \boldsymbol{x} \in \mathbb{B}^n, \boldsymbol{y} \in \mathbb{S}^m \end{aligned}}$$

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 Sect. 2.2 and the model (H_B) in Sect. 3.1.2. The key ingredient underlying the approximation method is again the multilinear form relaxation (T_{BS}), which admits a polynomial-time randomized approximation algorithm according to Theorem 3.5.3. Then by applying Lemma 2.2.1 as a linkage, together with the square-free property for the discrete variables $\boldsymbol{x}$, we are finally led to the following conclusion regarding (H_{BS}).

Theorem 3.5.4 *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 3.1.4, 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 $1 \le k \le d$ and $\hat{\boldsymbol{y}}^\ell \in \mathbb{S}^m$ for all $1 \le \ell \le d'$ in polynomial time, such that

$$F(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\cdots,\hat{\boldsymbol{x}}^d,\hat{\boldsymbol{y}}^1,\hat{\boldsymbol{y}}^2,\cdots,\hat{\boldsymbol{y}}^{d'}) \ge (2/\pi)^{\frac{2d-1}{2}} n^{-\frac{d-1}{2}} m^{-\frac{d'-1}{2}} v(H_{BS}).$$

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 2.3.3 (or Lemma 2.2.1 twice), we have

$$\mathrm{E}\left[\prod_{i=1}^d \xi_i \prod_{j=1}^{d'} \eta_j\, f\left(\sum_{k=1}^d \xi_k \hat{\boldsymbol{x}}^k, \sum_{\ell=1}^{d'} \eta_\ell \hat{\boldsymbol{y}}^\ell\right)\right] = d!d'!F(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\cdots,\hat{\boldsymbol{x}}^d,\hat{\boldsymbol{y}}^1,\hat{\boldsymbol{y}}^2,\cdots,\hat{\boldsymbol{y}}^{d'}). \tag{3.10}$$

Thus we are able to find binary vectors $\boldsymbol{\beta} \in \mathbb{B}^d$ and $\boldsymbol{\beta}' \in \mathbb{B}^{d'}$, such that

$$\prod_{i=1}^d \beta_i \prod_{j=1}^{d'} \beta'_j\, f\left(\sum_{k=1}^d \beta_k \hat{\boldsymbol{x}}^k, \sum_{\ell=1}^{d'} \beta'_\ell \hat{\boldsymbol{y}}^\ell\right) \ge d!d'!F(\hat{\boldsymbol{x}}^1,\hat{\boldsymbol{x}}^2,\cdots,\hat{\boldsymbol{x}}^d,\hat{\boldsymbol{y}}^1,\hat{\boldsymbol{y}}^2,\cdots,\hat{\boldsymbol{y}}^{d'}).$$

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 $\|\hat{\boldsymbol{y}}\| \le d'$ and combining the previous two inequalities, it follows that

$$f\left(\frac{\hat{\boldsymbol{x}}}{d}, \frac{\hat{\boldsymbol{y}}}{\|\hat{\boldsymbol{y}}\|}\right) \ge d^{-d} d'^{-d'} \prod_{i=1}^{d} \beta_i \prod_{j=1}^{d'} \beta'_j f\left(\sum_{k=1}^{d} \beta_k \hat{\boldsymbol{x}}^k, \sum_{\ell=1}^{d'} \beta'_\ell \hat{\boldsymbol{y}}^\ell\right) \ge \tau(H_{BS})\, v(H_{BS}).$$

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 for (3.2), and $f(\boldsymbol{x}, \tilde{\boldsymbol{y}})$ is square-free in $\boldsymbol{x}$, by applying Lemma 3.1.3, $\tilde{\boldsymbol{x}} \in \mathbb{B}^n$ can be found in polynomial time, such that

$$f(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}) \ge f(\hat{\boldsymbol{x}}/d, \tilde{\boldsymbol{y}}) \ge \tau(H_{BS})\, v(H_{BS}).$$

$\square$

We remark that in Theorem 3.5.4, if $d' = 2$ and d is odd, then the factor $d'!\, d'^{-d'}$ in $\tau(H_{BS})$ can be removed for the same reason as we argued in the proof of Theorem 2.3.2 (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 shall present the approximation result for the even degree case. The idea of the proof is quit similar to that of Theorem 2.2.4 and 3.1.5 and the details are thus omitted.

Theorem 3.5.5 *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 $\tau(H_{BS})$.*

3.5.3 Mixed Form

Finally, we shall bring together the models discussed in previous sections, and discuss a very general model, which includes (T_S), (H_S), (M_S), (T_B), (H_B), (M_B), (T_{BS}), and (H_{BS}) all as its special cases. The model is to maximize a mixed form over variables in binary hypercube, mixed with variables in the Euclidean sphere, i.e.,

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

where associated with 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, $n_1 \le n_2 \le \cdots \le n_s$ and $m_1 \le m_2 \le \cdots \le m_t$, and $d = d_1 + d_2 + \cdots + d_s$ and $d' = d'_1 + d'_2 + \cdots + d'_t$ are deemed as fixed constants.

In order to derive polynomial-time approximation algorithms for this general model, we relax (M_{BS}) to the multilinear form optimization (T_{BS}), and solve it approximately using Algorithm 3.5.1, and adjust its solution step by step using

Lemma 2.2.1 or 2.3.3 as a linkage, and further adjust each discrete variables $\boldsymbol{x}^k$ using Lemma 3.1.3, leading to the following general results in two settings.

Theorem 3.5.6 *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*

$$\hat{\tau}(M_{BS}) := \left(\frac{2}{\pi}\right)^{\frac{2d-1}{2}} \left(\prod_{k=1}^{s}\frac{d_k!}{{d_k}^{d_k}} \prod_{1\le\ell\le t,\,3\le d'_\ell}\frac{d'_\ell!}{{d'_\ell}^{d'_\ell}}\right)\left(\frac{\prod_{k=1}^{s}{n_k}^{d_k}\prod_{\ell=1}^{t}{m_\ell}^{d'_\ell}}{n_s\,m_t}\right)^{-\frac{1}{2}}$$
$$= \Omega\left(\left(\frac{\prod_{k=1}^{s}{n_k}^{d_k}\prod_{\ell=1}^{t}{m_\ell}^{d'_\ell}}{n_s\,m_t}\right)^{-\frac{1}{2}}\right).$$

Theorem 3.5.7 *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*

$$\tau(M_{BS}) := \left(\frac{2}{\pi}\right)^{\frac{2d-1}{2}} \left(\prod_{k=1}^{s}\frac{d_k!}{{d_k}^{d_k}} \prod_{\ell=1}^{t}\frac{d'_\ell!}{{d'_\ell}^{d'_\ell}}\right)\left(\frac{\prod_{k=1}^{s}{n_k}^{d_k}\prod_{\ell=1}^{t}{m_\ell}^{d'_\ell}}{n_s\,m_t}\right)^{-\frac{1}{2}}$$
$$= \Omega\left(\left(\frac{\prod_{k=1}^{s}{n_k}^{d_k}\prod_{\ell=1}^{t}{m_\ell}^{d'_\ell}}{n_s\,m_t}\right)^{-\frac{1}{2}}\right).$$

To conclude this section as well as this chapter, we remark here that the approximation methods presented in this section can be actually extended to homogeneous polynomial optimization model with other mixed constraints, e.g., to maximize a multilinear form over binary hypercube and general convex compact set, among many others.

