Optimization with Block Variables: Theory and Applications

CHEN, Bilian

A Thesis Submitted in Partial Fulfilment

of the Requirements for the Degree of

Doctor of Philosophy

in

Systems Engineering and Engineering Management

The Chinese University of Hong Kong

June 2012

Thesis/Assessment Committee

Professor LI Duan (Chair)

Professor ZHANG Shuzhong (Thesis Supervisor)

Professor SO Man Cho (Thesis Supervisor)

Professor CHEN Nan (Committee Member)

Doctor DANG Chuangyin (External Examiner)

Abstract

In this thesis we present a systematic analysis for optimization of a general nonlinear function, subject to some fairly general constraints. A typical example includes the optimization of a multilinear tensor function over spherical constraints. Such models have found wide applications in numerical linear algebra, material sciences, quantum physics, signal processing, speech recognition, biomedical engineering, and control theory. This thesis is mainly concerned with a specific approach to solve such generic models: the block variable improvement method. Specifically, we establish a block coordinate descent type search method for nonlinear optimization, which accepts only a block update that achieves the maximum improvement (hence the name of our new search method: maximum block improvement (MBI)). Then, we focus on the potential capability of this method for solving problems in various research area. First, we demonstrate that this method can be directly used in designing a new framework for co-clustering gene expression data in the area of bioinformatics. Second, we turn our attention to the spherically constrained homogeneous polynomial optimization problem, which is related to best rank-one approximation of tensors. The MBI method usually finds the global optimal solution at a low computational cost. Third, we continue to consider polynomial optimization problems. A general result between homogeneous polynomials and multi-linear forms under the concept of co-quadratic positive semidefinite is established. Finally, we consider the problem of finding the best multi-linear rank approximation of

a higher-order tensor under the framework of Tucker decomposition, and also propose a new model and algorithms for computing Tucker decomposition with unknown number of components. Some real application examples are discussed and tested, and numerical experiments are reported to reveal the good practical performance and efficiency of the proposed algorithms for solving those problems.

中文摘要

本博士论文对于有结构但又有相当一般性的约束条件下的非线性优化问题给出了系统性研究。比较经典的例子包括球面约束下的多重线性函数优化问题。这些模型已被广泛应用于数值线性代数、材料科学、量子物理学、信号处理、语音识别、生物医学工程以及控制论等。本论文着重探讨一类特定的方法来解这些广义模型，即块变量改进方法。具体地说，我们构造了一类块坐标下降型搜索算法来解带块变量结构的非线性优化问题。这类算法通过每次迭代中只更新一块变量以达到最大限度的目标函数值的改进（因而，这一新搜索算法命名为最优块改进算法（简称为 MBI））。之后，我们重点研究了该算法在求解众多领域中实际问题的潜在能力。首先，这一算法可以直接应用于生物信息学中聚类基因表达数据的一种新模型的设计及求解。接着，我们把注意力转移到球面约束下的齐次多项式优化问题，此问题与张量的最优秩-1 逼近问题相关。对于这一优化问题，MBI 算法通常可以在较少的计算时间内找到全局最优解。第三，我们继续深入研究多项式优化问题，在双协半正定的新概念下建立了齐次多项式优化问题与其多重线性优化问题关系的一般性结果。最后，我们在 *Tucker* 分解的框架下给出了求解高阶张量的最优多重线性秩的逼近问题的方法，并提出一种新的模型和算法来解未知变量数的 *Tucker* 分解问题。本论文讨论并试验了一些应用实例，数值实验表明所提出的算法分别对于求解以上这些问题是可行并有效的。

Acknowledgements

I would like to express my heartfelt thanks to my supervisor Professor Shuzhong Zhang, for his guidance and constant encouragement and support during the past three years. He brought me into various interesting research topics, and guided me through the research process. Under his guidance, I had the opportunity to get involved in some joint research work with Professor Xiuzhen Huang's research group in Arkansas State University, from which I had benefitted a lot. I would also like to thank Professor Zhang for giving me the opportunity to have a three-month visit at Industrial and Systems Engineering Program of University of Minnesota in the early year of 2012. I am greatly indebted to him.

I would like to thank my co-advisor Professor Anthony So for his kindly help and support when needed. I would also like to express my sincerest thanks to those who supported me to finish this thesis, especially to my former colleagues, Professors Simai He and Zhening Li, from whom I have been benefitted tremendously for their insightful comments and valuable suggestions in discussing technical problems. I would also like to thank Professors Duan Li, Nan Chen and Chuangyin Dang for agreeing to serve on my thesis committee.

I would also like to extend my thanks to the professors, supporting staff members and the fellow postgraduate students at Department of Systems Engineering and Engineering Management for providing me with technical and non-technical supports. It

has been a lot of fun to work with them.

Finally, I would like to express my deepest gratitude to my family, for their love and support with their whole heart all the way over the years. I am also deeply thankful to my boyfriend Boheng Qiu for his love and encouragement that helped me overcoming whatever difficulties encountered in my daily life during the pursuit of my Ph.D. degree.

This thesis is dedicated to my parents

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The optimization models whose objective and constraints are polynomial functions have recently attracted much research attention. This is in part due to an increased demand on the application side (cf. the sample applications in numerical linear algebra [91, 48, 51], material sciences [98], quantum physics [24, 33], and signal processing [29, 10, 94]) and in part due to its own strong theoretical appeal. Indeed, polynomial optimization is a challenging task; at the same time it is rich enough to be fruitful. For instance, even the simplest instances of polynomial optimization, such as maximizing a cubic polynomial over a sphere, is NP-hard (Nesterov [82]). However, the problem is so elementary that it can even be attempted in an undergraduate calculus class. For readers interested in polynomial optimization with simple constraints, see De Klerk [57] for a survey on the computational complexity of optimizing various classes of polynomial functions over a simplex, hypercube, or sphere. In particular, De Klerk *et al.* [58] designed a polynomial-time approximation scheme (PTAS) for minimizing polynomials of fixed degree over the simplex.

So far, a few results have been obtained for approximation algorithms with guaranteed worst-case performance ratios for higher degree polynomial optimization problems. Luo and Zhang [78] derived a polynomial-time approximation algorithm to optimize a multivariate quartic polynomial over a region defined by quadratic inequalities. Ling *et al.* [75] considered the problem of minimizing a biquadratic function over two spheres and proposed polynomial-time approximation algorithms. He *et al.* [40] discussed the optimization of homogeneous polynomial functions of any fixed degree over quadratic constraints and proposed approximation algorithms, with performance ratios improving that of [78, 75]. Recently, So [97] improved the approximation ratio in the case of spherically constrained homogeneous polynomial optimizations. For a general inhomogeneous polynomial optimization over convex compact sets, He *et al.* [41] proposed polynomial-time approximation algorithms with relative approximation ratios, which is the only result so far with regard to approximation algorithms for an inhomogeneous polynomial. Later, the authors extended their results in [42] by considering polynomials in discrete (typically binary) variables and designed randomized approximation algorithms. For a recent treatise on the topic, one may refer to the Ph.D. thesis of Li [73].

On the computational side, polynomial optimization problems can be treated as nonlinear programming, and many existing software packages are available, including KNITRO, BARON, MINOS, SNOPT, and the Matlab optimization toolbox. (We refer the interested reader to NEOS server [84] for further information.) However, these solvers are not tailor-made for polynomial optimization problems, and so the performance may vary greatly from one problem instance to another. Therefore, it is natural to wonder whether one can design efficient algorithms for specific types of polynomial optimization problems. Prajna *et al.* [90] presented the package SOSTOOLS for solving sum of squares (SOS) polynomial programs, based on the SOS decomposition for

multivariate polynomials, which can be computed using (possibly large-size) semidefinite programs. More recently, Henrion *et al.* [46] developed a specialized tool known as GloptiPoly 3 (a later version of GloptiPoly; see Henrion and Lasserre [45]) in finding global optimal solutions for polynomial optimizations based on the SOS approach (see [63, 64, 66, 87, 88] for details). GloptiPoly 3 calls the semidefinite programming (SDP) solver SeDuMi [99]. Therefore, due to the limitation to solve large SDP problems, GloptiPoly 3 may not be the right tool to deal with large-size polynomials (say, a sixth-degree polynomial in 20 variables). However, if the polynomial optimization model in question is sparse in some way, then it is possible to exploit the sparsity in GloptiPoly 3; see [65]. As a matter of fact, SparsePOP [110] makes use of the sparsity explicitly and is a more appropriate alternative for sparse polynomial optimization based on the SOS approach. For more information on polynomial optimization, we refer to the recent book of Anjos and Lasserre [5] and the references therein.

Polynomial optimization can be related to the problem of higher-order tensor decomposition. For instance, an important application of spherically constrained homogeneous polynomial optimizations is the best (in the least-squares sense) rank-one approximation of tensors (sometimes also known as the rank-one factorization): find vectors $x^1, x^2, \cdots, x^d$ for the following minimization problem

$$\min \sum_{i_1, i_2, \cdots, i_d} \left(x^1_{i_1} x^2_{i_2} \cdots x^d_{i_d} - \mathcal{F}_{i_1 i_2 \cdots i_d} \right)^2,$$

where $\mathcal{F} = (\mathcal{F}_{i_1 i_2 \cdots i_d})$ is a dth-order tensor. In particular, if the tensor $\mathcal{F}$ is supersymmetric (every element $\mathcal{F}_{i_1 i_2 \cdots i_d}$ is invariant under all permutations of $(i_1, i_2, \cdots, i_d)$), then the optimal vectors $x^1, x^2, \cdots, x^d$ should coincide (namely, they should be equal to each other). The main workhorse for solving the above tensor problem is known as the *alternating least square* (ALS) method proposed originally by Carroll and Chang [18] and Harshman [34]. However, the ALS method is not guaranteed to converge to a

global minimum or a stationary point, only to a solution where the objective function ceases to decrease. Anyway, this relationship suggests an alternative method to handle some polynomial optimizations. Hence, the higher-order tensor decomposition attracts our interest in this thesis.

In fact, the problem of high order tensor decomposition first developed in psychometrics (see, e.g., [107, 108, 109]) and chemometrics (see, e.g., [6]), since they need to analyze multiway data. Later it has been studied by mathematicians who are interested in algebraic properties of tensors, as well as by engineers and statisticians who are interested in high order (tensor) statistics and independent component analysis (ICA). There are two particular tensor decompositions that can be considered to be higher-order extensions of the matrix singular value decomposition (SVD): one is CANDECOMP/PARAFAC (CP), which decomposes a tensor as a sum of rank-one tensors, the other one is the Tucker decomposition, which is a higher-order form of principal component analysis (PCA). Tucker decomposition is often considered as best rank-$(r_1, r_2, \cdots, r_d)$ approximation of tensors; see, e.g., [69]. They are both NP-hard problems in general. However, a direct generalization of the SVD is nontrivial, since the definition of a rank that preserves all of the good properties of the SVD does not exist; see, e.g., [61, 68]. Due to the lack of a good tensor rank definition, there is no "best" way to define low-rank approximation for tensors of order higher than two, as pointed out in [71]. In [61], Kolda and Bader described many kinds of definitions of tensor rank, and to determine the value of which are usually NP-hard [37]. There are a number of attempts trying to find CP decomposition and Tucker decomposition; see, e.g., [69, 68, 59, 62, 71, 93]. Computationally, the existing popular algorithms for CP decomposition and Tucker decomposition are based on the ALS method and its extensions. However, as mentioned earlier, the ALS method has no convergence guarantee. Although it is hard to find the best CP decomposition given the rank of the tensor

in general, there are some algorithms designed to estimate an appropriate rank in the CP decomposition; see, e.g., a consistency diagnostic named CORCONDIA [16]. The same happened for the Tucker decomposition where one needs to select suitable ranks $r_1, r_2, \ldots, r_d$; see, e.g., [105, 56]. For an overview on the recent developments on tensor decomposition, we refer to the excellent survey by Kolda and Bader [61].

In this thesis we propose a new method, called Maximum Block Improvement (MBI) method. This method actually has a general appeal: it can be applied to solve any optimization model with separate block constraints. The general scheme of Maximum Block Improvement method is introduced and also its convergence analysis is given in Chapter 2. In the next following chapters, we will see that our new method, MBI, is capable of co-clustering of gene expression data with genes expressed at different multiway forms (see Chapter 3), solving spherically constrained homogeneous polynomial optimization problems (see Chapter 4), dealing with a particular polynomial optimization problem over any constraint set (see Chapter 5), and finding the best approximation for Tucker decomposition with unknown number of components (see Chapter 6). Numerical results show that MBI method is promising. And it has potential to solve other problems.

1.2 Notations and Preliminaries

Throughout this thesis, we use non-bold letters, boldface lowercase letters, capital letters, and calligraphic letters to denote scalars, vectors, matrices, and tensors in general, respectively. For example, scalar i, vector $\boldsymbol{x}$, matrix A, and tensor $\mathcal{F}$. And we use subscripts to denote the component of a vector, a matrix or a tensor, e.g., x_i being the i-th entry of the vector $\boldsymbol{x}$, A_{ij} being the (i,j)-th element of the matrix A, and $\mathcal{F}_{ijk}$ being the (i,j,k)-th component of the tensor $\mathcal{F}$. For vector $\boldsymbol{x} = (x_1, x_2, \cdots, x_d)^T \in \mathbb{R}^d$,

$\mathrm{Diag}\,(\boldsymbol{x})$ denotes diagonal matrix $\begin{pmatrix} x_1 & & & \\ & x_2 & & \\ & & \ddots & \\ & & & x_d \end{pmatrix}$.

1.2.1 The Tensor Operations

We introduce some tensor operations needed in this thesis, which is largely in line with that in [61, 68]. For an overview of tensor operations and properties, we refer to the survey paper [61].

A tensor is a multidimensional array, and the order of a tensor is its dimension, also known as the ways or the modes of a tensor. In particular, a vector is a tensor of order one, and a matrix is a tensor of order two. Let $\mathcal{A} = (\mathcal{A}_{i_1 i_2 \cdots i_d}) \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ be a tensor of order d, where $d \geq 3$. We stick to $\mathcal{A}$ as a standard tensor, for the purpose of introducing the tensor operations. Analogous to the definition of a symmetric matrix, we call tensor $\mathcal{A}$ *super-symmetric* if every element $\mathcal{A}_{i_1 i_2 \cdots i_d}$ is invariant under all permutations of $(i_1, i_2, \cdots, i_d)$ when $n_1 = n_2 = \cdots = n_d$.

The usual way to handle a tensor is to reorder its elements into a matrix; the process is called *matricization*, also known as *unfolding or flattening*. Tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ has totally d modes, namely, mode-1, mode-2, $\cdots$, mode-d. Denote the *mode-k matricization* of tensor $\mathcal{A}$ to be $A_{(k)}$, then the element $(i_1, i_2, \ldots, i_d)$ of tensor $\mathcal{A}$ is mapped to the element (i_k, j) of matrix $A_{(k)}$, where

$$j = 1 + \sum_{l=1, l \neq k}^{d} (i_l - 1) J_l \quad \text{with} \quad J_l = \prod_{m=1, m \neq k}^{l-1} n_m,$$

Therefore, $A_{(k)} \in \mathbb{R}^{n_k \times I_k}$, where $I_k = \prod_{l=1, l \neq k}^{d} n_l$. For example, consider third-order

tensor $\mathcal{G} \in \mathbb{R}^{2\times 3\times 2}$ with entries

$$\mathcal{G}_{111} = 1, \quad \mathcal{G}_{211} = 2, \quad \mathcal{G}_{121} = 3, \quad \mathcal{G}_{221} = 4, \quad \mathcal{G}_{131} = 5, \quad \mathcal{G}_{231} = 6,$$
$$\mathcal{G}_{112} = 7, \quad \mathcal{G}_{212} = 8, \quad \mathcal{G}_{122} = 9, \quad \mathcal{G}_{222} = 10, \quad \mathcal{G}_{132} = 11, \quad \mathcal{G}_{232} = 12.$$

Then the three mode-k unfoldings are

$$G_{(1)} = \begin{pmatrix} 1 & 3 & 5 & 7 & 9 & 11 \\ 2 & 4 & 6 & 8 & 10 & 12 \end{pmatrix},$$

$$G_{(2)} = \begin{pmatrix} 1 & 2 & 7 & 8 \\ 3 & 4 & 9 & 10 \\ 5 & 6 & 11 & 12 \end{pmatrix},$$

$$G_{(3)} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \end{pmatrix}.$$

Analogous to the Frobenius norm of a matrix, the *Frobenius norm* of tensor $\mathcal{A}$ is defined by

$$\|\mathcal{A}\| := \sqrt{\sum_{1\le i_1\le n_1, 1\le i_2\le n_2, \cdots, 1\le i_d\le n_d} {\mathcal{A}_{i_1 i_2 \cdots i_d}}^2}.$$

We shall use the 2-norm for vectors, and the Frobenius norm for matrices and tensors, all by using notation $\|\cdot\|$. The *inner product* of two same-sized tensors $\mathcal{A}, \mathcal{B}$ is given as

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} \mathcal{A}_{i_1 i_2 \cdots i_d} \mathcal{B}_{i_1 i_2 \cdots i_d}.$$

Hence, it is clear that $\langle \mathcal{A}, \mathcal{A} \rangle = \|\mathcal{A}\|^2$.

One important tensor operation in this thesis is the multiplication of a tensor by a matrix. The *k-mode product* of tensor $\mathcal{A}$ by a matrix $U \in \mathbb{R}^{m\times n_k}$, denoted by $\mathcal{A} \times_k U$, is a tensor in $\mathbb{R}^{n_1\times n_2\times\cdots\times n_{k-1}\times m\times n_{k+1}\times\cdots\times n_d}$, whose $(i_1, i_2, \cdots, i_{k-1}, l, i_{k+1}, \cdots, i_d)$-th component is defined by

$$(\mathcal{A} \times_k U)_{i_1\, i_2 \cdots i_{k-1}\, l\, i_{k+1} \cdots i_d} = \sum_{i_k=1}^{n_k} \mathcal{A}_{i_1\, i_2 \cdots i_{k-1}\, i_k\, i_{k+1} \cdots i_d} U_{l\, i_k}.$$

The equation can be rewritten in terms of tensor unfolding as well; i.e.,

$$\mathcal{Y} = \mathcal{A} \times_k U \Longleftrightarrow Y_{(k)} = U A_{(k)}.$$

We see that this multiplication changes the dimension of tensor $\mathcal{A}$ in mode-k. In particular, if U is a vector in space $\mathbb{R}^{n_k}$, order of tensor $\mathcal{A} \times_k U$ is reduced to $d-1$, whose size is $n_1 \times n_2 \times \cdots \times n_{k-1} \times n_{k+1} \times \cdots \times n_d$.

We use the $\circ$ symbol to represent the *vector outer product*. For example, for vectors $\boldsymbol{x} \in \mathbb{R}^{n_1}, \boldsymbol{y} \in \mathbb{R}^{n_2}, \boldsymbol{z} \in \mathbb{R}^{n_3}$, the notion $\boldsymbol{x} \circ \boldsymbol{y} \circ \boldsymbol{z}$ defines a third-order tensor $\mathcal{F} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, whose (i,j,k)-th element is given by

$$\mathcal{F}_{ijk} = x_i \, y_j \, z_k.$$

The $\otimes$ symbol denotes *matrix Kronecker product*. Let $A \in \mathbb{R}^{n_1 \times n_2}, B \in \mathbb{R}^{n_3 \times n_4}$, then $A \otimes B$ is a matrix of size $(n_1 n_2) \times (n_3 n_4)$ and defined by

$$A \otimes B = \begin{pmatrix} A_{11}B & A_{12}B & \cdots & A_{1\,n_2}B \\ A_{21}B & A_{22}B & \cdots & A_{2\,n_2}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{n_1\,1}B & A_{n_1\,2}B & \cdots & A_{n_1\,n_2}B \end{pmatrix}.$$

For example, $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$, $B = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$, then

$$A \otimes B = \begin{pmatrix} 1\cdot 5 & 1\cdot 6 & 2\cdot 5 & 2\cdot 6 \\ 1\cdot 7 & 1\cdot 8 & 2\cdot 7 & 2\cdot 8 \\ 3\cdot 5 & 3\cdot 6 & 4\cdot 5 & 4\cdot 6 \\ 3\cdot 7 & 3\cdot 8 & 4\cdot 7 & 4\cdot 8 \end{pmatrix} = \begin{pmatrix} 5 & 6 & 10 & 12 \\ 7 & 8 & 14 & 16 \\ 15 & 18 & 20 & 24 \\ 21 & 24 & 28 & 32 \end{pmatrix}.$$

Moreover, the matrix Kronecker product can be used in the following useful expression

$$\begin{aligned} & \mathcal{Y} = \mathcal{A} \times_1 U^{(1)} \times_2 U^{(2)} \cdots \times_d U^{(d)} \\ \Longleftrightarrow \quad & Y_{(k)} = U^{(k)} A_{(k)} \left(U^{(d)} \otimes \cdots \otimes U^{(k+1)} \otimes U^{(k-1)} \otimes \cdots \otimes U^{(1)} \right), \end{aligned}$$

for any $k \in \{1, 2, \cdots, d\}$, where $U^{(k)} \in \mathbb{R}^{m_k \times n_k}, k = 1, 2, \cdots, d$. The proof of this property can be found in [60].

In this special circumstance, throughout this thesis we shall use a subscript in parentheses to denote the matricization of a tensor (e.g., $A_{(1)}$ being mode-1 matricization of tensor $\mathcal{A}$), and use a superscript in parentheses to denote the matrix in the operation of mode product of a tensor (e.g., $U^{(1)}$ in appropriate dimensions showed in mode-1 product of a tensor).

1.2.2 The Tensor Ranks

There are two most common ways to define the rank of a tensor; see, e.g., [61, 68]. First, analogous to the notion of column and row rank of a matrix, one then have one way to define the rank of a tensor by using all the mode unfoldings. Formally, the *k-rank* of tensor $\mathcal{A}$, denoted by $\text{rank}_k(\mathcal{A})$, is the column rank of mode-k unfolding $A_{(k)}$, and

$$\text{rank}_k(\mathcal{A}) = \text{rank}\left(A_{(k)}\right).$$

For example, consider $(2 \times 2 \times 2)$-tensor $\mathcal{X}$ with entries

$$\mathcal{X}_{111} = \mathcal{X}_{221} = \mathcal{X}_{112} = \mathcal{X}_{222} = 1,$$
$$\mathcal{X}_{211} = \mathcal{X}_{121} = \mathcal{X}_{212} = \mathcal{X}_{122} = 0,$$

then the three mode-k unfoldings are

$$X_{(1)} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, X_{(2)} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, X_{(3)} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

Hence, we have $\text{rank}_1(\mathcal{X}) = \text{rank}_2(\mathcal{X}) = 2$, while $\text{rank}_3(\mathcal{X}) = 1$. We see that the k-ranks of a higher-order tensor are not necessarily the same, which is a major difference compared to the matrix case, where the column rank is equal to the row rank. As we shall see later, this definition corresponds to rank-$(r_1, r_2, \cdots, r_d)$ approximation of a tensor given that $r_k = \text{rank}_k(\mathcal{A})$ for $k = 1, 2, \cdots, d$.

Next, since a rank-k matrix can be decomposed as a sum of k rank-one terms, we can define the *rank* of tensor $\mathcal{A}$, denoted by $\operatorname{rank}(\mathcal{A})$, as the smallest number of rank-one tensors that generate $\mathcal{A}$ as their sum. This definition corresponds to the CP decomposition as follows:

$$\mathcal{A} = \sum_{k=1}^{\operatorname{rank}(\mathcal{A})} \lambda_k \, \boldsymbol{x}_k^1 \circ \boldsymbol{x}_k^2 \circ \cdots \circ \boldsymbol{x}_k^d.$$

In particular, it is related to the best rank-one decomposition of a tensor if $\operatorname{rank}(\mathcal{A}) = 1$. However, it turns out that determining the rank of a specific given tensor is NP-hard [37]. Besides, the rank of a tensor is not necessarily equal to a k-rank, even when all the k-ranks are the same. Let us consider a simple example. Let $(2 \times 2 \times 2)$-tensor $\mathcal{F}$ with entries

$$\begin{aligned} \mathcal{F}_{111} = \mathcal{F}_{221} = \mathcal{F}_{122} = \mathcal{F}_{212} = \mathcal{F}_{222} = 0, \\ \mathcal{F}_{211} = \mathcal{F}_{121} = \mathcal{F}_{112} = 1. \end{aligned}$$

Clearly, in this case 1-rank, 2-rank, and 3-rank are all equal to 2. However, $\operatorname{rank}(\mathcal{F}) = 3$, since

$$\mathcal{F} = \boldsymbol{x}_2 \circ \boldsymbol{y}_1 \circ \boldsymbol{z}_1 + \boldsymbol{x}_1 \circ \boldsymbol{y}_2 \circ \boldsymbol{z}_1 + \boldsymbol{x}_1 \circ \boldsymbol{y}_1 \circ \boldsymbol{z}_2,$$

in which

$$\boldsymbol{x}_1 = \boldsymbol{y}_1 = \boldsymbol{z}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \boldsymbol{x}_2 = \boldsymbol{y}_2 = \boldsymbol{z}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

is a decomposition in smallest number of rank-one tensors. The proof can be found in the Ph.D. thesis of Lathauwer [67].

For other rank definitions and properties concerning tensors (e.g., maximum rank, typical rank), we refer to the survey [61].

1.2.3 Polynomial Functions

In this part, we introduce three important polynomial functions. Let us first consider the following multilinear tensor function

$$F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) = \sum_{1\le i_1\le n_1, 1\le i_2\le n_2, \cdots, 1\le i_d\le n_d} \mathcal{F}_{i_1 i_2 \cdots i_d} x^1_{i_1} x^2_{i_2} \cdots x^d_{i_d}, \tag{1.1}$$

where $\boldsymbol{x}^k \in \mathbb{R}^{n_k}$ for $k = 1, 2, \ldots, d$, and $\mathcal{F} = (\mathcal{F}_{i_1 i_2 \cdots i_d}) \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is a dth-order tensor with F being its associated multilinear function. Closely related to the tensor form $\mathcal{F}$ is a general dth-degree homogeneous polynomial function $f(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{R}^n$, with its associated tensor $\mathcal{F}$ being super-symmetric. In fact, super-symmetric tensors are bijectively related to homogeneous polynomials; see [61]. Denote F to be the multilinear function defined by the super-symmetric tensor form $\mathcal{F}$, we then have

$$f(\boldsymbol{x}) = F(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x}}_{d}). \tag{1.2}$$

A generic multi-variate (inhomogeneous) polynomial function of degree d, denoted by $p(\boldsymbol{x})$, can be explicitly written as a summation of homogeneous polynomial functions in decreasing degrees, namely,

$$p(\boldsymbol{x}) := \sum_{i=1}^{d} f_i(\boldsymbol{x}) + f_0 = \sum_{i=1}^{d} F_i(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x}}_{i}) + f_0, \tag{1.3}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $f_0 \in \mathbb{R}$, and $f_i(\boldsymbol{x}) = F_i(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x}}_{i})$ is a homogeneous polynomial function of degree i for $i = 1, 2, ..., d$.

To avoid triviality, we assume that at least one component of any tensor form throughout this thesis (e.g., $\mathcal{F}$ in functions F and f, and $\mathcal{F}_d$ in function p) is nonzero.

Finally, as a matter of notation, for any given maximization problem (P) with objective function $p(\boldsymbol{x})$ and constraint set S we shall denote its optimal value by $v(P)$, and use $\underline{v}(P)$ to denote the optimal value of its minimization counterpart, i.e.,

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

Chapter 2

The Maximum Block Improvement Method

2.1 Introduction

The focus of this chapter is to design an algorithm for an important generic optimization problem, which plays an important role in this thesis. Specifically, the model we consider is in the form of

$$\begin{aligned} (G) \quad & \max \quad f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) \\ & \text{s.t.} \quad \boldsymbol{x}^i \in S^i \subseteq \mathbb{R}^{n_i},\ i = 1, 2, ..., d, \end{aligned}$$

where $f : \mathbb{R}^{n_1+\cdots+n_d} \to \mathbb{R}$ is a general continuous function, and S^i is a general set, $i = 1, 2, ..., d$. Remark that we do not make any convexity (or concavity) assumption on the objective function f, nor on the constraint sets S^i.

This model has attracted much research attention for years. For the popular special case where $S^i = \mathbb{R}^{n_i}$, $i = 1, 2, ..., d$, a method known as the *block coordinate descent* (BCD) is well studied; see Tseng [106] and the references therein. In fact, the so-called

ALS method for tensor decomposition problems (see Chapter 1) is a special form of the BCD method. The BCD method calls for maximizing one block, say, $\boldsymbol{x}^i \in \mathbb{R}^{n_i}$, at one time, while all other variables in other blocks are temporarily fixed. One then moves on to alter the choice of the blocks. Very recently, Wright [112] introduced an extension based on BCD. Typically, under various convexity assumptions on the objective function, one is able to show some convergence property of the BCD method (cf. [106]). In fact, the BCD method can be applied regardless of any convexity assumptions, as long as one is able to optimize over one block of variable while fixing all others. A summary of the BCD or other block search methods can be found in Bertsekas [12]. The approach has a relatively long history (it is also known as the block nonlinear Gauss–Seidel method). Without taking any precaution, the BCD method may not be convergent; see the examples in Powell [89]. In the literature, this issue of convergence has been thoroughly studied. However, the results were not entirely satisfactory. To ensure convergence, either one would require some type of convexity (cf. the discussion in [12]) or the search routine is modified (cf. a proximal-point modification in Grippo and Sciandrone [32]). As we will show later, our new block search method does not modify the objective function in the block-search subroutine and at the same time ensures the convergence to a stationary solution within the structure of the BCD framework.

Also, the model is reminiscent of a noncooperative game, where S^i can be regarded as the strategy set of player i, $i = 1, 2, ..., d$. Certainly, in the case of noncooperative game, the objective of each player may be different in general. In a channel spectrum allocation game in communication, the corresponding BCD approach is also known as the *iterative waterfilling algorithm* (IWA); Luo and Pang [76] show that the IWA is convergent to the Nash equilibrium under some fairly loose conditions. It is possible that the IWA may cycle in the absence of these conditions; see an example in [39].

Motivated by the BCD method for nondifferentiable minimization proposed by Tseng [106] and the IWA for multiuser power control in digital subscriber lines by Luo and Pang [76], in this chapter we shall propose a different method, to be called the *maximum block improvement* (MBI), which guarantees convergence to a stationary point of the optimization problem (G). We shall remark that this method is suitable for solving any optimization model with separate block constraints.

2.2 MBI Method and Convergence Analysis

In this section, the general scheme of the MBI method is introduced, and its convergence property is discussed.

To simplify the analysis, we assume here that S^i is compact, $i = 1, 2, ..., d$. But that alone is insufficient to guarantee the convergence, as we know that even for the special case of the ALS, the iterates may not converge to a stationary point; see e.g., [30, 69, 70, 96]. A sufficient condition for convergence is to take a step that corresponds to the *maximum* improvement. The enhanced procedure is as follows.

Algorithm MBI. The MBI method for solving (G).

---

0 *(Initialization).* Choose a feasible solution $(\boldsymbol{x}_0^1, \boldsymbol{x}_0^2, \cdots, \boldsymbol{x}_0^d)$ with $\boldsymbol{x}_0^i \in S^i$ for $i = 1, 2, \ldots, d$ and compute initial objective value $v_0 := f(\boldsymbol{x}_0^1, \boldsymbol{x}_0^2, \cdots, \boldsymbol{x}_0^d)$. Set $k := 0$.

1 *(Block Improvement).* For each $i = 1, 2, \ldots, d$, solve

$$(G_i) \quad \begin{array}{ll} \max & f(\boldsymbol{x}_k^1, \cdots, \boldsymbol{x}_k^{i-1}, \boldsymbol{x}^i, \boldsymbol{x}_k^{i+1}, \cdots, \boldsymbol{x}_k^d) \\ \text{s.t.} & \boldsymbol{x}^i \in S^i, \end{array}$$

and let

$$\begin{aligned} \boldsymbol{y}_{k+1}^i &:= \arg\max_{\boldsymbol{x}^i \in S^i} f(\boldsymbol{x}_k^1, \cdots, \boldsymbol{x}_k^{i-1}, \boldsymbol{x}^i, \boldsymbol{x}_k^{i+1}, \cdots, \boldsymbol{x}_k^d), \\ w_{k+1}^i &:= f(\boldsymbol{x}_k^1, \cdots, \boldsymbol{x}_k^{i-1}, \boldsymbol{y}_{k+1}^i, \boldsymbol{x}_k^{i+1}, \cdots, \boldsymbol{x}_k^d). \end{aligned}$$

2 *(Maximum Improvement).* Let $w_{k+1} := \max_{1\le i\le d} w_{k+1}^i$ and $i^* = \arg\max_{1\le i\le d} w_{k+1}^i$. Let

$$\begin{aligned} \boldsymbol{x}_{k+1}^i &:= \boldsymbol{x}_k^i, \ \forall\ i \in \{1, 2, \cdots, d\}\backslash\{i^*\}, \\ \boldsymbol{x}_{k+1}^{i^*} &:= \boldsymbol{y}_{k+1}^{i^*}, \\ v_{k+1} &:= w_{k+1}. \end{aligned}$$

3 *(Stopping Criterion).* If $|v_{k+1} - v_k| < \epsilon$, stop. Otherwise, set $k := k + 1$, and go to Step 1.