Chapter 4
Applications

The study of polynomial optimization models is rooted in various problems in scientific computation and other engineering applications. To illustrate some typical applications of the models studied in Chaps. 2 and 3, in this chapter we present some concrete examples in four categories: homogeneous polynomial over the Euclidean sphere; polynomial optimization over a general set; discrete polynomial optimization; and mixed integer programming. We shall note that the examples are selected to serve the purpose of illustration only; many more other interesting examples can be found in the literature. There is in fact an on-going effort to apply polynomial optimization models to science and engineering, management and computation, health care and data driven knowledge discovery, to name a few examples.

4.1 Homogeneous Polynomial Optimization Over the Euclidean Sphere

Polynomial optimization with spherical constraint has found many applications, including biomedical engineering (cf. [13, 38]), numerical linear algebra (cf. [90, 99, 100]), quantum mechanics (cf. [26, 40]). In this section we shall present four examples for the application of homogenous polynomial optimization over the Euclidean sphere.

4.1.1 Singular Values of Trilinear Forms

Trilinear forms play an important role in, e.g., 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 [17]). 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:

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

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 [17] showed in the binary case that $\|\boldsymbol{F}\|_S{}^2$ is 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. [27].

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 confining the field $\mathbb{K}$ to be the real domain, the problem of computing the largest singular value $\|\boldsymbol{F}\|_S$ is exactly $(T_{\bar{S}})$ when $d = 3$, or equivalently, (T_S) when $d = 3$. This is because the optimal value $\|\boldsymbol{F}\|_S$ is always attainable, and therefore "sup" can be replaced by "max." Moreover, 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 $|F(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})|$ is irrelevant. According to Proposition 2.1.3, the problem of computing $\|\boldsymbol{F}\|_S$ is NP-hard already in this real case. Together with Theorem 2.1.4, the spectrum norm of a trilinear form can be approximated in polynomial time within a factor of $\frac{1}{\sqrt{\min\{n_1,n_2,n_3\}}}$.

4.1.2 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, and graph analysis. For more information, one is referred to an excellent survey by Kolda and Bader [65]. The earliest development of tensor decomposition dates back to 1927, where Hitchcock [52, 53] put forward the polyadic form of a tensor. In modern days, tensor decomposition is often discussed in the form of canonical decomposition (CANDECOMP) by Carroll and Chang [22] and parallel factors (PARAFAC) by Harshman [43], or in short, the 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., [64]). 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, e.g., noises and errors in the data. Instead, one is naturally led to the following CP decomposition, which is in essence a least square formulation of the problem:

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

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

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

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 then have

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

Therefore, (TA) is equivalent to

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

which is exactly (T_S) discussed in Sect. 3.2. Remark that similar deductions can also be found in [28, 64, 119].

4.1.3 Eigenvalues and Approximation of Tensors

The notion of eigenvalues for a matrix has been naturally extended to higher order tensors; see, e.g., [71, 99]. As it turns out, for general tensor forms, the notion of eigenvalues becomes drastically more complex than its matrix counterpart. Qi [99] proposed several definitions of tensor eigenvalues, among which the most popular and straightforward one is named the *Z-eigenvalues*. For a given d-th order supersymmetric tensor $\boldsymbol{F} \in \mathbb{R}^{n^d}$, its Z-eigenvalue $\lambda \in \mathbb{R}$ with its corresponding Z-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 supersymmetric 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 supersymmetric tensor $\boldsymbol{F}$ is an important indicator of the 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 Z-eigenvector for the tensor $\boldsymbol{F}$, no matter whether d is even or odd. By Theorem 2.2.2, the largest Z-eigenvalue of an odd order supersymmetric 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 understood in the relative sense (see Theorem 3.2.3). 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 Sect. 4.1.2 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 supersymmetric, then the corresponding best rank-one approximation should be

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

Applying the same technique discussed in Sect. 4.1.2, we can equivalently reformulate the above problem as

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

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 $\lambda = F(\underbrace{\hat{\boldsymbol{x}},\hat{\boldsymbol{x}},\ldots,\hat{\boldsymbol{x}}}_{d})$, then the best rank-one approximation of the supersymmetric tensor $\boldsymbol{F}$ is $\lambda\,\underbrace{\hat{\boldsymbol{x}} \otimes \hat{\boldsymbol{x}} \otimes \cdots \otimes \hat{\boldsymbol{x}}}_{d}$.

4.1.4 Density Approximation in Quantum Physics

An interesting problem in physics is to precisely characterize the entanglement in a quantum system. The quantum entanglement describes an intensive interplay between subsystems of the full quantum system that go far beyond the statistical correlations found in a classical composite system. One existing mathematical description of quantum entanglement uses the following formulation, which is proposed by Dahl et al. [26].

Denote Δ_+^n to be the set of all $n \times n$ positive semedefinite matrices with trace being 1, i.e., $\Delta_+^n := \{\boldsymbol{A} \in \mathbb{R}^{n\times n} \,|\, \boldsymbol{A} \succeq 0, \mathrm{tr}\,(\boldsymbol{A}) = 1\}$ (aka the matrix simplex). Using the matrix decomposition method (see, e.g., Sturm and Zhang [110]), it is not hard to verify that the extreme points of Δ_+^n are all rank-one matrices, or specifically, $\Delta_+^n = \mathrm{conv}\,\{\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}} \,|\, \boldsymbol{x} \in \mathbb{S}^n\}$. 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

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

This projection problem is in general NP-hard. 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 [26]), i.e.,

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

Then 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

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

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 precisely (M_S) with $d = 4$ and $d_1 = d_2 = 2$ in Sect. 3.2.

4.2 Inhomogeneous Polynomial Optimization Over a General Set

The model of polynomial optimization over a general convex compact set has versatile applications, due primarily to its generic form. To better appreciate this new framework as well as the approximation algorithms that we have previously proposed, in this section we shall discuss a few specific examples from real applications, and show that they are readily formulated by the inhomogeneous polynomial optimization model.

4.2.1 Portfolio Selection with Higher Moments

The portfolio selection problem dates back to the early 1950s, when the seminal work of mean–variance model was proposed by Markowitz [80]. 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:

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

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean vector and covariance matrix of n given assets, respectively, and $\boldsymbol{e}$ is the all one vector. This model and its variations have been studied 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 [77] 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 [56], Kleniati et al. [60], and the references therein). In particular, a very general model in [60] is

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

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 framework of our model (P_G) in Sect. 3.4, as the constraint set is convex and compact. By directly applying Corollary 3.4.3 and the discussion on its applicability in a polytope, it admits a polynomial-time approximation algorithm with relative approximation ratio $\Omega\left(n^{-5}\right)$.