The key assumption in the above process is that problem (G_i) can be easily solved, which is the case for many applications. For instance, when $f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) = -\|\mathcal{F} - \boldsymbol{x}^1 \circ \boldsymbol{x}^2 \circ \cdots \circ \boldsymbol{x}^d\|^2$, and $S^i = \mathbb{R}^{n_i}$, then (G_i) is simply a least squares problem; when $f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d)$ is a multilinear tensor form and S^i is convex, then (G_i) is a convex optimization problem. A major difference between MBI and IWA (or, for that matter, ALS, BCD, or block nonlinear Gauss–Seidel) lies in Step 2: rather than improving among block decision variables alternatively or cyclically, MBI chooses to update the block variables that achieve the maximum improvement. If in Step 2, solving (G_i) is a least squares problem, then MBI becomes a variant of the ALS method, which is widely used in tensor decompositions (cf. [61]). Unlike the ALS method, as we show next, the MBI method guarantees converging to a stationary point.

Theorem 2.2.1. *If S^i is compact, $i = 1, 2, ..., d$, then any cluster point of the iterates $(\boldsymbol{x}_k^1, \boldsymbol{x}_k^2, \cdots, \boldsymbol{x}_k^d)$, say, $(\boldsymbol{x}_*^1, \boldsymbol{x}_*^2, \cdots, \boldsymbol{x}_*^d)$, will be a stationary point for (G); i.e.,*

$$\boldsymbol{x}_*^i = \arg\max_{\boldsymbol{x}^i \in S^i} f(\boldsymbol{x}_*^1, \cdots, \boldsymbol{x}_*^{i-1}, \boldsymbol{x}^i, \boldsymbol{x}_*^{i+1}, \cdots, \boldsymbol{x}_*^d) \quad \forall\, i = 1, 2, ..., d.$$

Proof. For each fixed $(\boldsymbol{x}^1, \cdots, \boldsymbol{x}^{i-1}, \boldsymbol{x}^{i+1}, \cdots, \boldsymbol{x}^d)$, denote $R_i(\boldsymbol{x}^1, \cdots, \boldsymbol{x}^{i-1}, \boldsymbol{x}^{i+1}, \cdots, \boldsymbol{x}^d)$ to be a best response function to $\boldsymbol{x}^i$, namely,

$$R_i(\boldsymbol{x}^1, \cdots, \boldsymbol{x}^{i-1}, \boldsymbol{x}^{i+1}, \cdots, \boldsymbol{x}^d) \in \arg\max_{\boldsymbol{x}^i \in S^i} f(\boldsymbol{x}^1, \cdots, \boldsymbol{x}^{i-1}, \boldsymbol{x}^i, \boldsymbol{x}^{i+1}, \cdots, \boldsymbol{x}^d).$$

Suppose that $(\boldsymbol{x}_{k_t}^1, \boldsymbol{x}_{k_t}^2, \cdots, \boldsymbol{x}_{k_t}^d) \to (\boldsymbol{x}_*^1, \boldsymbol{x}_*^2, \cdots, \boldsymbol{x}_*^d)$ as $t \to \infty$. Then, for any $1 \le i \le d$, we have

$$\begin{aligned}
& f(\boldsymbol{x}_{k_t}^1, \cdots, \boldsymbol{x}_{k_t}^{i-1}, R_i(\boldsymbol{x}_*^1, \cdots, \boldsymbol{x}_*^{i-1}, \boldsymbol{x}_*^{i+1}, \cdots, \boldsymbol{x}_*^d), \boldsymbol{x}_{k_t}^{i+1}, \cdots, \boldsymbol{x}_{k_t}^d) \\
\le\ & f(\boldsymbol{x}_{k_t}^1, \cdots, \boldsymbol{x}_{k_t}^{i-1}, R_i(\boldsymbol{x}_{k_t}^1, \cdots, \boldsymbol{x}_{k_t}^{i-1}, \boldsymbol{x}_{k_t}^{i+1}, \cdots, \boldsymbol{x}_{k_t}^d), \boldsymbol{x}_{k_t}^{i+1}, \cdots, \boldsymbol{x}_{k_t}^d) \\
\le\ & f(\boldsymbol{x}_{k_t+1}^1, \boldsymbol{x}_{k_t+1}^2, \cdots, \boldsymbol{x}_{k_t+1}^d) \\
\le\ & f(\boldsymbol{x}_{k_{t+1}}^1, \boldsymbol{x}_{k_{t+1}}^2, \cdots, \boldsymbol{x}_{k_{t+1}}^d).
\end{aligned}$$

By continuity, when $t \to \infty$, it follows that

$$f(\boldsymbol{x}_*^1, \cdots, \boldsymbol{x}_*^{i-1}, R_i(\boldsymbol{x}_*^1, \cdots, \boldsymbol{x}_*^{i-1}, \boldsymbol{x}_*^{i+1}, \cdots, \boldsymbol{x}_*^d), \boldsymbol{x}_*^{i+1}, \cdots, \boldsymbol{x}_*^d) \le f(\boldsymbol{x}_*^1, \boldsymbol{x}_*^2, \cdots, \boldsymbol{x}_*^d),$$

which implies that the above should hold as an equality, since the other inequality is true by the definition of the best response function R_i. Thus, $\boldsymbol{x}_*^i$ is the best response to $(\boldsymbol{x}_*^1, \cdots, \boldsymbol{x}_*^{i-1}, \boldsymbol{x}_*^{i+1}, \cdots, \boldsymbol{x}_*^d)$, or equivalently, $\boldsymbol{x}_*^i$ is the optimal solution for the problem

$$\max_{\boldsymbol{x}^i \in S^i} \ f(\boldsymbol{x}_*^1, \cdots, \boldsymbol{x}_*^{i-1}, \boldsymbol{x}^i, \boldsymbol{x}_*^{i+1}, \cdots, \boldsymbol{x}_*^d),$$

for all $i = 1, 2, ..., d$. □

In many applications, S^i is described by inequalities and equalities; e.g.,

$$S^i = \{\boldsymbol{x}^i \in \mathbb{R}^{n_i} \mid g_j^i(\boldsymbol{x}^i) \le 0,\ j = 1, 2, \ldots, m_i;\ h_j^i(\boldsymbol{x}^i) = 0,\ j = 1, 2, \ldots, \ell_i\},$$

where $i = 1, 2, ..., d$. It is then more convenient to use the so-called KKT conditions, instead of an abstract form of the stationarity, under some constraint qualifications (CQ).[1]

Corollary 2.2.2. *If* $S^i = \{\boldsymbol{x}^i \in \mathbb{R}^{n_i} \mid g_j^i(\boldsymbol{x}^i) \leq 0,\ j = 1, 2, \ldots, m_i;\ h_j^i(\boldsymbol{x}^i) = 0,\ j = 1, 2, \ldots, \ell_i\}$ *is compact for all* $i = 1, 2, ..., d$, *and it satisfies a suitable constraint qualification (cf. footnote 1), then any cluster point of the iterates* $(\boldsymbol{x}_k^1, \boldsymbol{x}_k^2, \cdots, \boldsymbol{x}_k^d)$, *say* $(\boldsymbol{x}_*^1, \boldsymbol{x}_*^2, \cdots, \boldsymbol{x}_*^d)$, *will be a KKT point for* (G).

Proof. As asserted by Theorem 2.2.1, $(\boldsymbol{x}_*^1, \boldsymbol{x}_*^2, \cdots, \boldsymbol{x}_*^d)$ is a stationary point. Moreover, since a constraint qualification is satisfied for S^i, we know that $\boldsymbol{x}_*^i$ is a KKT point as well. Namely, there exist u_j^i and v_j^i such that $\boldsymbol{x}^i = \boldsymbol{x}_*^i$ satisfies the equations

$$\begin{cases} \nabla_{\boldsymbol{x}^i} f(\boldsymbol{x}_*^1, \cdots, \boldsymbol{x}_*^{i-1}, \boldsymbol{x}^i, \boldsymbol{x}_*^{i+1}, \cdots, \boldsymbol{x}_*^d) = \displaystyle\sum_{j=1}^{m_i} u_j^i \nabla g_j^i(\boldsymbol{x}^i) + \sum_{j=1}^{\ell_i} v_j^i \nabla h_j^i(\boldsymbol{x}^i), \\ u_j^i g_j^i(\boldsymbol{x}^i) = 0,\ u_j^i \geq 0,\ j = 1, 2, \ldots, m_i, \\ \boldsymbol{x}^i \in S^i, \end{cases}$$

where u_j^i is the Lagrangian multiplier corresponding to the inequality constraint $g_j^i(\boldsymbol{x}^i) \leq 0$ for $j = 1, 2, ..., m_i$, and v_j^i is the Lagrangian multiplier corresponding to the equality constraint $h_j^i(\boldsymbol{x}^i) = 0$ for $j = 1, 2, ..., \ell_i$. Therefore, $(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) = (\boldsymbol{x}_*^1, \boldsymbol{x}_*^2, \cdots, \boldsymbol{x}_*^d)$ is a solution for

$$\begin{cases} \nabla_{\boldsymbol{x}^i} f(\boldsymbol{x}^1, \cdots, \boldsymbol{x}^{i-1}, \boldsymbol{x}^i, \boldsymbol{x}^{i+1}, \cdots, \boldsymbol{x}^d) = \displaystyle\sum_{j=1}^{m_i} u_j^i \nabla g_j^i(\boldsymbol{x}^i) + \sum_{j=1}^{\ell_i} v_j^i \nabla h_j^i(\boldsymbol{x}^i), i = 1, 2, \ldots, d, \\ u_j^i g_j^i(\boldsymbol{x}^i) = 0,\ u_j^i \geq 0,\ j = 1, 2, \ldots, m_i,\ i = 1, 2, \ldots, d, \\ \boldsymbol{x}^i \in S^i,\ i = 1, 2, \ldots, d, \end{cases}$$

[1] The most widely used CQs include the Slater condition, the linear independence constraint qualification, the Mangasarian–Fromowitz constraint qualification, the constant rank constraint qualification, the constant positive linear dependence constraint qualification, and the quasi-normality constraint qualification. For details see a textbook on nonlinear programming, e.g., Bertsekas [12].

which is exactly the KKT system for (G). Therefore, $(\boldsymbol{x}_*^1, \boldsymbol{x}_*^2, \cdots, \boldsymbol{x}_*^d)$ must be a KKT point of (G) as well. □

Remark that since not all KKT points are stationary, Theorem 2.2.1 is in fact a stronger statement; however, Corollary 2.2.2 is convenient to use in many applications.

Due to the generality of model (G) and the nice convergence property of MBI method, this method should be encouraged. In the following chapters, we will consider some particular constraint sets S^i, and see how MBI method can be used to tackle problems in practice. Besides, the efficiency of MBI method will be tested.

Chapter 3

Co-Clustering of Gene Expression Data

3.1 Introduction

This chapter presents one direct application of the MBI method in a bioinformatics application. We will actually use the MBI method to tackle large-scale high-dimensional genome-wide gene expression data. Before we present a mathematical model for analyzing gene expression data, let us first briefly review the background on clustering and co-clustering.

With the development of high-throughput gene expression technology, it is possible to measure expressions of thousands of genes simultaneously. In order to exploit the inherent structure of genes, there is a strong need to develop analytical methodology to analyze the information embedded in gene expression data. Due to the large number of genes and the complexity of biological networks, clustering technique has been developed as a useful exploratory tool for analysis of gene expression data. The goal of clustering is to subdivide a set of items (in our context, genes) in such a way that

similar items fall into the same cluster, whereas dissimilar items belong to different clusters.

Clustering has been investigated by various areas of researchers, ranging from computer science (e.g., web mining, image segmentation), engineering (e.g., machine learning, pattern recognition, mechanical engineering), life and medical science (e.g., biology, genetics, pathology) to social science (e.g., psychology, archeology), and even economics (e.g., business and marketing); see books [28, 36]. For classical clustering technique in gene expression analysis, D′haeseleer [25] discussed two classes of clustering, i.e., hierarchical clustering and partitioning, and three popular clustering methods, i.e., Eisen hierarchical clustering [26], k-means [104] and self organizing map (SOM) method [103]. For more information about clustering, one is referred to [28, 36] and the references therein.

Cheng and Church [22] first introduced the concept of co-clustering for two dimensional (2D) gene expression data and developed an effective measure of the co-clusters based on the mean square residue and a greedy node-deletion algorithm. Their algorithm could cluster genes and conditions simultaneously and thus could discover the similar expression of a certain group of genes on a certain group of conditions and vice versa. Later many different co-clustering algorithms were developed. For example, the authors in [23] formulated the objective functions based on minimizing two measures of squared residue that are similar to those used by Cheng and Church [22] and Hartigan [35]. Their iterative algorithm could directly minimize the squared residues and find $k \times l$ co-clusters simultaneously as opposed to finding a single co-cluster at a time like Cheng and Church. For the ideas of other co-clustering algorithms, we refer to the references [79, 50, 23].

In this chapter, we are going to propose a new framework to study the co-clustering of gene expression data. This new framework is based on a generic tensor optimization

model and the MBI method presented in Chapter 2. This framework not only can be used for co-clustering of gene expression data with genes expressed at different conditions (genes × conditions) represented in 2D matrices, but also it can be readily applied for co-clustering of gene expression data in 3D, 4D, 5D with genes expressed at different tissues, different development stages, different time points, different stimulations, and so on and so forth (e.g., genes×tissues×development stages×time points×stimulations) and even more complex high-dimensional matrices. Moreover, this framework is flexible enough to incorporate different objective functions. We demonstrate this new framework by providing the details of the algorithm for one model with one specific objective function under the framework, the implementation of the algorithm and the experimental testing on microarray gene expression datasets. Our algorithm turns out to be very efficient (which runs for only a few minutes on a regular PC for large gene expression datasets) and performs well for identifying patterns in microarray data sets compared with other approaches. It is worth to mention a recent paper by Zhang *et al.* [114] here, they introduce a new notion of "co-identification", and build a computational framework of co-identification that enables clustering to be multi-dimensional and adaptive based on the MBI method.

We organize the following sections as follows. In Section 3.2, we present a new generic co-clustering framework based on the MBI method for analyzing gene expression data. In order to express our idea more explicitly, we describe one specific 2D gene expression co-clustering model and the detailed implementation method in Section 3.3. Finally, numerical performance of the proposed method on gene expression datasets will be reported in Section 3.4.

3.2 A New Generic Framework for Co-Clustering Gene Expression Data

In this section we first present our model for the co-clustering problem based on tensor optimization and then give a generic algorithm for high-dimensional gene expression data co-clustering.

3.2.1 Tensor Optimization Model of The Co-Clustering Problem

We begin with the modeling of gene expression co-clustering problem. Here, we use tensors to describe gene expression data in high dimensions. Suppose that d-dimensional tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is a set of given gene expression data. Let $I_j = \{1, 2, \cdots, n_j\}$ be the set of indices on the j-th dimension, $j = 1, 2, \ldots, d$. We wish to find a p_j-partition of the index set I_j, say $I_j = I_1^j \bigcup I_2^j \bigcup \cdots \bigcup I_{p_j}^j$, where $j = 1, 2, \ldots, d$, in such a way that each of the sub-tensor $\mathcal{A}_{I_{i-1}^1 \times I_{i_2}^2 \times \cdots \times I_{i_d}^d}$ is as tightly packed up as possible, where $1 \leq i_j \leq n_j$ and $j = 1, 2, \ldots, d$.

Suppose that $\mathcal{X} \in \mathbb{R}^{p_1 \times p_2 \times \cdots \times p_d}$ is the tensor for the co-cluster values, then the component $\mathcal{X}_{j_1, j_2, \cdots, j_{i-1}, j_i, j_{i+1}, \cdots, j_d}$ is the value of the co-cluster $(j_1, j_2, \cdots, j_{i-1}, j_i, j_{i+1}, \cdots, j_d)$ with $1 \leq j_i \leq p_i, i = 1, 2, \ldots, d$. Next, we define a row-to-column assignment matrix $Y^j \in \mathbb{R}^{n_j \times p_j}$ for the indices for the j-th array of tensor $\mathcal{A}$, with

$$Y_{ik}^j = \begin{cases} 1, & \text{if } i \text{ is assigned to the } k\text{-th partition } I_k^j; \\ 0, & \text{otherwise.} \end{cases}$$

Then, we introduce a proximity measure $f(s) : \mathbb{R} \to \mathbb{R}_+$, with the property that $f(s) \geq 0$ for all $s \in \mathbb{R}$ and $f(s) = 0$ if and only if $s = 0$. The co-clustering problem can

be formulated as

$$\begin{aligned}(CC)\quad \min\quad & \sum_{j_1,j_2,\cdots,j_d} f(\mathcal{A}_{j_1j_2\cdots j_d} - (\mathcal{X}\times_1 Y^1\times_2 Y^2\times_3\cdots\times_d Y^d)_{j_1j_2\cdots j_d})\\ \text{s.t.}\quad & \mathcal{X}\in\mathbb{R}^{p_1\times p_2\times\cdots\times p_d},\\ & Y^j\in\mathbb{R}^{n_j\times p_j} \text{ is an assignment matrix}, j=1,2,\ldots,d.\end{aligned}$$

A variety of proximity measures could be considered. For instance, if $f(s)=s^2$, then (CC) can be written as

$$\begin{aligned}(P_1)\quad \min\quad & \|\mathcal{A}-\mathcal{X}\times_1 Y^{(1)}\times_2 Y^{(2)}\times_3\cdots\times_d Y^{(d)}\|\\ \text{s.t.}\quad & \mathcal{X}\in\mathbb{R}^{p_1\times p_2\times\cdots\times p_d},\\ & Y^{(j)}\in\mathbb{R}^{n_j\times p_j} \text{ is an assignment matrix}, j=1,2,\ldots,d.\end{aligned}$$

If $f(s)=|s|$, then (CC) can be written as

$$\begin{aligned}(P_2)\quad \min\quad & \sum_{j_1=1}^{n_1}\sum_{j_2=1}^{n_2}\cdots\sum_{j_d=1}^{n_d}|\mathcal{A}_{j_1j_2\cdots j_d} - (\mathcal{X}\times_1 Y^{(1)}\times_2 Y^{(2)}\times_3\cdots\times_d Y^{(d)})_{j_1j_2\cdots j_d}|\\ \text{s.t.}\quad & \mathcal{X}\in\mathbb{R}^{p_1\times p_2\times\cdots\times p_d},\\ & Y^{(j)}\in\mathbb{R}^{n_j\times p_j} \text{ is an assignment matrix}, j=1,2,\ldots,d.\end{aligned}$$

A third possible formulation can be

$$\begin{aligned}(P_3)\quad \min\quad & \max_{1\le j_i\le n_i; i=1,2,\ldots,d}|\mathcal{A}_{j_1j_2\cdots j_d} - (\mathcal{X}\times_1 Y^{(1)}\times_2 Y^{(2)}\times_3\cdots\times_d Y^{(d)})_{j_1j_2\cdots j_d}|\\ \text{s.t.}\quad & \mathcal{X}\in\mathbb{R}^{p_1\times p_2\times\cdots\times p_d},\\ & Y^{(j)}\in\mathbb{R}^{n_j\times p_j} \text{ is an assignment matrix}, j=1,2,\ldots,d.\end{aligned}$$

3.2.2 The MBI Method for Co-Clustering Problem

Notice that model (CC) is in the format of (G), and all the block variables in the constraints of model (CC) are separable, which enables the application of the MBI method. Clearly, we can transform the minimization problem (CC) into a maximization problem simply by letting the objective function $f:=-f$. Without loss of generality, we still use notation f. Our generic algorithm for the co-clustering maximization form based on the MBI method can be formulated as follows.

Algorithm CC. Generic co-clustering algorithm.

- **Input**: A set of data $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, which is a d-dimensional tensor; parameters k_1, $k_2, \cdots, k_d$, which are all positive integers and $0 < k_i \leq n_i$, for $i = 1, 2, \ldots, d$.

- **Output**: $k_1 \times k_2 \times \cdots \times k_d$ co-clusters of $\mathcal{A}$.

- **Main variables**:

  A non-negative integer k as the loop counter;

  A $k_1 \times k_2 \cdots \times k_d$-tensor $\mathcal{X}$ with each entry a real number as the artificial central point of one of the co-clusters;

  A $n_i \times k_i$-matrix Y^i as the assignment matrix with $\{0, 1\}$ as the value of each entry, for $i = 1, 2, \ldots, d$.

0 Let $Y^0 := \mathcal{X}$. Choose a feasible solution $(Y_0^0, Y_0^1, Y_0^2, \cdots, Y_0^d)$ and compute the initial objective value $v_0 := f(Y_0^0, Y_0^1, Y_0^2, \cdots, Y_0^d)$. Set the loop counter $k := 0$.

1 Solve the following $d + 1$ subproblems

$$(G_0) \quad \max_{Z^0} \quad f(Z^0, Y_k^1, Y_k^2, \cdots, Y_k^d)$$

and

$$\begin{aligned} (G_i) \quad & \max \quad f(Y_k^0, Y_k^1, \cdots, Y_k^{i-1}, Z^i, Y_k^{i+1}, \cdots, Y_k^d) \\ & \text{s.t.} \quad Z^i \in \mathbb{R}^{n_j \times p_j} \text{ is an assignment matrix,} \end{aligned}$$

for each $i = 1, 2, \ldots, d$. And let

$$\begin{aligned} Z_{k+1}^i &:= \arg\max f(Y_k^0, Y_k^1, \cdots, Y_k^{i-1}, Z^i, Y_k^{i+1}, \cdots, Y_k^d), \\ w_{k+1}^i &:= f(Y_k^0, Y_k^1, \cdots, Y_k^{i-1}, Z_{k+1}^i, Y_k^{i+1}, \cdots, Y_k^d). \end{aligned}$$

2 Let $w_{k+1} := \max_{1\le i\le d} w^i_{k+1}$ and $i^* = \arg\max_{1\le i\le d} w^i_{k+1}$. Update

$$\begin{aligned} Y^i_{k+1} &:= Y^i_k,\ \forall\ i \in \{0,1,2,\ldots,d\}\backslash\{i^*\} \\ Y^{i^*}_{k+1} &:= Z^{i^*}_{k+1} \\ v_{k+1} &:= w_{k+1}. \end{aligned}$$

3 If $|v_{k+1} - v_k| < \epsilon$, go to Step 4. Otherwise, set $k := k+1$, and go to Step 1.

4 Print the $k_1 \times k_2 \times \cdots \times k_d$ co-clusters of $\mathcal{A}$ according to the assignment matrices $Y^1_{k+1}, Y^2_{k+1}, \cdots, Y^d_{k+1}$.

As shown in Chapter 2, our MBI method guarantees converging to a stationary point of model (CC). For the special problems $(P_1), (P_2)$ and (P_3), Algorithm CC will automatically apply. Regarding the issue on solving subproblems in these three particular cases, they are all polynomial-time solvable. For instance, for the fixed variables Y^j, $j = 1,2,\ldots,d$, the search of $\mathcal{X}$ becomes:

- In the case of (P_1), the problem is a least square problem;
- In the case of (P_2) or (P_3), the problems are linear programming.

To appreciate the computational complexity of the models under consideration, we remark that even if the block variable $\mathcal{X}$ is fixed, to search for the two joint optimal assignments of, say, Y^1 and Y^2, while all other $Y^i (i = 3,4,\ldots,d)$ are fixed, is already NP-hard.

3.3 Algorithm for Co-Clustering 2D Matrix Data

We have implemented the algorithm for co-clustering gene expression data in 2D-matrices when the (P_1) formulation is used. Given a 2D-matrix A with m rows and n

columns, which represents the gene expressions of m different genes under n different conditions. We apply our co-clustering algorithm to partition the genes and conditions at the same time to get $k_1 \times k_2$ submatrices, where k_1 is the number of partitions of the m genes and k_2 is the number of partitions of the n conditions; see the following procedure.

Algorithm for 2D-matrix co-clustering based on the (P_1) model.

- **Input**: A 2D-matrix A with m rows and n columns. Two parameters k_1 and k_2, where k_1 and k_2 are both positive integers.
- **Output**: $(k_1 \times k_2)$ co-clusters of the matrix A, where k_1 is the number of partitions of the m rows and k_2 is the number of partitions of the n columns.
- **Main Variables**:

  A non-negative integer k as the loop counter;

  A $k_1 \times k_2$ matrix X with each entry a real number as the artificial central point of one of the $k_1 \times k_2$ co-clusters of the matrix A;

  A $m \times k_1$ matrix Y^1 as the row assignment matrix with $\{0, 1\}$ as the value of each entry; and

  A $n \times k_2$ matrix Y^2 as the column assignment matrix with $\{0, 1\}$ as the value of each entry.

0 Set the loop counter $k := 0$.

Randomly set the initial values of the three matrices X_k, Y_k^1 and Y_k^2, and compute the initial objective value $v_0 := \max - \left\| A - X_k \times_1 Y_k^1 \times_2 Y_k^2 \right\|$.

1 Fixed matrices X_k and Y_k^1, get the optimal column assignment matrix Y^2 and compute the objective value $v_{Y^2} := \max - \left\| A - X_k \times_1 Y_k^1 \times_2 Y^2 \right\|$;

Fixed matrices X_k and Y_k^2, get the optimal row assignment matrix Y^1 and compute the objective value $v_{Y^1} := \max - \left\|A - X_k \times_1 Y^1 \times_2 Y_k^2\right\|$;

Fixed matrices Y_k^1 and Y_k^2, get the optimal matrix X and compute the objective value $v_X := \max - \left\|A - X \times_1 Y_k^1 \times_2 Y_k^2\right\|$.

2 Compute $v_{k+1} := \max\{v_{Y^2}, v_{Y^1}, v_X\}$;

If $v_{k+1} = v_{Y^2}$ then update: $X_{k+1} = X_k$, $Y_{k+1}^1 = Y_k^1$, and $Y_{k+1}^2 = Y^2$;

If $v_{k+1} = v_{Y^1}$ then update: $X_{k+1} = X_k$, $Y_{k+1}^1 = Y^1$, and $Y_{k+1}^2 = Y_k^2$;

If $v_{k+1} = v_X$ then update: $X_{k+1} = X$, $Y_{k+1}^1 = Y_k^1$, and $Y_{k+1}^2 = Y_k^2$;

3 If $|v_{k+1} - v_k| < \epsilon$, go to Step 4. Otherwise, set $k := k + 1$, and go to Step 1.

4 Print the $k_1 \times k_2$ co-clusters of A according to the assignment matrices Y_{k+1}^1, Y_{k+1}^2.

3.4 Numerical Experiments

We use two microarray datasets to test our algorithm and make comparisons with other clustering and co-clustering methods. The first dataset is the gene expression of a yeast cell cycle dataset with 2884 genes and 17 conditions, where the expression values are in the range 0 to 595. The second dataset is the gene expression of a human B-cell lymphoma dataset with 4026 genes and 96 conditions, where the values are in the range -749 and 642. The detailed information about the datasets could be found in Cheng and Church [22], Tavazoie *et al.* [104] and Alizadeh *et al.* [3].

3.4.1 Implementation Details and Some Discussions

Our algorithm is implemented using C++. The experimental testing is performed on a regular PC (processor: Pentium dual-core CPU, T4200 @ 2.00GHz 2.00GHz; memory:

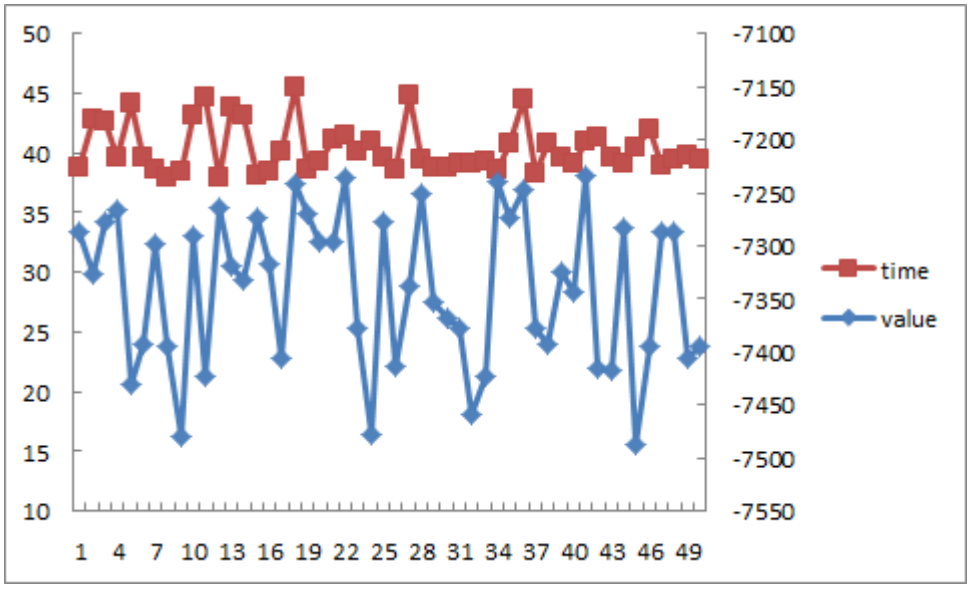

Figure 3.1: The final objective function values (the right axis) and the running time (the left axis, in seconds) of 50 runs of our algorithm with random initial values of the three matrices X, Y^1 and Y^2 on the yeast dataset to generate 30×3 co-clusters.

3GB; operating system: 64-bit windows 7; compiler: Microsoft visual C++ 2010). The figures are generated using MATLAB R2010a.

We tested our algorithm using different initial values of the three matrices X, Y^1 and Y^2. The setup of the initial values of the three matrices includes using random values for the three matrices, using subsets of values in A to initialize X, limiting the number of 1s to be one in each row of matrices Y^1 and Y^2, and using the values of the matrices Y^1 and Y^2 to calculate the values of the matrix X. We found out that the initial values of the three matrices will not significantly affect the convergence of our algorithm; see Figure 3.1 for the final objective function values and the running times over 50 runs for the yeast dataset to generate 30×3 co-clusters.

We also tested our algorithm for different numbers of partitions of the rows and the columns, that is, different values of k_1 and k_2. For example, when $k_1 = 30$ and

$k_2 = 3$, our program generates the co-clusters of the yeast cell dataset in 40.252 seconds with the final objective function value -7386.75, and when $k_1 = 100$ and $k_2 = 5$, our program generates the co-clusters of the yeast cell dataset in 90.138 seconds with the final objective function value -6737.86. The running time of our algorithm is comparable to the running time of the algorithms developed in [23].

Refer to Figure 3.2 for the objective function value versus iteration of our algorithm on the yeast cell dataset and the human lymphoma dataset. The average initial and final objective function values over 20 runs for the yeast dataset to generate 30×3 co-clusters are -25818.1 and -7323.42. The average initial and final objective function values over 20 runs for the human lymphoma dataset to generate 150×7 co-clusters are -143958 and -119766. There are 100 iterations of our implemented algorithm. We can see that our algorithm converges rapidly.

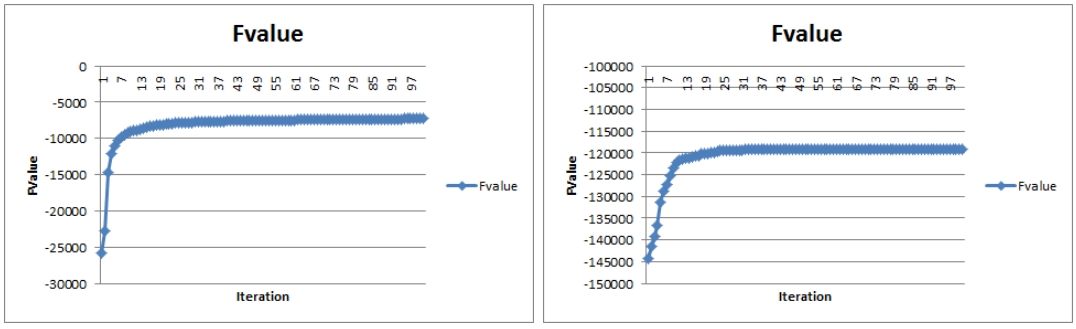

Figure 3.2: The figure on the left shows the objective function value vs. iteration of our algorithm on the yeast dataset to generate 30×3 co-clusters. The figure on the right shows the objective function value vs. iteration of our algorithm on the human dataset to generate 150×7 co-clusters.

3.4.2 Testing Results using Microarray Datasets

In the following we present some exemplary co-clusters identified by our algorithm. We compare the co-clusters with those identified by other approaches. For all the figures presented here, the x-axis represents the different number of conditions and the y-axis represents the values of the gene expression level.

Figure 3.3 shows four co-clusters of the yeast cell dataset generated when the two parameters $k_1 = 20$ and $k_2 = 3$. In Figure 3.3, the two co-clusters in the same row contain the same sets of genes but in two different sets of conditions, and the two co-clusters in the same column show two different groups of genes on the same set of conditions. Each of the four co-clusters from top-left to bottom-right has the following (number of genes, [list of conditions]) respectively (148, [condition 0, 1, 5, 8, 11, 12]), (148, [condition 2, 3, 4, 6, 7]), (292, [condition 0, 1, 5, 8, 11, 12]), and (292, [condition 2, 3, 4, 6, 7]).

We can see from the co-clusters shown in Figures 3.3, 3.4 and other generated co-clusters that our algorithm can effectively identify groups of genes and groups of conditions that exhibit similar expression patterns. It can discover the same subset of genes that have different expression levels over different subsets of conditions, and can also discover different subsets of genes that have different expression levels over the same subset of conditions.