4.2.2 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 [76] proposed a least square formulation to this sensor network localization problem. Specifically, the problem takes the form of

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

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 $\mathscr{E}_{n,3n}$. According to the discussion at the end of Sect. 3.4, and noticing in this case $t(G;n,3n) \le 3$ is a constant, (SNL) admits a polynomial-time randomized approximation algorithm with relative approximation ratio $\Omega\left(\frac{1}{n\ln^3 n}\right)$.

4.3 Discrete Polynomial Optimization

Given the generic nature of the discrete polynomial optimization model, its applicability is 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. We shall present three problems in this section and show that they are readily formulated by the discrete polynomial optimization models.

4.3.1 The 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., [3, 35]). 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 [58]. 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 3.1.2, 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 4.3.1 *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

$$\begin{aligned}
F(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^d) &= \sum_{1\le i_k\le n_k,k=1,2,\ldots,d} F_{i_1 i_2\cdots i_d} x^1_{i_1} x^2_{i_2}\cdots x^d_{i_d} \\
&= \sum_{\boldsymbol{\beta}\in\mathbb{B}^d} \sum_{i_k\in\{j|x^k_j=\beta_k,1\le j\le n_k\},k=1,2,\ldots,d} F_{i_1 i_2\cdots i_d} x^1_{i_1} x^2_{i_2}\cdots x^d_{i_d} \\
&= \sum_{\boldsymbol{\beta}\in\mathbb{B}^d} \left(\prod_{1\le k\le d} \beta_k \sum_{i_k\in\{j|x^k_j=\beta_k,1\le j\le n_k\},k=1,2,\ldots,d} F_{i_1 i_2\cdots i_d} \right) \\
&\le \sum_{\boldsymbol{\beta}\in\mathbb{B}^d} \left| \sum_{i_k\in\{j|x^k_j=\beta_k,1\le j\le n_k\},k=1,2,\ldots,d} F_{i_1 i_2\cdots i_d} \right| \\
&\le \sum_{\boldsymbol{\beta}\in\mathbb{B}^d} \|\boldsymbol{F}\|_C = 2^d \|\boldsymbol{F}\|_C,
\end{aligned}$$

which implies $\|\boldsymbol{F}\|_{\infty\mapsto 1} \le 2^d \|\boldsymbol{F}\|_C$.

It is easy to observe that $\|\boldsymbol{F}\|_C = \max_{\boldsymbol{z}^k\in\{0,1\}^{n_k},k=1,2,\ldots,d} |F(\boldsymbol{z}^1,\boldsymbol{z}^2,\ldots,\boldsymbol{z}^d)|$. 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

$$\begin{aligned}
F(\boldsymbol{z}^1,\boldsymbol{z}^2,\ldots,\boldsymbol{z}^d) &= F\left(\frac{\boldsymbol{e}+\boldsymbol{x}^1}{2},\frac{\boldsymbol{e}+\boldsymbol{x}^2}{2},\ldots,\frac{\boldsymbol{e}+\boldsymbol{x}^d}{2}\right) \\
&= \frac{F(\boldsymbol{e},\boldsymbol{e},\ldots,\boldsymbol{e}) + F(\boldsymbol{x}^1,\boldsymbol{e},\ldots,\boldsymbol{e}) + \cdots + F(\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^d)}{2^d} \\
&\le \frac{1}{2^d}\cdot \|\boldsymbol{F}\|_{\infty\mapsto 1}\cdot 2^d = \|\boldsymbol{F}\|_{\infty\mapsto 1},
\end{aligned}$$

which implies $\|\boldsymbol{F}\|_C \le \|\boldsymbol{F}\|_{\infty\mapsto 1}$. □

4.3.2 Maximum Complete Satisfiability

The usual maximum satisfiability problem (see, e.g., [37]) 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 $(1 \le k \le m)$ has the following form:

$$\{x_{k_1}, x_{k_2}, \dots, x_{k_{s_k}}, \bar{x}_{k'_1}, \bar{x}_{k'_2}, \dots, \bar{x}_{k'_{t_k}}\},$$

where $s_k + t_k \le d$, associated with a weight $w_k \ge 0$ for $k = 1, 2, \dots, m$. 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 3.1.9 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.

4.3.3 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., [2,15]), the market split problem [25], as well as other 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 $\boldsymbol{0} \le \boldsymbol{x} \le \boldsymbol{u}$, such that $\boldsymbol{Ax} = \boldsymbol{b}$. The problem can be formulated by the least square method as

$$\begin{array}{lll} (\text{DE}) & \max & -(\boldsymbol{Ax} - \boldsymbol{b})^{\mathrm{T}}(\boldsymbol{Ax} - \boldsymbol{b}) \\ & \text{s.t.} & \boldsymbol{x} \in \mathbb{Z}^n, \boldsymbol{0} \le \boldsymbol{x} \le \boldsymbol{u}. \end{array}$$

According to the discussion at the end of Sect. 3.1.4, 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 3.1.9, (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 3.1.9, this problem admits a polynomial-time randomized approximation algorithm with relative approximation ratio $\Omega\left(\left(\sum_{i=1}^{n} \log_2 u_i\right)^{-(d-1)} \right)$.

4.4 Mixed Integer Programming

The generality of the mixed integer polynomial optimization studied gives rises to some interesting applications. It is helpful to present a few examples at this point with more details. Here 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 polynomial optimization models.

4.4.1 Matrix Combinatorial Problem

The combinatorial problem of interest is as follows. 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:

$$\begin{array}{lll} (\mathrm{MCP}) & \max & \sigma_{\max}\left(\sum_{i=1}^{n} x_i \boldsymbol{A}_i\right) \\ & \text{s.t.} & x_i \in \{1, -1\},\ i = 1, 2, \ldots, n, \end{array}$$

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 replaced 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

$$\begin{array}{ll} \max & \left\| \sum_{i=1}^{n} x_i \boldsymbol{a}^i \right\| \\ \text{s.t.} & x_i \in \{1, -1\},\ i = 1, 2, \ldots, n. \end{array}$$

This is equivalent to the model (T_{BS}) with $d = d' = 1$, whose NP-hardness is asserted by Proposition 3.5.2.