The four co-clusters in Figure 3.3 are closely related to the clusters of Tavazoie *et al.* [104], where the classical k-means clustering algorithm was applied and the yeast cell cycle gene expression dataset was clustered into 30 clusters. The bottom two co-clusters are mainly related to their clusters 2, 3, 4, 6 and 18. The top two co-clusters are mainly related to their cluster 1. This shows that the same group of genes have different expression patterns over different subsets of conditions. This also shows that

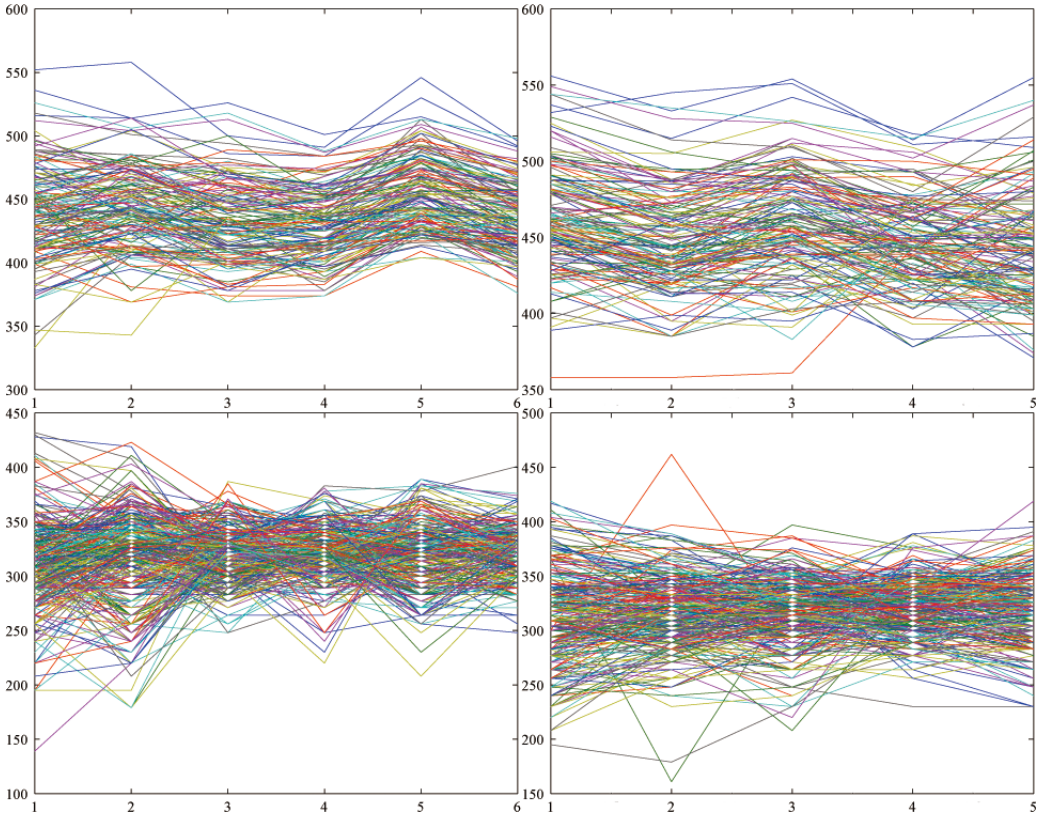

Figure 3.3: Four co-clusters of the yeast cell dataset generated when the two parameters $k_1 = 20$ and $k_2 = 3$.

one or more than one co-clusters could correspond to one cluster of Tavazoie *et al.* [104].

We use the mean square residue score developed in [22] to evaluate the co-clusters generated by our algorithm. We identify 12 co-clusters with the best mean square residue scores of the yeast cell dataset when $k_1 = 30$ and $k_2 = 3$. The list of the scores are 168.05, 182.04, 215.69, 335.72, 365.01, 378.37, 408.98, 410.03, 413.08, 416.63, 420.37, and 421.49. All the 12 co-clusters have the mean square residue scores less than 450. They are meaningful co-clusters.

We conduct similar experimental testing on the human lymphoma dataset. Figure 3.4 shows four exemplary co-clusters of the dataset generated when the two parameters $k_1 = 150$ and $k_2 = 7$. In Figure 3.4, the two co-clusters in the same row contain

the same sets of genes but in different sets of conditions, and the two co-clusters in the same column show two different groups of genes on the same set of conditions. Each of the four co-clusters has the following (number of genes, number of conditions): (57, 9), (57,45), (27,9), and (27,45).

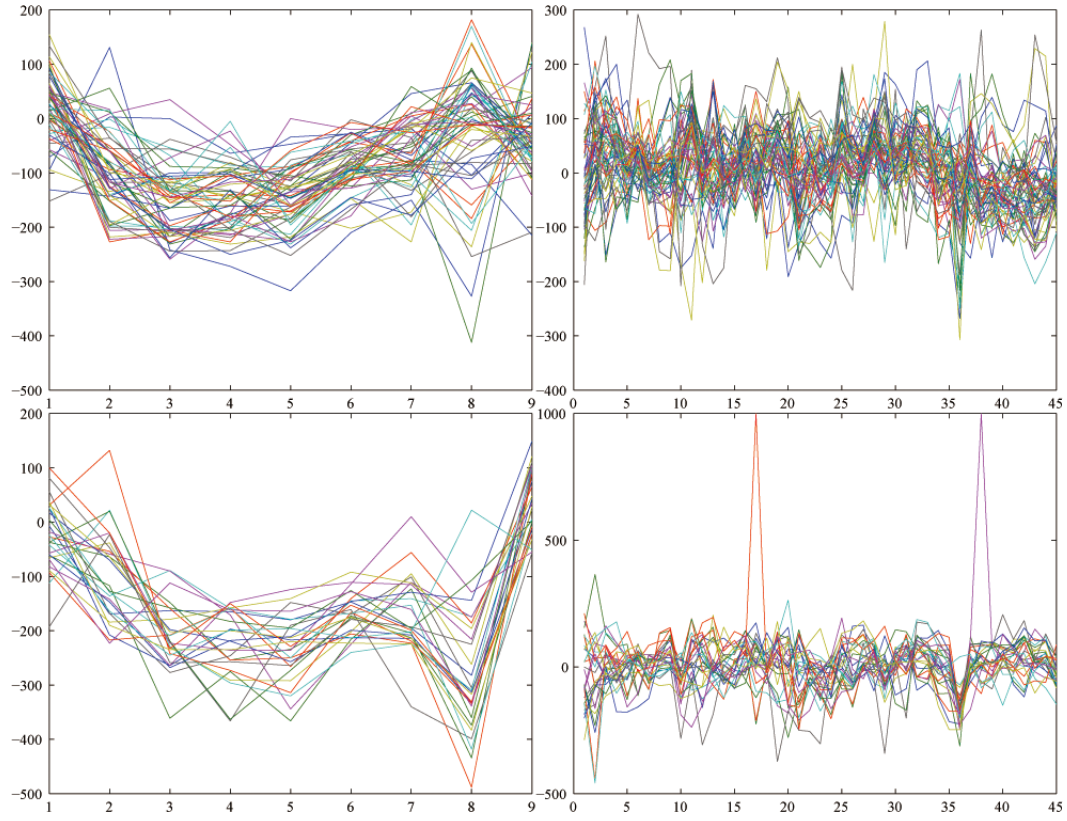

Figure 3.4: Four co-clusters of human cell dataset generated when the two parameters $k_1 = 150$ and $k_2 = 7$.

3.4.3 Testing Results using 3D Synthesis Dataset

We test our algorithm using the 3D synthetic dataset from [101] which has six files with each file containing 1,000 genes measured over 10 conditions with 6 time-points for each condition. The co-clusters in Figure 3.5 show clear coherent patterns of the 3D dataset. In Figure 3.5, each curve corresponds to the expression of one gene. The

x-axis represents the different number of time points with every 6 time-points in one condition, while the y-axis represents the values of the gene expression level.

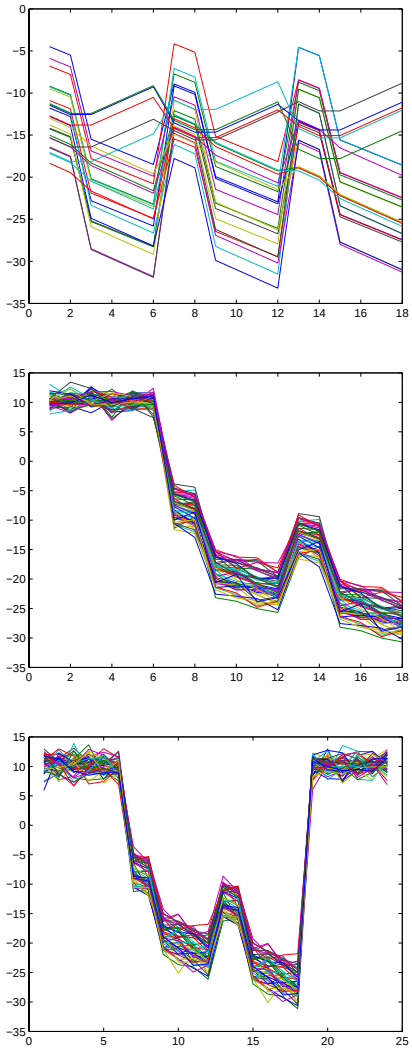

Figure 3.5: Co-clusters of the 3D dataset generated when the three parameters $k_1 = 10$, $k_2 = 1$ and $k_3 = 3$.

Chapter 4

Polynomial Optimization with Spherical Constraint

4.1 Introduction

In this chapter, we shall study some particular and practical polynomial optimization problems. The objective functions include multilinear tensor functions, homogeneous polynomials (or forms) and general inhomogeneous polynomials; see functions (1.1), (1.2), and (1.3) defined in Section 1.2.3. Following the notations there, throughout this chapter we use F to denote a multilinear function defined by a tensor form $\mathcal{F}$, f for a homogeneous polynomial function, and p for an inhomogeneous polynomial function. For the constraint sets of polynomial optimization are typically homogeneous quadratic polynomial equalities or inequalities. In particular, we shall pay special attention to the models, which include maximizing a homogeneous form over the Euclidean ball constraint

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

and maximizing a multilinear tensor form under spherical constraint

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

Also, we consider a general model where an inhomogeneous polynomial function is maximized over the intersection of co-centered ellipsoids

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

where matrices $Q_j \succeq 0$ for $j = 1, 2, ..., m$, and $\sum_{j=1}^m Q_j \succ 0$.

Actually, the above three polynomial optimization models have been investigated by He *et al.* [40, 41], and they designed approximation algorithms for the three models (cf. [40, 41]). It is worth mentioning that problem (T) can be viewed as a relaxation of (H), which played a crucial role in the approximation algorithms for solving (H) in He *et al.* [40]. One of the main contributions of this chapter is to reveal an intrinsic relationship between the optimal solutions of (T) and (H). As we shall see later, (H) has applications in magnetic resonance imaging (MRI), the best rank-one approximation of the super-symmetric tensor $\mathcal{F}$, and the problem of finding the largest eigenvalue of the tensor $\mathcal{F}$; see, e.g., [29, 59, 91, 92]. The work on designing an efficient algorithm for (H) becomes much more important. Before proceeding, let us review the existing popular algorithms for solving spherically constrained homogeneous polynomial optimization problems.

As we know, spherically constrained homogeneous polynomial optimization models have received some recent research attention, theoretically as well as numerically. One direct approach is to apply the method of Lagrange multipliers to reach a set of multivariate polynomial equations, namely, the Karush-Kuhn-Tucker (KKT) system, which provides the necessary conditions for optimality; see, e.g., [29, 51, 116]. In [29], one

strives to enumerate *all* the solutions of a KKT system, not only the global optimum, as all the KKT solutions will be meaningful in this application. Indeed, the authors develop special algorithms for that purpose, e.g., the subdivision methods proposed by Mourrain and Pavone [80] and the generalized normal forms algorithms designed by Mourrain and Trébuchet [81]. However, the shortcomings of these methods are apparent if the degree of the polynomial is high. An important application of spherically constrained homogeneous polynomial optimizations is the best rank-one approximation of supersymmetric tensors. As mentioned before, the main workhorse for solving it is the ALS method. However, the ALS method is not guaranteed to converge to a global minimum or a stationary point, only to a solution where the objective function ceases to decrease. There are numerous extensions of the ALS method (e.g., incorporating a line-search procedure in the ALS procedure [86, 96]). Along a related line, De Lathauwer *et al.* [69] proposed the higher-order power method (HOPM) on rank-one approximation of higher-order tensors, which can also be viewed as an extension of the ALS method. Following up on that approach, Kofidis and Regalia [59] devised the symmetric higher-order power method (S-HOPM) to rank-one approximation of super-symmetric tensors and proved its convergence for super-symmetric tensors whenever their corresponding polynomial forms have convexity or concavity. Furthermore, Wang and Qi [111] proposed a greedy method, which iteratively computes the best super-symmetric rank-one approximation of the residual tensors in order to obtain a successive super-symmetric rank-one decomposition. Those methods have nice properties; however, they all fail to guarantee convergence for the best rank-one approximation of a tensor, whether the tensor is super-symmetric or not. Another entirely different but very interesting approach, known as the Z-eigenvalue method, was proposed by Qi *et al.* [95]. This heuristic cross-hill Z-eigenvalue method aims to solve homogeneous polynomial functions with degree at most three.

In this setting, we can use the MBI method to handle spherically constrained homogeneous polynomial optimization problem, and guarantee that the algorithm converges to a stationary point, which is typically also global optimal in our numerical experiences. The proposed MBI approach can naturally be regarded as a local improvement scheme for polynomial optimization, to start from any good initial solutions. Therefore, the new MBI method can be used in combination with any approximation algorithms (such as Khot and Naor [55] and He *et al.* [40, 41, 42]) to achieve excellent performance in practice while enjoying the theoretical worst case performance guarantees.

The remainder of this chapter is organized as follows. We establish a generic equivalence result between the homogeneous polynomial optimization and its tensor relaxation problem in Section 4.2. This will enable the application of the MBI method to solve the polynomial optimization model. In Section 4.3, we present the implementation details for solving spherically constrained homogeneous polynomial optimization. Our numerical experiments by using randomly generated data is reported in Section 4.4. Finally, in Section 4.5, two real applications of homogeneous polynomial optimization are provided: one is magnetic resonance imaging (MRI), the other is the best rank-one approximation of the super-symmetric tensor.

4.2 Generalized Equivalence Result

In He *et al.* [40], problem (T) is regarded as a relaxation of (H), and an approximate solution for (T) is used to construct an approximate solution for (H). As we shall see later, these two problems are actually *equivalent*. In fact, we can prove the following general result.

Theorem 4.2.1. *Suppose that $\mathcal{F} \in \mathbb{R}^{n^d}$ is a dth-order super-symmetric tensor with F being its corresponding multilinear function. Let $\mathcal{G}_i \in \mathbb{R}^{m^t}$ be a tth-order super-*

symmetric tensor, with G_i being its corresponding multilinear function, $i = 1, 2, ..., n$. Consider a mapping $g : \mathbb{R}^m \mapsto \mathbb{R}^n$ where the i-th component of g is given by $g_i(\boldsymbol{x}) = G_i(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x}}_{t})$, $i = 1, 2, ..., n$. If the image set $g(\mathbb{R}^m) \subseteq \mathbb{R}^n$ is a linear subspace of $\mathbb{R}^n$, then

$$\max_{\|g(\boldsymbol{x})\|=1} |F(\underbrace{g(\boldsymbol{x}), g(\boldsymbol{x}), \cdots, g(\boldsymbol{x})}_{d})| = \max_{\|g(\boldsymbol{x}^i)\|=1,\, i=1,2,...,d} |F(g(\boldsymbol{x}^1), g(\boldsymbol{x}^2), \cdots, g(\boldsymbol{x}^d))|.$$

Proof. Denote the linear subspace $g(\mathbb{R}^m)$ to be $K \subseteq \mathbb{R}^n$. It is clear that the two optimization problems in Theorem 4.2.1 are equivalent to

$$\begin{aligned} (H_d) \quad \max \quad & |F(\underbrace{\boldsymbol{y}, \boldsymbol{y}, \cdots, \boldsymbol{y}}_{d})| \\ \text{s.t.} \quad & \|\boldsymbol{y}\| = 1,\ \boldsymbol{y} \in K \end{aligned}$$

and

$$\begin{aligned} (T_d) \quad \max \quad & |F(\boldsymbol{y}^1, \boldsymbol{y}^2, \cdots, \boldsymbol{y}^d)| \\ \text{s.t.} \quad & \|\boldsymbol{y}^i\| = 1,\ \boldsymbol{y}^i \in K,\ i = 1, 2, \ldots, d, \end{aligned}$$

respectively. We shall aim to prove that $v(T_d) = v(H_d)$.