Turning back to the general matrix version (MCP), the problem has an equivalent formulation

$$\begin{array}{ll} \max & (\boldsymbol{y}^1)^{\mathrm{T}} \left(\sum_{i=1}^{n} x_i \boldsymbol{A}_i\right) \boldsymbol{y}^2 \\ \text{s.t.} & \boldsymbol{x} \in \mathbb{B}^n,\ \boldsymbol{y}^1 \in \mathbb{S}^{m_1},\ \boldsymbol{y}^2 \in \mathbb{S}^{m_2}, \end{array}$$

which is essentially the model (T_{BS}) with $d = 1$ and $d' = 2$

$$\begin{array}{ll} \max & F(\boldsymbol{x},\boldsymbol{y}^1,\boldsymbol{y}^2) \\ \text{s.t.} & \boldsymbol{x}\in\mathbb{B}^n, \boldsymbol{y}^1\in\mathbb{S}^{m_1}, \boldsymbol{y}^2\in\mathbb{S}^{m_2}, \end{array}$$

where associated with the trilinear form 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 3.5.3, 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.,

$$\begin{array}{ll} \max & \lambda_{\max}(\sum_{i=1}^n x_i\boldsymbol{A}_i) \\ \text{s.t.} & x_i\in\{1,-1\}, i=1,2,\ldots,n. \end{array}$$

It is also easy to formulate this problem as the model (H_{BS}) with $d=1$ and $d'=2$

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

whose optimal value can also be approximated with a factor of $\sqrt{\frac{2}{\pi m}}$ by Theorem 3.5.4 and the remarks that followed.

4.4.2 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 [39]), 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

$$\boldsymbol{w}'_{ij} = \begin{cases} -\boldsymbol{w}_{ij} & i \neq j, \\ -\boldsymbol{w}_{ij} + \sum_{k=1}^{n} \boldsymbol{w}_{ik} & i = j. \end{cases} \tag{4.1}$$

Observing the Cauchy–Schwartz inequality, we further formulate the above problem as

$$\begin{array}{ll} \max & \left(\sum_{1\le i,j\le n} x_i x_j \boldsymbol{w}'_{ij}\right)^{\mathrm{T}} \boldsymbol{y} = F(\boldsymbol{x},\boldsymbol{x},\boldsymbol{y}) \\ \text{s.t.} & \boldsymbol{x} \in \mathbb{B}^n, \boldsymbol{y} \in \mathbb{S}^m. \end{array}$$

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 3.5.4), 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 3.5.4.

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

$$\begin{array}{ll} \max & \lambda_{\max}\left(\sum_{1\le i,j\le n} x_i x_j \boldsymbol{W}'_{ij}\right) \\ \text{s.t.} & \boldsymbol{x} \in \mathbb{B}^n, \end{array}$$

where $\boldsymbol{W}'_{ij}$ is defined similarly as (4.1); or equivalently,

$$\begin{array}{ll} \max & \boldsymbol{y}^{\mathrm{T}}\left(\sum_{1\le i,j\le n} x_i x_j \boldsymbol{W}'_{ij}\right)\boldsymbol{y} \\ \text{s.t.} & \boldsymbol{x} \in \mathbb{B}^n, \boldsymbol{y} \in \mathbb{S}^m, \end{array}$$

the model (H_{BS}) with $d = d' = 2$. Similar to the vector-valued case, by the diagonal dominant property and Theorem 3.5.5, 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 3.5.5 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 5
Concluding Remarks

This brief discusses various classes of polynomial optimization models, and our focus is to devise polynomial-time approximation algorithms with worst-case performance guarantees. These classes of problems include many frequently encountered constraint sets in the literature, such as the Euclidean ball, the Euclidean sphere, binary hypercube, hypercube, intersection of co-centered ellipsoids, a general convex compact set, and even a mixture of them. The objective functions range from multilinear tensor functions, homogeneous polynomials, to general inhomogeneous polynomials. Multilinear tensor function optimization plays a key role in the design of algorithms. For solving multilinear tensor optimization the main construction include the following inductive components. First, for the low order cases, such problems are typically either exactly solvable, or at least approximately solvable with an approximation ratio. Then, for a one-degree-higher problem, it is often possible to relax the problem into a polynomial optimization in lower degree, which is solvable by induction. The issue of how to recover a solution for the original (higher degree) polynomial optimization problem involves a carefully devised decomposition step. We also discuss the connections between multilinear functions, homogenous polynomials, and inhomogeneous polynomials, which are established to carry over the approximation ratios to such cases. All the approximation results are listed in Table 5.1 for a quick reference. Several concrete application examples of the polynomial optimization models are presented as well; they manifest unlimited potentials of the modeling opportunities for polynomial optimization to come in the future. Table 5.1 summarizes the structure of this brief and the approximation results.

The approximation algorithms for high degree polynomial optimization discussed in this brief are certainly of great theoretical importance, considering that the worst-case approximation ratios for such optimization models are mostly new. As a matter of fact, the significance goes beyond mere theoretical bounds: they are practically efficient and effective as well. This enables us to model and solve a much broader class of problems arising from a wide variety of application domains. Furthermore, the scope of polynomial optimization can be readily extended. In fact,

Table 5.1 Brief organization and theoretical approximation ratios

Section	Model	Algorithm	Theorem	Approximation performance ratio
2.1 3.2	$(T_{\bar{S}})$ (T_S)	2.1.3 2.1.3	2.1.5 3.2.1	$\left(\prod_{k=1}^{d-2} n_k\right)^{-\frac{1}{2}}$
2.2 3.2	$(H_{\bar{S}})$ (H_S)	2.2.1, 2.2.2 2.2.1, 3.2.1	2.2.2, 2.2.4 3.2.1, 3.2.3	$d!\,d^{-d} n^{-\frac{d-2}{2}}$
2.3 3.2	$(M_{\bar{S}})$ (M_S)		2.3.2, 2.3.4 3.2.1, 3.2.4	$\begin{cases} \left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}}\right)\left(\frac{\prod_{k=1}^{s-1} {n_k}^{d_k}}{n_{s-1}}\right)^{-\frac{1}{2}} & d_s = 1 \\ \left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}}\right)\left(\frac{\prod_{k=1}^{s} {n_k}^{d_k}}{{n_s}^2}\right)^{-\frac{1}{2}} & d_s \geq 2 \end{cases}$
2.4	$(P_{\bar{S}})$	2.4.1	2.4.2	$2^{-\frac{5d}{2}}(d+1)!\,d^{-2d}(n+1)^{-\frac{d-2}{2}}$
3.1.1 3.1.5	(T_B) $(T_{\bar{B}})$	3.1.2	3.1.2 3.1.10	$\left(\frac{2}{\pi}\right)^{d-1} \ln\left(1+\sqrt{2}\right)\left(\prod_{k=1}^{d-2} n_k\right)^{-\frac{1}{2}}$
3.1.2 3.1.5	(H_B) $(H_{\bar{B}})$	3.1.3	3.1.4, 3.1.5 3.1.10	$\left(\frac{2}{\pi}\right)^{d-1} \ln\left(1+\sqrt{2}\right) d!\,d^{-d} n^{-\frac{d-2}{2}}$
3.1.3 3.1.5	(M_B) $(M_{\bar{B}})$		3.1.6, 3.1.7 3.1.10	$\begin{cases} \left(\frac{2}{\pi}\right)^{d-1} \ln\left(1+\sqrt{2}\right)\left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}}\right)\left(\frac{\prod_{k=1}^{s-1} {n_k}^{d_k}}{n_{s-1}}\right)^{-\frac{1}{2}} & d_s = 1 \\ \left(\frac{2}{\pi}\right)^{d-1} \ln\left(1+\sqrt{2}\right)\left(\prod_{k=1}^{s} \frac{d_k!}{{d_k}^{d_k}}\right)\left(\frac{\prod_{k=1}^{s} {n_k}^{d_k}}{{n_s}^2}\right)^{-\frac{1}{2}} & d_s \geq 2 \end{cases}$
3.1.4 3.1.5	(P_B) $(P_{\bar{B}})$	3.1.4	3.1.9 3.1.10	$\frac{\ln\left(1+\sqrt{2}\right)}{2(1+\mathrm{e})\pi^{d-1}}(d+1)!\,d^{-2d}(n+1)^{-\frac{d-2}{2}}$