The proof is based on the induction on the order of the tensor d. It is trivially true when $d = 1$. Suppose that $v(T_d) = v(H_d)$ for d with $d \geq 1$. Then, for the case $d + 1$, denote $(\hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2, \cdots, \hat{\boldsymbol{y}}^d, \hat{\boldsymbol{y}}^{d+1})$ to be an optimal solution of (T_{d+1}). By induction, we have

$$\begin{aligned} v(T_{d+1}) &= \max_{\|\boldsymbol{y}^i\|=1,\, \boldsymbol{y}^i \in K,\, i=1,2,...,d} |F(\boldsymbol{y}^1, \boldsymbol{y}^2, \cdots, \boldsymbol{y}^d, \hat{\boldsymbol{y}}^{d+1})| \\ &= \max_{\|\boldsymbol{y}\|=1,\, \boldsymbol{y} \in K} |F(\underbrace{\boldsymbol{y}, \boldsymbol{y}, \cdots, \boldsymbol{y}}_{d}, \hat{\boldsymbol{y}}^{d+1})|. \end{aligned} \tag{4.1}$$

Denote S to be the set of all optimal solutions of (T_{d+1}) with support 1 or 2, i.e., the number of distinctive vectors of $\{\boldsymbol{y}^1, \boldsymbol{y}^2, \cdots, \boldsymbol{y}^d, \boldsymbol{y}^{d+1}\}$ is less than or equal to 2. From (4.1), we know that S is nonempty. By the continuity of F and compactness of

the feasible region of $\boldsymbol{y}^i$ for $i = 1, 2, \ldots, d$, it is not hard to verify that S is compact. Now consider the following optimization problem:

$$(A) \qquad \max_{(\boldsymbol{y},\boldsymbol{y},\cdots,\boldsymbol{y};\,\boldsymbol{z},\boldsymbol{z},\cdots,\boldsymbol{z})\in S} \boldsymbol{y}^{\mathrm{T}}\boldsymbol{z}.$$

If the optimal value $v(A) < 1$, then let one of its optimal solution be $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{y}}, \cdots, \hat{\boldsymbol{y}}; \hat{\boldsymbol{z}}, \hat{\boldsymbol{z}}, \cdots, \hat{\boldsymbol{z}})$. Clearly, $\hat{\boldsymbol{y}} \neq \pm\hat{\boldsymbol{z}}$, because otherwise $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{y}}, \cdots, \hat{\boldsymbol{y}}) \in S$ would have $v(A) = 1$, a contradiction to $v(A) < 1$. Now denote $\hat{\boldsymbol{w}} = (\hat{\boldsymbol{y}}+\hat{\boldsymbol{z}})/\|\hat{\boldsymbol{y}}+\hat{\boldsymbol{z}}\|$. Since $\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}} \in K$, $\hat{\boldsymbol{y}} \neq \pm\hat{\boldsymbol{z}}$, and K is a linear subspace of $\mathbb{R}^n$, we shall have $\|\hat{\boldsymbol{w}}\| = 1$ and $\hat{\boldsymbol{w}} \in \mathrm{span}(\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}) \subset K$.

Without loss of generality, we may let $F(\hat{\boldsymbol{y}}, \hat{\boldsymbol{y}}, \cdots, \hat{\boldsymbol{y}}; \hat{\boldsymbol{z}}, \hat{\boldsymbol{z}}, \cdots, \hat{\boldsymbol{z}}) = v(T_{d+1})$. (Otherwise use $-\mathcal{F}$ instead of $\mathcal{F}$.) Since $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{y}}, \cdots, \hat{\boldsymbol{y}}; \hat{\boldsymbol{z}}, \hat{\boldsymbol{z}}, \cdots, \hat{\boldsymbol{z}})$ is an optimal solution for (T_{d+1}) and $\mathrm{span}(\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}) \subset K$, it is easy to show that $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{y}}, \cdots, \hat{\boldsymbol{y}}; \hat{\boldsymbol{w}}, \hat{\boldsymbol{w}}; \hat{\boldsymbol{z}}, \hat{\boldsymbol{z}}, \cdots, \hat{\boldsymbol{z}})$ (namely, replacing the middle $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ by $(\hat{\boldsymbol{w}}, \hat{\boldsymbol{w}})$) is also optimal for (T_{d+1}). Apply this replacement procedures until either $\hat{\boldsymbol{y}}$ or $\hat{\boldsymbol{z}}$ exhausts, while keeping the optimality for (T_{d+1}). Without loss of generality, we may come to an optimal solution in a form of $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{y}}, \cdots, \hat{\boldsymbol{y}}; \hat{\boldsymbol{w}}, \hat{\boldsymbol{w}}, \cdots, \hat{\boldsymbol{w}}) \in S$.

Let $\cos\theta = v(A)$ for some $\theta \in (0, \pi]$. Now we shall have

$$\hat{\boldsymbol{w}}^{\mathrm{T}}\hat{\boldsymbol{y}} = \cos(\theta/2) > \cos\theta = \hat{\boldsymbol{y}}^{\mathrm{T}}\hat{\boldsymbol{z}} = v(A),$$

which contradicts the optimality of $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{y}}, \cdots, \hat{\boldsymbol{y}}; \hat{\boldsymbol{z}}, \hat{\boldsymbol{z}}, \cdots, \hat{\boldsymbol{z}})$ for (A). Thus $v(A)$ must be 1, implying that (A) has a solution with support 1, which proves $v(H_{d+1}) = v(T_{d+1})$.

□

One may be led to the question: are there interesting cases where $g(\mathbb{R}^m)$ is a subspace? The answer is yes, and the most obvious case is to let $t = 1$ and $g(\boldsymbol{x}) = G\boldsymbol{x}$ with $G \in \mathbb{R}^{n\times m}$, and then Theorem 4.2.1 leads us to the next corollary.

Corollary 4.2.2. *If $\mathcal{F} \in \mathbb{R}^{m^d}$ is a dth-order super-symmetric tensor with F being its*

corresponding multilinear function, then

$$\max_{\|G\boldsymbol{x}\|=1} |F(\underbrace{\boldsymbol{x},\boldsymbol{x},\cdots,\boldsymbol{x}}_{d})| = \max_{\|G\boldsymbol{x}^i\|=1,\, i=1,2,\ldots,d} |F(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^d)|.$$

In our particular context, our models (H) and (T) correspond to G being the identity matrix and $m = n$. The relationship between the two models was also pointed out by Banach [9] in 1930's. This corollary connects to the so-called "generalized multilinear version of the Cauchy–Bouniakovski–Schwarz inequality" (Hiriart-Urruty [49]), which states that

> Let F be a super-symmetric multilinear tensor form of order d (≥ 3), and A be a positive semidefinite matrix. If
>
> $$|F(\boldsymbol{x},\boldsymbol{x},\cdots,\boldsymbol{x})| \leq (\boldsymbol{x}^{\mathrm{T}}A\boldsymbol{x})^{d/2} \quad \forall \boldsymbol{x} \in \mathbb{R}^n,$$
>
> then
>
> $$|F(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^d)|^2 \leq \prod_{i=1}^{d} (\boldsymbol{x}^i)^{\mathrm{T}}A\boldsymbol{x}^i \quad \forall \boldsymbol{x}^i \in \mathbb{R}^n,\ i = 1,2,\ldots,d.$$

The above inequality was shown by Lojasiewicz (see [15]), and an alternative proof can be found in Nesterov and Nemirovski [83]. Yet, it also follows from Corollary 4.2.2 by setting $A = G^{\mathrm{T}}G$.

On the other hand, Corollary 4.2.2 may not hold for other symmetric convex constraints, such as hypercube or simplex; see an example below for the case of a box.

Example 4.2.3. *Denote* $\mathcal{F}$ *a diagonal matrix* $\mathrm{Diag}\,(-1,1)$*, and the boxed constraints are* $-\boldsymbol{e} \leq \boldsymbol{x},\boldsymbol{y} \leq \boldsymbol{e}$ *with* $\boldsymbol{e} = (1,1)^{\mathrm{T}}$*. Then* $\max|F(\boldsymbol{x},\boldsymbol{y})| = \max|-x_1y_1 + x_2y_2| = 2$*, while* $\max|F(\boldsymbol{x},\boldsymbol{x})| = \max|-x_1^2 + x_2^2| = 1$*.*

Another nontrivial special case when the condition holds is when $t = 2$, $n = 4$, $m \geq 3$, $\boldsymbol{x} \in \mathbf{C}^m$ is in the complex-valued domain and $G_i(\boldsymbol{x},\boldsymbol{x})$ is *block square-free*,

i.e., the vector $\boldsymbol{x}$ can be partitioned into two parts, $\tilde{\boldsymbol{x}}$ and $\hat{\boldsymbol{x}}$, and $g_i(\boldsymbol{x}) = G_i(\boldsymbol{x}, \boldsymbol{x}) = (\tilde{\boldsymbol{x}}^{\mathrm{H}} G_i \hat{\boldsymbol{x}} + \hat{\boldsymbol{x}}^{\mathrm{H}} G_i^{\mathrm{H}} \tilde{\boldsymbol{x}})/2$, $i = 1, 2, 3, 4$. In that case, Ai, Huang, and Zhang [2] proved that the joint numerical range $g(\mathbf{C}^m)$ is a convex cone. Due to the block square-free property, it is also not pointed at any direction and hence is a subspace.

4.3 Spherically Constrained Homogeneous Polynomial Optimization

It is not hard to verify that (T) is actually equivalent to the so-called best rank-one tensor approximation problem given as

$$\begin{array}{ll} \min & \|\mathcal{F} - \lambda \cdot \boldsymbol{x}^1 \circ \boldsymbol{x}^2 \circ \cdots \circ \boldsymbol{x}^d\| \\ \text{s.t.} & \lambda \in \mathbb{R},\ \|\boldsymbol{x}^i\| = 1,\ \boldsymbol{x}^i \in \mathbb{R}^{n_i},\ i = 1, 2, \ldots, d. \end{array}$$

Indeed, the equivalence can be established by the following derivation.

$$\begin{aligned} & \|\mathcal{F} - \lambda \cdot \boldsymbol{x}^1 \circ \boldsymbol{x}^2 \circ \cdots \circ \boldsymbol{x}^d\| \\ = & \sqrt{\|\mathcal{F}\|^2 - 2\lambda\, \mathcal{F} \bullet (\boldsymbol{x}^1 \circ \boldsymbol{x}^2 \circ \cdots \circ \boldsymbol{x}^d) + \lambda^2 \|\boldsymbol{x}^1 \circ \boldsymbol{x}^2 \circ \cdots \circ \boldsymbol{x}^d\|^2} \\ = & \sqrt{\|\mathcal{F}\|^2 - 2\lambda\, F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d) + \lambda^2}, \end{aligned}$$

hence, the optimal λ should be equal to $F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d)$. Substituting the optimal λ into the above equation, then we need only to optimize

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

which is exactly the model (T). The similar derivation can be found in [73, 69, 116, 59]. Traditionally, the ALS method is a popular solution method for such models (see [59, 69]). However, the convergence of the ALS method is not guaranteed in general, as we remarked before, and the new MBI method avoids the pitfalls regarding the convergence.

In the case when the given dth-order tensor $\mathcal{F} \in \mathbb{R}^{n^d}$ is super-symmetric, then the corresponding super-symmetric rank-one approximation should be

$$\begin{aligned} \min \quad & \left\| \mathcal{F} - \lambda \cdot \underbrace{\boldsymbol{x} \circ \boldsymbol{x} \circ \cdots \circ \boldsymbol{x}}_{d} \right\| \\ \text{s.t.} \quad & \lambda \in \mathbb{R},\ \boldsymbol{x} \in \mathbb{R}^n. \end{aligned}$$

Similar to the nonsymmetric case, by imposing the vector $\boldsymbol{x}$ on the unit sphere, we can also verify that the rank-one approximation of super-symmetric tensor problem is indeed equivalent to:

$$\begin{aligned} (H) \quad \max \quad & f(\boldsymbol{x}) = F(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x}}_{d}) \\ \text{s.t.} \quad & \|\boldsymbol{x}\| = 1,\ \boldsymbol{x} \in \mathbb{R}^n, \end{aligned}$$

where F is the multilinear tensor function defined by the super-symmetric tensor form $\mathcal{F}$. In fact, the above problem (H) is also directly related to computing the maximal eigenvalue of the tensor $\mathcal{F}$; see Qi [91, 92].

The main contribution of this section is to present a new procedure, based on the MBI method, to effectively compute a KKT point for (H) via (T). The following subsections describe the detailed procedure step by step.

4.3.1 Implementing MBI on Multilinear Tensor Form

Toward eventually solving (H), let us first consider the multilinear tensor optimization (T) as follows

$$\begin{aligned} (T) \quad \max \quad & \sum_{1 \le i_1 \le n_1, 1 \le i_2 \le n_2, \cdots, 1 \le i_d \le n_d} \mathcal{F}_{i_1 i_2 \cdots i_d} x^1_{i_1} x^2_{i_2} \cdots x^d_{i_d} \\ \text{s.t.} \quad & \|\boldsymbol{x}^i\| = 1,\ \boldsymbol{x}^i \in \mathbb{R}^{n_i},\ i = 1, 2, \ldots, d, \end{aligned}$$

which is clearly a special case of (G). Moreover, Algorithm MBI is simple to implement in this case, as optimizing one block while fixing all other blocks is a trivial problem to solve. In fact, simultaneously optimizing over *two* vectors of variables, while fixing

other vectors, is also easy to implement; see [100, 113]. In particular, if d is even, then we may partition the blocks as $\{\boldsymbol{x}^1, \boldsymbol{x}^2\}, \cdots, \{\boldsymbol{x}^{d-1}, \boldsymbol{x}^d\}$, and then the subroutine reduces to an eigenvalue problem, rather than least square. (Some numerical results for the latter implementation will be presented in Section 4.4.) The flexibility in the design of the blocks is an important factor to consider in order for the MBI method to achieve its full efficiency.

4.3.2 Relationship between Homogeneous Polynomial Optimization over Spherical Constraint and Tensor Relaxation Form

As mentioned in Section 4.2, we have the nice equivalence between (H) and (T), which is a special case of Corollary 4.2.2. That is,

Corollary 4.3.1. *If $\mathcal{F} \in \mathbb{R}^{n^d}$ is a dth-order super-symmetric tensor with F being its corresponding multilinear function, then*

$$\max_{\|\boldsymbol{x}\|=1} |F(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x}}_{d})| = \max_{\|\boldsymbol{x}^i\|=1,\, i=1,2,\ldots,d} |F(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^d)|.$$

For our subsequent discussion, the main purpose is to solve (H) via (T), and so we shall focus on the application of Corollary 4.3.1. First we remark that the absolute value sign in the objective function of (T) can actually be removed, since its optimal value is always nonnegative. Similarly, if d is odd, then the absolute value sign in (H) can also be removed, due to the symmetry of the constraint set; however, for even d, this absolute value sign in (H) is necessary. Ni and Wang [85] proved that Corollary 4.3.1 holds only for a special case $d = 4$ and $n = 2$. Very recently, Qi [93] verified that Corollary 4.3.1 holds for general dimensions, but the order is strict to $d = 3$. We have showed that this property can be extended to a super-symmetric tensor for general dimensions and general order tensors, which automatically gives the positive answers to both the first two conjectures in Qi [93]. Interestingly, this result also implies that

the best super-symmetric rank-one decomposition of a super-symmetric tensor remains optimal even among all nonsymmetric rank-one tensors.

Corollary 4.3.1 establishes the equivalence between (H) and (T) for odd d, as we discussed before. For an even degree d, one may consider $\mathcal{H}$ as the dth-order super-symmetric tensor associated with the homogeneous polynomial $h(\boldsymbol{x}) := (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x})^{d/2}$, and let $f(\boldsymbol{x}) := f(\boldsymbol{x}) + \tau h(\boldsymbol{x})$, where $\tau = \|\mathcal{F}\|$. In that case, f becomes nonnegative on the sphere, and so we can again drop the absolute value sign without affecting the optimal solutions. In both cases, solving (H) can be equivalently transformed into solving (T), where the MBI method applies.

One can further generalize Corollary 4.3.1 to allow the following mixed homogeneous polynomial function (see, e.g., [42, 73]):

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

where $\boldsymbol{x}^k \in \mathbb{R}^{n_k}$ for $k = 1, 2, \ldots, s$, and the tensor form $F \in \mathbb{R}^{n_1{}^{d_1} \times n_2{}^{d_2} \times \cdots \times n_s{}^{d_s}}$ has partial symmetric property, namely, for any fixed $(\boldsymbol{x}^2, \boldsymbol{x}^3, \cdots, \boldsymbol{x}^s)$, $F(\underbrace{\cdot, \cdot, \cdots, \cdot}_{d_1}, \underbrace{\boldsymbol{x}^2, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^2}_{d_2}, \cdots, \underbrace{\boldsymbol{x}^s, \boldsymbol{x}^s, \cdots, \boldsymbol{x}^s}_{d_s})$ is a super-symmetric d_1th-order tensor form, and so on. Denote the order of the tensor F to be $d := \sum_{k=1}^{s} d_s$; then Corollary 4.3.1 immediately implies that

$$\begin{array}{lll} \max & |f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s)| & = \quad \max \quad |F(\boldsymbol{y}^1, \boldsymbol{y}^2, \cdots, \boldsymbol{y}^d)| \\ \text{s.t.} & \|\boldsymbol{x}^i\| = 1,\ i = 1, 2, \ldots, s & \quad \text{s.t.} \quad \|\boldsymbol{y}^i\| = 1,\ i = 1, 2, \ldots, d. \end{array} \tag{4.2}$$

This equation resolves and extends a conjecture in Qi [93] (with $s = 2$ and $d_1 = d_2 = 2$). Up to now, we give the positive answers to all the conjectures of Qi [93].