3.3.1	(T_Q)	3.3.2	3.3.4	$\left(\prod_{k=1}^{d-2} n_k\right)^{-\frac{1}{2}} \Omega\left(\ln^{-(d-1)} \max_{1\le k\le d} m_k\right)$
3.3.2	(H_Q)	3.3.3	3.3.5, 3.3.6	$d!\,d^{-d} n^{-\frac{d-2}{2}} \Omega\left(\ln^{-(d-1)} m\right)$
3.3.3	(M_Q)		3.3.7, 3.3.8	$\begin{cases} \left(\prod_{k=1}^{s} \frac{d_k!}{d_k^{d_k}}\right)\left(\frac{\prod_{k=1}^{s-1} n_k{}^{d_k}}{n_{s-1}}\right)^{-\frac{1}{2}} \Omega\left(\ln^{-(d-1)} \max_{1\le k\le s} m_k\right) d_s = 1 \\ \left(\prod_{k=1}^{s} \frac{d_k!}{d_k^{d_k}}\right)\left(\frac{\prod_{k=1}^{s} n_k{}^{d_k}}{n_s{}^2}\right)^{-\frac{1}{2}} \Omega\left(\ln^{-(d-1)} \max_{1\le k\le s} m_k\right) d_s \ge 2 \end{cases}$
3.3.4	(P_Q)	3.3.4	3.3.9	$2^{-\frac{5d}{2}}(d+1)!\,d^{-2d}(n+1)^{-\frac{d-2}{2}} \Omega\left(\ln^{-(d-1)} m\right)$
3.4	(P_G)	3.4.1	3.4.2, 3.4.3	$2^{-2d}(d+1)!\,d^{-2d}(n+1)^{-\frac{d-2}{2}}(t^2+1)^{-\frac{d}{2}}$
3.5.1	(T_{BS})	3.5.1	3.5.3	$\left(\frac{2}{\pi}\right)^{\frac{2d-1}{2}} \left(\prod_{k=1}^{d-1} n_k \prod_{\ell=1}^{d'-1} m_\ell\right)^{-\frac{1}{2}}$
3.5.2	(H_{BS})		3.5.4, 3.5.5	$\left(\frac{2}{\pi}\right)^{\frac{2d-1}{2}} d!\,d^{-d} d'!\,d'^{-d'} n^{-\frac{d-1}{2}} m^{-\frac{d'-1}{2}}$
3.5.3	(M_{BS})		3.5.6, 3.5.7	$\left(\frac{2}{\pi}\right)^{\frac{2d-1}{2}} \left(\prod_{k=1}^{s} \frac{d_k!}{d_k^{d_k}} \prod_{\ell=1}^{t} \frac{d'_\ell!}{d'^{d'_\ell}_\ell}\right)\left(\frac{\prod_{k=1}^{s} n_k{}^{d_k} \prod_{\ell=1}^{t} m_\ell{}^{d'_\ell}}{n_s m_t}\right)^{-\frac{1}{2}}$

a number of polynomial optimization models can be straightforwardly dealt with by directly adapting our methods. Notably, the methods discussed in this brief represent one type of approach: there are alternative approximation methods for other polynomial optimization models. Before concluding this brief, we shall discuss some recent developments regarding other solution algorithms for polynomial optimization models.

As we discussed in Chap. 1, much of the theoretical development on polynomial optimization in the last ten years has been on solving general polynomial optimization problems by the theory of nonnegative polynomials and sums of squares; see, e.g., Lasserre [67]. Via a hierarchy of SDP relaxations, this method is capable of finding an optimal solution of the general model (PO). However, the computational complexity increases quickly as the hierarchy of the relaxation moves up. The size of the resulting SDP relaxation poses a serious restriction from a practical point of view. Although the method is theoretically important, numerically it only works for small size polynomial optimization models. It is certainly possible to view Lasserre's relaxation scheme within the realm of approximation method; see, e.g., the recent papers of De Klerk and Laurent [62], and Nie [91]. For instance, the hierarchical SDP relaxation scheme always yields a bound on the optimal value due to the duality theory. However, unless the optimality is reached, there is no approximate solution to be found. In this sense, the hierarchical SDP relaxation scheme and the approximation methods proposed in this brief are actually complementary to each other.

Historically, the approximation algorithms for optimizing higher degree polynomials originated from that for quadratic models, based on the SDP relaxation. Naturally, the first such attempts were targeted towards the quartic models; see Luo and Zhang [76], and Ling et al. [72]. Typically, following that direction one would end up dealing with a quadratic SDP relaxation model. In general, such relaxations are still hard to solve to optimality, but approximation solutions can be found in polynomial time. Guided by an approximate solution for the relaxed model, one can further obtain an approximate solution for the original polynomial (say quartic) optimization model. In the particular case of the models considered in Luo and Zhang [76], an approximation bound of $\Omega\left(\frac{1}{n^2}\right)$ is obtained through that route. The solution obtained through the new scheme presented in this brief, however, turns out to be better; in particular, the approximation ratio is $\Omega\left(\frac{1}{n}\right)$ if we specialize to degree four. Remark that the recent papers of Zhang et al. [120] and Ling et al. [73] considered biquadratic function optimization over quadratic constraints. Both of these papers derived approximation bounds that are data dependent.

The approach presented in this brief relies on the operations and properties of the tensor forms. Thus it is generic in some sense, and indeed it has attracted some follow-up researches. For instance, So [106] improved the approximation ratios of the models (T_S) and (H_S) to $\Omega\left(\prod_{k=1}^{d-2}\sqrt{\frac{\ln n_k}{n_k}}\right)$ and $\Omega\left(\left(\frac{\ln n}{n}\right)^{\frac{d-2}{2}}\right)$, respectively. The motivation for the study in So [106] stems from the geometric problem of which was first considered in Khot and Naor [59], who derived approximation bound on maximizing a cubic form over binary hypercube, i.e., the model (H_B) when $d = 3$. However, the method in [59] is a randomized algorithm, while that in [106] is deterministic. Later, this approach was extended to solve trilinear form optimization with non-convex constraints by Yang and Yang [114].

Very recently, He et al. [44] proposed some fairly simple randomization methods, which could further improve the approximation ratios of homogeneous form optimization over the Euclidean sphere and/or the binary hypercube, with the worst-case performance ratios/approximation ratios comparable to that in [106]. The technical analysis involves bounding the cumulative probability distribution of a polynomial of random variables. The method is simple to implement but its analysis is involved. This work was actually motivated by the analysis in Khot and Naor [59]. Moreover, the approach in [44] is capable of deriving approximation ratios for maximizing an even degree square-free homogeneous form over binary hypercube; i.e., the model (H_B) when d is even.

Given a good approximate solution, the next natural question is how to improve its quality further. In this regard, one may be led to consider some sort of local search procedure. In Chen et al. [24], a local search procedure was proposed; the local improve procedure was termed *maximum block improvement* (MBI). Specifically, they established the tightness result of multilinear form relaxation (T_S) for the model (H_S), and showed that the MBI method can be applied to enhance the approximate solution. They showed in [24] that the approach is numerically very efficient.

In the past few years, polynomial optimization has been a topic attracting much research attention. The aim of this brief is to focus on the aspect of approximation algorithms for polynomial optimization, in the hope that it will become a timely reference for the researchers in the field.

References

1. Aardal, K., Hurkens, C.A.J., Lenstra, A.K.: Solving a system of linear Diophantine equations with lower and upper bounds on the variables. Math. Oper. Res. **25**, 427–442 (2000)
2. Alfonsín, J.L.R.: The Diophantine Frobenius Problem. Oxford University Press, Oxford (2005)
3. Alon, N., de la Vega, W.F., Kannan, R., Karpinski, M.: Random sampling and approximation of MAX-CSP problems. In: Proceedings of the 34th Annual ACM Symposium on Theory of Computing, pp. 232–239 (2002)
4. Alon, N., Makarychev, K., Makarychev, Y., Naor, A.: Quadratic forms on graphs. Inventiones Mathematicae **163**, 499–522 (2006)
5. Alon, N., Naor, A.: Approximating the cut-norm via grothendieck's inequality. SIAM J. Comput. **35**, 787–803 (2006)
6. Ansari, N., Hou, E.: Computational Intelligence for Optimization. Kluwer Academic Publishers, Norwell (1997)
7. Arora, S., Berger, E., Hazan, E., Kindler, G., Safra, M.: On non-approximability for quadratic programs. In: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, pp. 206–215 (2005)
8. Artin, E.: Über die Zerlegung Definiter Funktionen in Quadrate. Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg **5**, 100–115 (1927)
9. Atamturk, A., Nemhauser, G.L., Savelsbergh, M.W.P.: Conflict graphs in solving integer programming problems. Eur. J. Oper. Res. **121**, 40–55 (2000)
10. De Athayde, G.M., Flôres, Jr., R.G.: Incorporating skewness and kurtosis in portfolio optimization: A multidimensional efficient set. In: Satchell, S., Scowcroft, A. (eds.) Advances in Portfolio Construction and Implementation, pp. 243–257, Ch. 10. Butterworth-Heinemann, Oxford (2003)
11. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties. Springer, Berlin (1999)
12. Balinski, M.L.: On a selection problem. Manag. Sci. **17**, 230–231 (1970)
13. Barmpoutis, A., Jian, B., Vemuri, B.C., Shepherd, T.M.: Symmetric positive 4th order tensors and their estimation from diffusion weighted MRI. In: Proceedings of the 20th International Conference on Information Processing in Medical Imaging, pp. 308–319 (2007)
14. Barvinok, A.: Integration and optimization of multivariate polynomials by restriction onto a random subspace. Found. Comput. Math. **7**, 229–244 (2006)
15. Beihoffer, D., Hendry, J., Nijenhuis, A., Wagon, S.: Faster algorithms for Frobenius numbers. Electr. J. Comb. **12**, R27 (2005)