Let us call the left model in the above equation

$$(M) \quad \max_{\|\boldsymbol{x}^i\|=1,\, i=1,2,\ldots,s} f(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^s).$$

Clearly, (M) is a generalization of the biquadratic model considered in Ling et al. [75] (with $s = 2$ and $d_1 = d_2 = 2$) and the multiquadratic model considered by So [97] (with $d_1 = d_2 = \cdots = d_s = 2$). Equation (4.2) also suggests a method to solve (M) by resorting to its multilinear form relaxation (T), where the MBI method applies. On the other hand, model (M) can also be solved by directly adopting the MBI method, given that for any fixed $(\boldsymbol{x}^1, \cdots, \boldsymbol{x}^{i-1}, \boldsymbol{x}^{i+1}, \cdots, \boldsymbol{x}^s)$, the maximization over $\|\boldsymbol{x}^i\| = 1$ can be efficiently solved, which in this case is the model (H) with degree d_i. In particular, we can immediately apply the MBI method to solve the biquadratic model and the multiquadratic model, since the corresponding subproblem is an eigenvalue problem. We will also test our MBI method in a triquadratic case of the model (M) in Section 4.4.

4.3.3 Finding a KKT point for Homogeneous Polynomial Optimization over Spherical Constraint

Corollary 4.3.1 suggests a way to solve the homogeneous polynomial optimization model (H) by resorting to a seemingly more relaxed tensor optimization model (T). However, the equivalence is only established at optimality. Nevertheless, one may still search for a KKT solution for (H) by means of searching for a KKT solution for (T) with identical block variables. (Corollary 4.3.1 guarantees that such a special KKT point exists, and so the search is valid.) According to our computational experiences, this local search process works very well. In most cases, the KKT solution so obtained is the true global optimal solution of (H).

Let us formalize this search process as follows. We shall work with the version of (T) and (H) with an absolute sign in the objective function, like in Corollary 4.3.1. This allows us to swap the direction from $\boldsymbol{x}$ to $-\boldsymbol{x}$ without affecting its objective. As we discussed earlier, adding an absolute sign does not change the problem when d is

odd, and it also solves (H) when d is even if we modify the objective by adding a (constant) positive term, as we discussed in the previous subsection.

Algorithm KKT. Finding a KKT point for (H).

---

0 Input a KKT solution, say $(\boldsymbol{x}_0^1, \boldsymbol{x}_0^2, \cdots, \boldsymbol{x}_0^d)$, of (T) with objective value f_0. Set $k := 0$ and $(\boldsymbol{r}_0^1, \boldsymbol{r}_0^2, \cdots, \boldsymbol{r}_0^d) := (\boldsymbol{x}_0^1, \boldsymbol{x}_0^2, \cdots, \boldsymbol{x}_0^d)$.

1 If $\boldsymbol{x}_k^1 = \pm\boldsymbol{x}_k^2 = \cdots = \pm\boldsymbol{x}_k^d$, stop. Otherwise, find the closest but not identical pair among these d vectors, i.e., solve

$$\max_{1\le i<j\le d,\,(\boldsymbol{x}_k^i)^{\mathrm{T}}\boldsymbol{x}_k^j\neq 1} (\boldsymbol{x}_k^i)^{\mathrm{T}}\boldsymbol{x}_k^j.$$

Denote its optimal solution to be (i_k, j_k), and compute $\boldsymbol{z}_k := (\boldsymbol{x}_k^{i_k} + \boldsymbol{x}_k^{j_k})/\|\boldsymbol{x}_k^{i_k} + \boldsymbol{x}_k^{j_k}\|$.

2 Set $\boldsymbol{x}_{k+1}^{i_k} := \boldsymbol{z}_k$, $\boldsymbol{x}_{k+1}^{j_k} := \boldsymbol{z}_k$ and $\boldsymbol{x}_{k+1}^{i} := \boldsymbol{x}_k^{i}$ for $i \in \{1, 2, \cdots, d\}\backslash\{i_k, j_k\}$. Update the objective value of (T):

$$f_{k+1} := F(\boldsymbol{x}_{k+1}^1, \boldsymbol{x}_{k+1}^2, \cdots, \boldsymbol{x}_{k+1}^d).$$

3 If $f_{k+1} > f_k$; or if $f_{k+1} = f_k$ and there is a vector $\boldsymbol{x}^i$ $(i \in \{1, 2, \cdots, d\}\backslash\{i_k, j_k\})$ such that

$$\boldsymbol{x}^i \neq \frac{F(\boldsymbol{x}_{k+1}^1, \cdots, \boldsymbol{x}_{k+1}^{i-1}, \cdot, \boldsymbol{x}_{k+1}^{i+1}, \cdots, \boldsymbol{x}_{k+1}^d)}{\|F(\boldsymbol{x}_{k+1}^1, \cdots, \boldsymbol{x}_{k+1}^{i-1}, \cdot, \boldsymbol{x}_{k+1}^{i+1}, \cdots, \boldsymbol{x}_{k+1}^d)\|};$$

in either case, starting from $(\boldsymbol{x}_{k+1}^1, \boldsymbol{x}_{k+1}^2, \cdots, \boldsymbol{x}_{k+1}^d)$, apply Algorithm MBI to yield a KKT point $(\boldsymbol{r}_{k+1}^1, \boldsymbol{r}_{k+1}^2, \cdots, \boldsymbol{r}_{k+1}^d)$ with a larger objective value for (T). Otherwise, it is already a KKT point for (T); set $(\boldsymbol{r}_{k+1}^1, \boldsymbol{r}_{k+1}^2, \cdots, \boldsymbol{r}_{k+1}^d) := (\boldsymbol{x}_{k+1}^1, \boldsymbol{x}_{k+1}^2, \cdots, \boldsymbol{x}_{k+1}^d)$.

4 Let $k := k + 1$, and go to Step 1.

The following property of Algorithm KKT is immediate.

Proposition 4.3.2. *For Algorithm KKT, the following hold.*

1. *Each element in the sequence* $\{(\boldsymbol{r}_k^1, \boldsymbol{r}_k^2, \cdots, \boldsymbol{r}_k^d)\}$ *is a KKT point for* (T). *The sequence of the objective values* $\{f_k\}$ *for* (T) *is nondecreasing.*

2. *If* $(\boldsymbol{r}_*^1, \boldsymbol{r}_*^2, \cdots, \boldsymbol{r}_*^d)$ *is a cluster point of the sequence* $\{(\boldsymbol{r}_k^1, \boldsymbol{r}_k^2, \cdots, \boldsymbol{r}_k^d)\}$, *then* $\boldsymbol{r}_* := \boldsymbol{r}_*^1 = \pm\boldsymbol{r}_*^2 = \cdots = \pm\boldsymbol{r}_*^d$. *Moreover,* $(\boldsymbol{r}_*, \boldsymbol{r}_*, \cdots, \boldsymbol{r}_*)$ *or* $(-\boldsymbol{r}_*, -\boldsymbol{r}_*, \cdots, -\boldsymbol{r}_*)$ *is a KKT point for* (T), *and* $\boldsymbol{r}_*$ *or* $-\boldsymbol{r}_*$ *is a KKT point for* (H).

4.4 Numerical Experiments on Randomly Simulated Data

In this section, we shall present some preliminary test results for the algorithms proposed in this chapter. All the computations are conducted in an Intel Core2 Quad CPU 2.66 GHz computer with 4 GB RAM. The supporting software is MATLAB 7.8.0 (R2009a) as a platform. We use MATLAB Tensor Toolbox Version 2.4 [8] whenever tensor operations are called, and we use GloptiPoly 3 [46] for general polynomial optimization for the purpose of comparison and set the relaxation order of GloptiPoly 3 by default. To simplify our implementation, we use cvx v1.2 (Grant and Boyd [31]) as a modeling tool for our MBI subroutine. The (termination) precision for these algorithms is set to be 10^{-6}. For a given maximization problem dimension/structure, we run the algorithms on a number of random instances. GloptiPoly 3 produces an upper bound for the optimal value of that instance, which turns out to be equal to the optimal value in many cases, since the MBI method typically would find a KKT solution equal to the upper bound computed by GloptiPoly 3. We count the percentage of times when

this happens in our tests. Moreover, the MBI method is essentially a local improvement, and so it can be started from different initial solutions. Our tests are designed to see the performance of the MBI method over various settings. The following list of abbreviations refers to the results summarized in the tables to follow:

mean(P):	average ratio between solution found by MBI and upper bound by GLP;
mean(T):	average cpu seconds to solve one instance;
mean(I):	average number of iterations to solve one instance;
mean(T/I):	average cpu seconds per iteration;
dim.:	the (d, n) dimension of the test problem;
GLP:	GloptiPoly 3;
# samples:	total number of test instances;
# starts:	number of times to run MBI from random initial solutions (keep the best one);
Opt:	percentage where the MBI solutions attain the upper bound of GLP.

Throughout this section, all the data for testing problems are generated in the following manner. First, a dth-order tensor $\mathcal{F}'$ is randomly generated with its n^d entries following i.i.d. normal distribution; we then symmetrize $\mathcal{F}'$ to form a super-symmetric tensor $\mathcal{F}$. For co-centered ellipsoidal constraints, we generate an $n \times n$ matrix Q'_j $(j = 1, 2, \ldots, m)$, whose entries follow i.i.d. normal distribution, and then let $Q_j = (Q'_j)^{\mathrm{T}} Q'_j$. For comparison, we call GloptiPoly 3 to get optimal value and optimal solution if possible, or else we get an upper bound of the optimal value if GloptiPoly 3 fails to solve the given problem instance.

4.4.1 Multilinear Tensor Function over Spherical Constraints

Here, we present some numerical tests on (T). In particular, we consider

$$\begin{aligned} (E_1) \quad \max \quad & F(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{w}) = \textstyle\sum_{1\le i,j,k,l\le n} \mathcal{F}_{ijkl} x_i y_j z_k w_l \\ \text{s.t.} \quad & \|\boldsymbol{x}\| = \|\boldsymbol{y}\| = \|\boldsymbol{z}\| = \|\boldsymbol{w}\| = 1, \\ & \boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{w} \in \mathbb{R}^n, \end{aligned}$$

where tensor $\mathcal{F}$ is super-symmetric. The starting points $(\boldsymbol{x}_0,\boldsymbol{y}_0,\boldsymbol{z}_0,\boldsymbol{w}_0)$ for Algorithm MBI in our numerical experiments are all randomly generated. In our tests, we consider all the variables in the same constraint set, and dimensions are set to be $n = 2$ or $n = 3$. Here the total dimension of the test problems is chosen to be low since for our comparison we need to use GloptiPoly 3, which works only for low dimensions.

The comparison is listed in Table 4.1 for (E_1). Evidently, the results show that Algorithm MBI finds good-quality solutions very quickly. The more starts we use to run the MBI algorithm, the higher the chance we get an optimal solution. In some cases, GloptiPoly 3 is only capable of providing an upper bound; however, our MBI solution achieves these upper bounds, proving the optimality of both the GloptiPoly 3 bound and the MBI solution. Besides, a majority of our simulation results show that the KKT point $(\boldsymbol{x}^*,\boldsymbol{y}^*,\boldsymbol{z}^*,\boldsymbol{w}^*)$ of (E_1) is automatically a KKT point for the homogeneous polynomial case, namely, their block variables are identical already.

4.4.2 Tests of Another Implementation of MBI

Algorithm MBI is optimizing one block while fixing all other blocks. As mentioned in Section 4.3, simultaneously optimizing over *two* blocks of variables while fixing other blocks works under the MBI framework as well. Indeed, the similar procedures of Algorithm MBI still perform efficiently, and the convergence is guaranteed. For convenience, we call this modified procedures MBI$'$. Here, we test the performance of our methods

Table 4.1: Numerical results for (E_1) when $n = 2$ and $n = 3$

dim.	# samples	# starts	GLP	MBI		
			mean(T)	Opt	mean(T)	mean(P)
$(4, 2)$	10	1	1.5961	80%	0.1257	90.83%
		2	idem	90%	0.1352	94.62%
		3	idem	100%	0.1472	100%
$(4, 3)$	10	1	31.6348	20%	0.1735	84.03%
		2	idem	50%	0.2595	92.67%
		3	idem	60%	0.3113	93.52%
		4	idem	90%	0.3466	98.97%

MBI and MBI$'$ for (T) when $d = 6$:

$$
\begin{aligned}
(E_2) \quad \max \quad & M(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{w}, \boldsymbol{p}, \boldsymbol{q}) = \textstyle\sum_{1 \le i,j,k,l,s,t \le n} \mathcal{M}_{ijklst} x_i y_j z_k w_l p_s q_t \\
\text{s.t.} \quad & \|\boldsymbol{x}\| = \|\boldsymbol{y}\| = \|\boldsymbol{z}\| = \|\boldsymbol{w}\| = \|\boldsymbol{p}\| = \|\boldsymbol{q}\| = 1, \\
& \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{w}, \boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^n,
\end{aligned}
$$

where tensor $\mathcal{M}$ is super-symmetric. In our tests, we choose blocks $\boldsymbol{x}, \boldsymbol{y}$ as a group, blocks $\boldsymbol{z}, \boldsymbol{w}$ as another group, and blocks $\boldsymbol{p}, \boldsymbol{q}$ as the last group when implementing MBI$'$. Algorithms MBI and MBI$'$ start from the same point $(\boldsymbol{x}_0, \boldsymbol{y}_0, \boldsymbol{z}_0, \boldsymbol{w}_0, \boldsymbol{p}_0, \boldsymbol{q}_0)$, which are all randomly generated as before.

Two test sets are reported for (E_2). Table 4.2 reports the average computational time, and Table 4.3 reports the average objective value, where $(d, n) = (6, 10)$. In Table 4.3, we test 10 random instances, and each entry is the average objective value by running the corresponding algorithm 20 times. Tables 4.2 and 4.3 show that Algorithm MBI$'$ is comparable to Algorithm MBI in terms of the solution quality produced; however, Algorithm MBI$'$ requires much less computational effort on average. This means that the MBI approach is quite flexible and various innovative implementations

Table 4.2: Numerical results for (E_2) when $n = 5, 10, 15$

dim.	# samples	MBI			MBI′		
		mean(T)	mean(I)	mean(T/I)	mean(T)	mean(I)	mean(T/I)
$(6, 5)$	10	0.3087	57.3	0.0055	0.1438	20.4	0.0077
$(6, 10)$	10	4.7894	86.7	0.0551	1.1106	38.0	0.0297
$(6, 15)$	10	75.1913	127.3	0.5901	22.9004	68.3	0.3346

Table 4.3: Numerical results for (E_2) when $(d, n) = (6, 10)$

	1	2	3	4	5	6	7	8	9	10
MBI	4.3856	4.6422	4.8539	4.6369	4.6196	4.2168	4.5176	4.6628	4.5077	4.3039
MBI′	4.2235	4.8136	4.7079	4.5767	4.6906	4.4538	4.3806	4.7177	4.2873	4.3228

are possible, and it should in fact be encouraged.

4.4.3 General Polynomial Function over Quadratic Constraints

In this part, we report numerical tests on (Q) when $d = 4$:

$$\begin{aligned} (E_3) \quad \max \quad & p(\boldsymbol{x}) = F_4(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x}) + F_3(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{x}) + F_2(\boldsymbol{x}, \boldsymbol{x}) + F_1(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x}^{\mathrm{T}} Q_j \boldsymbol{x} \le 1, j = 1, 2, ..., m, \\ & \boldsymbol{x} \in \mathbb{R}^n, \end{aligned}$$

where tensors $\mathcal{F}_4 \in \mathbb{R}^{n^4}$, $\mathcal{F}_3 \in \mathbb{R}^{n^3}$, $\mathcal{F}_2 \in \mathbb{R}^{n^2}$, and $\mathcal{F}_1 \in \mathbb{R}^n$ are super-symmetric and $Q_j \succeq 0$ for $j = 1, 2, \ldots, m$. One natural way to handle an inhomogeneous polynomial function $p(\boldsymbol{x})$ is through homogenization, e.g., the technique used in [41]. To be specific, by introducing an auxiliary new variable x_h, which is set to be 1, we can homogenize function $p(\boldsymbol{x})$ as

$$p(\boldsymbol{x}) = F\left(\begin{pmatrix}\boldsymbol{x}\\ x_h\end{pmatrix}, \begin{pmatrix}\boldsymbol{x}\\ x_h\end{pmatrix}, \begin{pmatrix}\boldsymbol{x}\\ x_h\end{pmatrix}, \begin{pmatrix}\boldsymbol{x}\\ x_h\end{pmatrix}\right) := F(\bar{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{x}}) = f(\bar{\boldsymbol{x}}),$$

where $f(\bar{\boldsymbol{x}})$ is an $(n+1)$-dimensional homogeneous polynomial function of degree four, and its associated fourth-order super-symmetric tensor form $\mathcal{F} \in \mathbb{R}^{(n+1)^4}$. Therefore, by denoting $\bar{\boldsymbol{x}} := \binom{\boldsymbol{x}}{1}$, we may equivalently rewrite (E_3) as

$$\begin{array}{lll} (\bar{E}_3) & \max & f(\bar{\boldsymbol{x}}) = F\left(\binom{\boldsymbol{x}}{1}, \binom{\boldsymbol{x}}{1}, \binom{\boldsymbol{x}}{1}, \binom{\boldsymbol{x}}{1}\right) \\ & \text{s.t.} & \boldsymbol{x}^{\mathrm{T}} Q_j \boldsymbol{x} \leq 1, j = 1, 2, ..., m, \\ & & \boldsymbol{x} \in \mathbb{R}^n. \end{array}$$