16. Benson, S.J., Ye, Y.: Algorithm 875: DSDP5—Software for semidefinite programming. ACM Trans. Math. Softw. **34**, 1–16 (2008)
17. Bernhardsson, B., Peetre, J.: Singular values of trilinear forms. Exp. Math. **10**, 509–517 (2001)
18. Bertsimas, D., Ye, Y.: Semidefinite relaxations, multivariate normal distributions, and order statistics. In: Du, D.-Z., Pardalos, P.M. (eds.) Handbook of Combinatorial Optimization, vol. 3, pp. 1–19. Kluwer Academic Publishers, Norwell (1998)
19. Borchers, B.: CSDP, A C library for semidefinite programming. Opt. Meth. Softw. **11**, 613–623 (1999)
20. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
21. Bruck, J., Blaum, M.: Neural networks, error-correcting codes, and polynomials over the binary n-cube. IEEE Trans. Inf. Theory **35**, 976–987 (1989)
22. Carroll, J.D., Chang, J.-J.: Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. Psychometrika **35**, 283–319 (1970)
23. Charikar, M., Wirth, A.: Maximizing quadratic programs: Extending Grothendieck's inequality. In: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 54–60 (2004)
24. Chen, B., He, S., Li, Z., Zhang, S.: Maximum block improvement and polynomial optimization. SIAM J. Optim. **22**, 87–107 (2012)
25. Cornuéjols, G., Dawande, M.: A class of hard small 0–1 programs. INFORMS J. Comput. **11**, 205–210 (1999)
26. Dahl, G., Leinaas, J.M., Myrheim, J., Ovrum, E.: A tensor product matrix approximation problem in quantum physics. Linear Algebra Appl. **420**, 711–725 (2007)
27. De Lathauwer, L., De Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM J. Matrix Anal. Appl. **21**, 1253–1278 (2000)
28. De Lathauwer, L., De Moor, B., Vandewalle, J.: On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher order tensors. SIAM J. Matrix Anal. Appl. **21**, 1324–1342 (2000)
29. Delzell, C.N.: A continuous, constructive solution to Hilbert's 17th problem. Inventiones Mathematicae **76**, 365–384 (1984)
30. Dreesen, P., De Moor, B.: Polynomial optimization problems are eigenvalue problems, In Van den Hof, P.M.J., Scherer, C., Heuberger, P.S.C., (eds.), Model-Based Control: Bridging Rigorous Theory and Advanced Technology, Springer, Berlin, 49–68 (2009)
31. Feige, U.: Relations between average case complexity and approximation complexity. In: Proceedings of the 34th Annual ACM Symposium on Theory of Computing, pp. 534–543 (2002)
32. Feige, U., Kim, J.H., Ofek, E.: Witnesses for non-satisfiability of dense random 3CNF formulas. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 497–508 (2006)
33. Feige, U., Ofek, E.: Easily refutable subformulas of large random 3CNF forumlas. Theory Comput. **3**, 25–43 (2007)
34. Friedman, J., Goerdt, A., Krivelevich, M.: Recognizing more unsatisfiable random k-SAT instances efficiently. SIAM J. Comput. **35**, 408–430 (2005)
35. Frieze, A.M., Kannan, R.: Quick approximation to matrices and applications. Combinatorica **19**, 175–200 (1999)
36. Fujisawa, K., Kojima, M., Nakata, K., Yamashita, M.: 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 (1995)
37. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York (1979)
38. Ghosh, A., Tsigaridas, E., Descoteaux, M., Comon, P., Mourrain, B., Deriche, R.: 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. In: Proceedings of the 11th International Conference on Medical Image Computing and Computer Assisted Intervention, pp. 237–248 (2008)
39. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM **42**, 1115–1145 (1995)
40. Gurvits, L.: Classical deterministic complexity of edmonds' problem and quantum entanglement. In: Proceedings of the 35th Annual ACM Symposium on Theory of Computing, pp. 10–19 (2003)
41. Hammer, P.L., Rudeanu, S.: Boolean Methods in Operations Research. Springer, New York (1968)
42. Hansen, P.: Methods of nonlinear 0–1 programming. Ann. Discrete Math. **5**, 53–70 (1979)
43. Harshman, R.A.: 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. He, S., Jiang, B., Li, Z., Zhang, S.: Probability bounds for polynomial functions in random variables, Technical Report. Department of Industrial and Systems Engineering, University of Minnesota, Minneapolis (2012)
45. He, S., Li, Z., Zhang, S.: General constrained polynomial optimization: An approximation approach, Technical Report. Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong (2009)
46. He, S., Li, Z., Zhang, S.: Approximation algorithms for discrete polynomial optimization. Math. Progr. Ser. B **125**, 353–383 (2010)
47. He, S., Li, Z., Zhang, S.: Approximation algorithms for discrete polynomial optimization, Technical Report. Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong (2010)
48. He, S., Luo, Z.-Q., Nie, J., Zhang, S.: Semidefinite relaxation bounds for indefinite homogeneous quadratic optimization. SIAM J. Optim. **19**, 503–523 (2008)
49. Henrion, D., Lasserre, J.B.: GloptiPoly: Global optimization over polynomials with Matlab and SeDuMi. ACM Trans. Math. Softw. **29**, 165–194 (2003)
50. Henrion, D., Lasserre, J.B., Loefberg, J.: GloptiPoly 3: Moments, optimization and semidefinite programming. Optim. Meth. Softw. **24**, 761–779 (2009)
51. Hilbert, D.: Über die Darstellung Definiter Formen als Summe von Formenquadraten. Mathematische Annalen **32**, 342–350 (1888)
52. Hitchcock, F.L.: The expression of a tensor or a polyadic as a sum of products. J. Math. Phys. **6**, 164–189 (1927)
53. Hitchcock, F.L.: Multilple invariants and generalized rank of a p-way matrix or tensor. J. Math. Phys. **6**, 39–79 (2007)
54. Hopfield, J.J., Tank, D.W.: "Neural" computation of decisions in optimization problem. Biolog. Cybernet. **52**, 141–152 (1985)
55. Huang, Y., Zhang, S.: Approximation algorithms for indefinite complex quadratic maximization problems. Sci. China Math. **53**, 2697–2708 (2010)
56. Jondeau, E., Rockinger, M.: Optimal portfolio allocation under higher moments. Eur. Financ. Manag. **12**, 29–55 (2006)
57. Kann, V.: On the approximability of NP-complete optimization problems. Ph.D. Dissertation, Royal Institute of Technology, Stockholm (1992)
58. Kannan, R.: Spectral methods for matrices and tensors. In: Proceedings of the 42nd Annual ACM Symposium on Theory of Computing, pp. 1–12 (2010)
59. Khot, S., Naor, A.: Linear equations modulo 2 and the L_1 diameter of convex bodies. In: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science, pp. 318–328 (2007)
60. Kleniati, P.M., Parpas, P., Rustem, B.: Partitioning procedure for polynomial optimization: Application to portfolio decisions with higher order moments, COMISEF Working Papers Series, WPS-023 (2009)
61. De Klerk, E.: The complexity of optimizing over a simplex, hypercube or sphere: A short survey. Cent. Eur. J. Oper. Res. **16**, 111–125 (2008)

62. De Klerk, E., Laurent, M.: Error bounds for some semidefinite programming approaches to polynomial minimization on the hypercube. SIAM J. Optim. **20**, 3104–3120 (2010)
63. De Klerk, E., Laurent, M., Parrilo, P.A.: A PTAS for the minimization of polynomials of fixed degree over the simplex. Theoret. Comput. Sci. **261**, 210–225 (2006)
64. Kofidis, E., Regalia, Ph.: On the best rank-1 approximation of higher order supersymmetric tensors. SIAM J. Matrix Anal. Appl. **23**, 863–884 (2002)
65. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**, 455–500 (2009)
66. Kroó, A., Szabados, J.: Joackson-type theorems in homogeneous approximation. J. Approx. Theory **152**, 1–19 (2008)
67. Lasserre, J.B.: Global optimization with polynomials and the problem of moments. SIAM J. Optim. **11**, 796–817 (2001)
68. Lasserre, J.B.: Polynomials nonnegative on a grid and discrete representations. Trans. Am. Math. Soc. **354**, 631–649 (2002)
69. Laurent, M.: Sums of squares, moment matrices and optimization over polynomials. In: Putinar, M., Sullivant, S. (eds.) Emerging Applications of Algebraic Geometry, The IMA Volumes in Mathematics and Its Applications, vol. 149, pp. 1–114 (2009)
70. Li, Z.: Polynomial optimization problems—Approximation algorithms and applications. Ph.D. Thesis, The Chinese Univesrity of Hong Kong, Hong Kong (2011)
71. Lim, L.-H.: Singular values and eigenvalues of tensors: A variantional approach. In: Proceedings of the IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, vol. 1, pp. 129–132 (2005)
72. Ling, C., Nie, J., Qi, L., Ye, Y.: Biquadratic optimization over unit spheres and semidefinite programming relaxations. SIAM J. Optim. **20**, 1286–1310 (2009)
73. Ling, C., Zhang, X., Qi, L.: Semidefinite relaxation approximation for multivariate biquadratic optimization with quadratic constraints. Numerical Linear Algebra Appl. **19**, 113–131 (2012)
74. Luo, Z.-Q., Sidiropoulos, N.D., Tseng, P., Zhang, S.: Approximation bounds for quadratic optimization with homogeneous quadratic constraints. SIAM J. Optim. **18**, 1–28 (2007)
75. Luo, Z.-Q., Sturm, J.F., Zhang, S.: Multivariate nonnegative quadratic mappings. SIAM J. Optim. **14**, 1140–1162 (2004)
76. Luo, Z.-Q., Zhang, S.: A semidefinite relaxation scheme for multivariate quartic polynomial optimization with quadratic constraints. SIAM J. Optim. **20**, 1716–1736 (2010)
77. Mandelbrot, B.B., Hudson, R.L.: The (Mis)Behavior of Markets: A Fractal View of Risk, Ruin, and Reward. Basic Books, New York (2004)
78. Maricic, B., Luo, Z.-Q., Davidson, T.N.: Blind constant modulus equalization via convex optimization. IEEE Trans. Signal Process. **51**, 805–818 (2003)
79. Maringer, D., Parpas, P.: Global optimization of higher order moments in portfolio selection. J. Global Optim. **43**, 219–230 (2009)
80. Markowitz, H.M.: Portfolio selection. J. Finance **7**, 79–91 (1952)
81. Micchelli, C.A., Olsen, P.: Penalized maximum-likelihood estimation, the baum-welch algorithm, diagonal balancing of symmetric matrices and applications to training acoustic data. J. Comput. Appl. Math. **119**, 301–331 (2000)
82. Motzkin, T.S., Straus, E.G.: Maxima for graphs and a new proof of a theorem of Túran. Can. J. Math. **17**, 533–540 (1965)
83. Mourrain, B., Pavone, J.P.: Subdivision methods for solving polynomial equations. J. Symb. Comput. **44**, 292–306 (2009)
84. Mourrain, B., Trébuchet, P.: Generalized normal forms and polynomial system solving. In: Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation, pp. 253–260 (2005)
85. Nemirovski, A.: Lectures on Modern Convex Optimization. The H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta (2005)
86. Nemirovski, A., Roos, C., Terlaky, T.: On maximization of quadratic form over intersection of ellipsoids with common center. Math. Progr. Ser. A **86**, 463–473 (1999)

87. Nesterov, Yu.: Semidefinite relaxation and nonconvex quadratic optimization. Optim. Meth. Softw. **9**, 141–160 (1998)
88. Nesterov, Yu.: Squared functional systems and optimization problems. In: Frenk, H., Roos, K., Terlaky, T., Zhang, S. (eds.) High Performance Optimization, pp. 405–440. Kluwer Academic Press, Dordrecht (2000)
89. Nesterov, Yu.: Random walk in a simplex and quadratic optimization over convex polytopes, CORE Discussion Paper 2003/71. Université catholique de Louvain, Louvain-la-Neuve (2003)
90. Ni, Q., Qi, L., Wang, F.: An eigenvalue method for testing positive definiteness of a multivariate form. IEEE Trans. Autom. Control **53**, 1096–1107 (2008)
91. Nie, J.: An approximation bound analysis for Lasserre's relaxation in multivariate polynomial optimization, Preprint. Department of Mathematics, University of California, San Diego (2011)
92. Parpas, P., Rustem, B.: Global optimization of the scenario generation and portfolio selection problems. In: Proceedings of the International Conference on Computational Science and Its Applications, pp. 908–917 (2006)
93. Parrilo, P.A.: Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization. Ph.D. Dissertation, California Institute of Technology, Pasadena (2000)
94. Parrilo, P.A.: Semidefinite programming relaxations for semialgebraic problems. Math. Progr. Ser. B **96**, 293–320 (2003)
95. Peng, L., Wong, M.W.: Compensated compactness and paracommutators. J. London Math. Soc. **62**, 505–520 (2000)
96. Prakash, A.J., Chang, C.-H., Pactwa, T.E.: Selecting a portfolio with skewness: Recent evidence from US, European, and Latin American equity markets. J. Banking Finance **27**, 1375–1390 (2003)
97. Purser, M.: Introduction to Error-Correcting Codes. Artech House, Norwood (1995)
98. Qi, L.: Extrema of a real polynomial. J. Global Optim. **30**, 405–433 (2004)
99. Qi, L.: Eigenvalues of a real supersymmetric tensor. J. Symb. Comput. **40**, 1302–1324 (2005)
100. Qi, L.: Eigenvalues and invariants of tensors. J. Math. Anal. Appl. **325**, 1363–1377 (2007)
101. Qi, L., Teo, K.L.: Multivariate polynomial minimization and its applications in signal processing. J. Global Optim. **26**, 419–433 (2003)
102. Qi, L., Wan, Z., Yang, Y.-F.: Global minimization of normal quadratic polynomials based on global descent directions. SIAM J. Optim. **15**, 275–302 (2004)
103. Qi, L., Wang, F., Wang, Y.: Z-eigenvalue methods for a global polynomial optimization problem. Math. Progr. Ser. A **118**, 301–316 (2009)
104. Rhys, J.M.W.: A selection problem of shared fixed costs and network flows. Manag. Sci. **17**, 200–207 (1970)
105. Roberts, A.P., Newmann, M.M.: Polynomial optimization of stochastic feedback control for stable plants. IMA J. Math. Control Inf. **5**, 243–257 (1988)
106. So, A.M.-C.: Deterministic approximation algorithms for sphere constrained homogeneous polynomial optimization problems. Math. Progr. Ser. B **129**, 357–382 (2011)
107. So, A.M.-C., Ye, Y., Zhang, J.: A unified theorem on SDP rank reduction. Math. Oper. Res. **33**, 910–920 (2008)
108. Soare, S., Yoon, J.W., Cazacu, O.: On the use of homogeneous polynomials to develop anisotropic yield functions with applications to sheet forming. Int. J. Plast. **24**, 915–944 (2008)
109. Sturm, J.F.: SeDuMi 1.02, A Matlab toolbox for optimization over symmetric cones. Optim. Meth. Softw. **11 & 12**, 625–653 (1999)
110. Sturm, J.F., Zhang, S.: On cones of nonnegative quadratic functions. Math. Oper. Res. **28**, 246–267 (2003)
111. Sun, W., Yuan, Y.-X.: Optimization Theory and Methods: Nonlinear Programming. Springer, New York (2006)

112. Toh, K.C., Todd, M.J., Tütüncü, R.H.: SDPT3—A Matlab software package for semidefinite programming, version 1.3. Optim. Meth. Softw. **11**, 545–581 (1999)
113. Varjú, P.P.: Approximation by homogeneous polynomials. Construct. Approx. **26**, 317–337 (2007)
114. Yang, Y., Yang, Q.: Approximation algorithms for trilinear optimization with nonconvex constraints and its extensions, Research Report. School of Mathematics Science and LPMC, Nankai University, Tianjin (2011)
115. Ye, Y.: Approximating quadratic programming with bound and quadratic constraints. Math. Progr. **84**, 219–226 (1999)
116. Ye, Y.: Approximating global quadratic optimization with convex quadratic constraints. J. Global Optim. **15**, 1–17 (1999)
117. Zhang, S.: Quadratic maximization and semidefinite relaxation. Math. Progr. Ser. A **87**, 453–465 (2000)
118. Zhang, S., Huang, Y.: Complex quadratic optimization and semidefinite programming. SIAM J. Optim. **16**, 871–890 (2006)
119. Zhang, T., Golub, G.H.: Rank-one approximation to high order tensors. SIAM J. Matrix Anal. Appl. **23**, 534–550 (2001)
120. Zhang, X., Ling, C., Qi, L.: Semidefinite relaxation bounds for bi-quadratic optimization problems with quadratic constraints. J. Global Optim. **49**, 293–311 (2011)
121. Zhang, X., Qi, L., Ye, Y.: The cubic spherical optimization problems. Math. Comput. **81**, 1513–1525 (2012)