We shall first call Algorithm MBI to solve the multilinear relaxation problem for $(\bar{E}_3)$ and get a KKT point, to be denoted by $(\boldsymbol{x}_*^1, \boldsymbol{x}_*^2, \boldsymbol{x}_*^3, \boldsymbol{x}_*^4)$. Then, we select the best one from those four vectors as a feasible point for the original model (E_3), namely, $\boldsymbol{x}_{\mathrm{MBI}} = \arg\max_{1 \leq i \leq 4}\{p(\boldsymbol{x}_*^i)\}$. Unlike the equivalence between (H) and (T) followed from Corollary 4.3.1 due to its special structure, $(\bar{E}_3)$ may not be equivalent to its tensor relaxation problem. Hence, in the last set of tests, starting from the point $\boldsymbol{x}_{\mathrm{MBI}}$, we further apply a projected gradient method [17] (denoted by PGM in Table 4.5) to improve the solution of (E_3). For an overview of gradient projection methods, one is referred to [12]. This method is also used as a supplement in [111, 95] for handling homogeneous polynomial optimization over ball constraint or spherical constraint. The projected gradient method is applied because this method converges to a KKT point of the problem concerned, and also the optimal projection from $\mathbb{R}^n$ onto the ellipsoidal constraints set $E = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x}^{\mathrm{T}} Q_j \boldsymbol{x} \leq 1, j = 1, 2, ..., m\}$ can be formulated as a second-order cone program (SOCP):

$$\begin{array}{ll} \min & \|\boldsymbol{x} - \boldsymbol{y}\| \\ \text{s.t.} & \boldsymbol{x} \in E, \end{array}$$

where $\boldsymbol{y} \in \mathbb{R}^n$ is given, which we call cvx to solve under the same computational platform. The starting points for the MBI are all randomly generated as before. Two test sets are constructed for (E_3). First, we fix $m = 15$ for varying n, and we test the

Table 4.4: CPU seconds of GloptiPoly 3 and MBI for (E_3) when $m = 15$

n	5	10	15	20	30	40
GLP	2.1830	14.6947	578.3944	∞	∞	∞
MBI	24.8763	42.1047	42.9723	43.6567	44.8106	44.9569

performance of GloptiPoly 3 and MBI in terms of the computational time, regardless of the quality of $\boldsymbol{x}_{\mathrm{MBI}}$ obtained by running MBI once. (Recall the relaxation order of GloptiPoly 3 is set by default.) Numerical results are listed in Table 4.4. Each entry is the average time of 10 randomly generated instances. From Table 4.4, we conclude that the computational time of MBI is insensitive to the dimension n, while GloptiPoly 3 is very sensitive to the dimension. In fact, the computational time of MBI is much less than that of GloptiPoly 3 when the dimension n gets large.

Second, we fix $m = 10$ and pick some lower dimensions n, whose problems can be efficiently solved by GloptiPoly 3. We then test the performance of MBI. Specifically, we solve (E_3) by three different approaches: (1) directly using GloptiPloy 3; (2) applying MBI with randomly generated starting points to get the point $\boldsymbol{x}_{\mathrm{MBI}}$; and (3) using projected gradient method with the starting point $\boldsymbol{x}_{\mathrm{MBI}}$. Numerical results are summarized in Table 4.5, which shows the excellent performance of the MBI method. GloptiPoly 3 is a powerful tool for solving (E_3) with low dimensions. However, the MBI method works very well for polynomial optimization over ellipsoidal constraints even in large dimensions.

4.5 Applications

Finally, we shall test our proposed algorithms by using data from real applications, including rank-one approximation of super-symmetric tensors, and magnetic resonance

Table 4.5: Numerical results for (E_3) when $m = 10$

dim.	# samples	GLP	MBI			MBI+PGM		
		mean(T)	Opt	mean(T)	mean(P)	Opt	mean(T)	mean(P)
$(4, 5)$	20	1.79	20%	22.81	85.57%	95%	29.68	98.21%
$(4, 10)$	20	13.01	0%	38.95	85.79%	95%	48.66	99.93%
$(4, 12)$	20	66.73	0%	41.36	89.61%	100%	50.13	100%

imaging (MRI).

4.5.1 Rank-One Approximation of Super-Symmetric Tensors

As discussed in Section 4.3, homogeneous polynomial optimization over spherical constraint is equivalent to the best rank-one approximation of super-symmetric tensors and hence is solvable by our methods. We consider an example in this part from Kofidis and Regalia (Example 1 of [59]). The authors of [59] used this example to show that their proposed method S-HOPM did not converge for the particular super-symmetric tensor $\mathcal{G} \in \mathbb{R}^{3\times3\times3\times3}$ with entries

$$\begin{aligned}
&\mathcal{G}_{1111} = 0.2883, \quad \mathcal{G}_{1112} = -0.0031, \quad \mathcal{G}_{1113} = 0.1973, \quad \mathcal{G}_{1122} = -0.2485,\\
&\mathcal{G}_{1123} = -0.2939, \quad \mathcal{G}_{1133} = 0.3847, \quad \mathcal{G}_{1222} = 0.2972, \quad \mathcal{G}_{1223} = 0.1862,\\
&\mathcal{G}_{1233} = 0.0919, \quad \mathcal{G}_{1333} = -0.3619, \quad \mathcal{G}_{2222} = 0.1241, \quad \mathcal{G}_{2223} = -0.3420,\\
&\mathcal{G}_{2233} = 0.2127, \quad \mathcal{G}_{2333} = 0.2727, \quad \mathcal{G}_{3333} = -0.3054.
\end{aligned}$$

We will test this example using MBI. In our setting, the best rank-one approximation of the tensor $\mathcal{G}$ is formulated as

$$\begin{aligned}
(E_4) \quad \max \quad & \sum_{1\le i,j,k,l\le 3} \mathcal{G}_{ijkl} x_i x_j x_k x_l\\
\text{s.t.} \quad & \|\boldsymbol{x}\| = 1,\ \boldsymbol{x} \in \mathbb{R}^3.
\end{aligned}$$

Since the order of $\mathcal{G}$ is even, we choose $\eta = 6$ and construct a modified and equivalent optimization problem of (E_4)

$$\begin{aligned} (E_5) \quad \max \quad & \sum_{1\le i,j,k,l\le 3} (\mathcal{G} + \eta\mathcal{H})_{ijkl} x_i x_j x_k x_l \\ \text{s.t.} \quad & \|\boldsymbol{x}\| = 1,\ \boldsymbol{x} \in \mathbb{R}^3, \end{aligned}$$

where $\mathcal{H}$ is a fourth-order super-symmetric tensor associated with the homogeneous polynomial $h(\boldsymbol{x}) = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x})^2$. For the reformulated problem, we apply Algorithm MBI to the multilinear tensor form relaxation of (E_5):

$$\begin{aligned} (E_6) \quad \max \quad & G(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{w}) = \sum_{1\le i,j,k,l\le 3} (\mathcal{G} + \eta\mathcal{H})_{ijkl} x_i y_j z_k w_l \\ \text{s.t.} \quad & \|\boldsymbol{x}\| = \|\boldsymbol{y}\| = \|\boldsymbol{z}\| = \|\boldsymbol{w}\| = 1, \\ & \boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{w} \in \mathbb{R}^3. \end{aligned}$$

By using MBI with randomly generated starting points, we get three local maximum solutions for (E_6). For each local maxima $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{w})$ we found, it shares the same directions among these four vectors when the MBI stops, i.e., $\boldsymbol{x} = \boldsymbol{y} = \boldsymbol{z} = \boldsymbol{w}$. Hence, it provides a local maxima for the original model (E_5), which is also a local maxima for (E_4). In Figure 4.1, the total number of iterations in each round of MBI is presented for each local maxima we have found. Indeed, MBI converges very quickly to a local maxima. The optimal value for (E_4) is 0.8893 (recall we should subtract 6 in the function $G(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{w})$), and the optimal solutions are $\boldsymbol{x}^* = \pm(0.6671, 0.2487, -0.7022)$. Hence, the best rank-one approximation for the super-symmetric tensor $\mathcal{G}$ is $0.8893\,\boldsymbol{x}^* \circ \boldsymbol{x}^* \circ \boldsymbol{x}^* \circ \boldsymbol{x}^*$.

4.5.2 Magnetic Resonance Imaging

Next, we shall conclude this section by considering one real data set for polynomial optimization in MRI. Ghosh *et al.* [29] formulated a fiber detection problem in diffusion MRI by maximizing a homogeneous polynomial function over spherical constraint. In

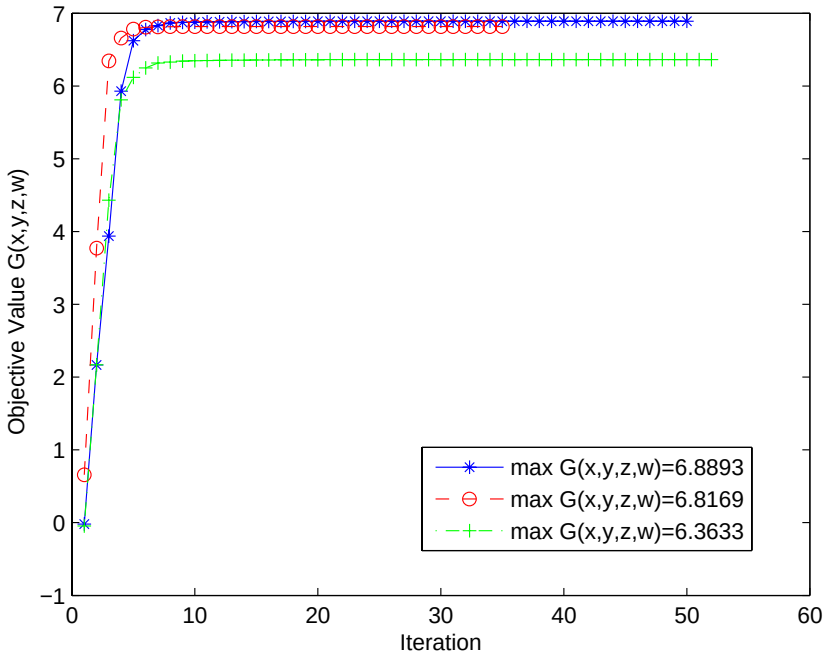

Figure 4.1: Convergence Results of MBI for (E_6)

this particular case, the following polynomial optimization model is considered

$$\begin{aligned} \max \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & \|\boldsymbol{x}\| = 1,\ \boldsymbol{x} \in \mathbb{R}^3, \end{aligned}$$

where $f(\boldsymbol{x})$ is a homogeneous polynomial of even degree d. The problem lives in three dimensions as in the real world, and all its local maxima have physical meanings for MRI.

We shall test our Algorithm MBI by using a set of data provided by Ghosh and Deriche. The corresponding objective function $f(\boldsymbol{x})$ is

$$\begin{aligned} & 0.74694\,x_0^4 - 0.435103\,x_0^3x_1 + 0.454945\,x_0^2x_1^2 + 0.0657818\,x_0x_1^3 \\ + & \quad x_1^4 + 0.37089\,x_0^3x_2 - 0.29883\,x_0^2x_1x_2 - 0.795157\,x_0x_1^2x_2 \\ + & \,0.139751\,x_1^3x_2 + 1.24733\,x_0^2x_2^2 + 0.714359\,x_0x_1x_2^2 + 0.316264\,x_1^2x_2^2 \\ - & \,0.397391\,x_0x_2^3 - 0.405544\,x_1x_2^3 + 0.794869\,x_2^4, \end{aligned}$$

Table 4.6: Numerical results for MRI

Method	KKT solution	Objective value
MBI	$\pm(0.0116, 0.9992, 0.0382)$	1.0031
	$\pm(0.3166, 0.2130, -0.9243)$	0.9213
	$\pm(0.9542, -0.1434, 0.2624)$	0.8428
GLP	$\pm(0.0116, 0.9992, 0.0382)$	1.0031

where $\boldsymbol{x} = (x_0, x_1, x_2)^{\mathrm{T}}$. By choosing $\eta = 2$, we adopt the same procedures for rank-one approximation of super-symmetric tensors discussed in Section 4.5.1 to solve the MRI problem. GloptiPoly 3 is also called for comparison. The numerical results are reported in Table 4.6. The MBI method is able to find all three local maxima, while GloptiPoly 3 finds only the global maximum.

Chapter 5

Logarithmically Quasiconvex Optimization

5.1 Introduction

As seen from the previous chapter, polynomial optimization with spherical constraint is theoretically interesting and practically solvable. In this chapter, we continue to investigate polynomial optimization models, with general constraint. In Algorithm KKT, we designed a polynomial-time algorithm to find a KKT solution for (H) via its relaxation model (T). The approach is quite straightforward but looks tedious. One may then wonder if there is any simpler way to construct a KKT solution for homogeneous function optimization. The answer is yes. Here in this chapter, we propose an alternative method to search KKT solutions for homogeneous polynomial optimization via multilinear function optimization. To begin with, let us start by considering some special objective functions in optimization.

Convexity or concavity plays an important role in optimization. For example, minimizing a convex function over a convex set can be solved in general in polynomial-time.

It is not hard to check the convexity of a quadratic function, which only needs to test the positive semidefiniteness of its Hessian matrix. How about checking the convexity of quartic (fourth degree) polynomial function? It turns out to be a very challenging question. Recently Ahmadi *et al.* [1] proved that checking the convexity of a quartic polynomial function is actually strongly NP-hard in general, which settles a long-standing open question. Meanwhile, their results help to highlight a crucial difference between quadratic and quartic polynomials. An interesting work is conducted by Jiang *et al.* [53], where they studied six fundamentally important convex cones of homogeneous quartic functions, including the cone of nonnegative quartic forms, the sum of squared quartic forms, the convex quartic forms, and the sum of fourth powered polynomials. The complexity status of these cones are discussed as well. This work also motivates us to study quartic or even higher degree polynomial optimization problems.

As studied in Chapter 4, Corollary 4.3.1 established an important linkage between homogeneous polynomial optimization over spherical constraint and its multilinear form relaxation problem. In order to drop the absolute value sign for the even degree case in the corollary, the nonnegativity of the objective function is required. It is natural to ask whether we can design a reasonable nonnegative function based on some particular domain. In fact, there is an intrinsic connection between optimizing a polynomial function and the description of all polynomial functions which are nonnegative over a given domain. This connection was explored by Sturm and Zhang [100] for the case of quadratic polynomials, and Luo *et al.* [77] for the bi-quadratic functions. We also refer to [44, 47, 13, 1, 53] and a recent book [14] for investigating the relationship between nonnegative polynomial functions and the sum of squares (SOS) of polynomials.

Motivated by all the discussions above, here we define a new function, to be called *logarithmically quasiconvex* function, in Section 5.2. Inspired by the nice property of the new function and Theorem 4.2.1, we establish an equivalence between homogeneous

polynomial optimization and its tensor relaxation problem, based on a specific type of non-negativity of the tensor form. This enables the application of the MBI method to solve some polynomial optimization models. This also suggests a simple way to find KKT solutions for homogeneous polynomial optimization.

5.2 Logarithmically Quasiconvex Optimization

Let us begin with the definition of logarithmically quasiconvex function.

Definition 5.2.1. *Suppose F is a multilinear function induced by a $2m$-th order super-symmetric tensor $\mathcal{F} \in \mathbb{R}^{n^{2m}}$. Function F is called logarithmically quasiconvex (log-quasiconvex), if*

$$F(\underbrace{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_1}_{\lambda_1}, \underbrace{\boldsymbol{x}_2, \cdots, \boldsymbol{x}_2}_{\lambda_2}, \cdots, \underbrace{\boldsymbol{x}_s, \cdots, \boldsymbol{x}_s}_{\lambda_s}) \leq \max_{1 \leq i \leq s} \{F(\underbrace{\boldsymbol{x}_i, \boldsymbol{x}_i, \cdots, \boldsymbol{x}_i}_{2m})\}$$
$$\forall\, \boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_s \in \mathbb{R}^n$$

for any integers $\lambda_i \geq 0$ $(i = 1, 2, \ldots, s)$ with $\sum_{i=1}^{s} \lambda_i = 2m$. In particular, if the above inequality holds in a special case when $s = m$ and $\lambda_1 = \lambda_2 = \cdots = \lambda_m = 2$, F is called co-quadratic quasiconvex.

To make the notation simpler, whenever appropriate in this chapter we shall use superscripts to simplify $F(\underbrace{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_1}_{\lambda_1}, \underbrace{\boldsymbol{x}_2, \cdots, \boldsymbol{x}_2}_{\lambda_2}, \cdots, \underbrace{\boldsymbol{x}_s, \cdots, \boldsymbol{x}_s}_{\lambda_s})$, i.e., $F(\boldsymbol{x}_1^{\lambda_1} \boldsymbol{x}_2^{\lambda_2} \cdots \boldsymbol{x}_s^{\lambda_s})$. For example, $F(\boldsymbol{x}^2 \boldsymbol{y}^2)$ denotes $F(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{y})$. We shall establish an equivalence result between homogeneous polynomial optimization and multilinear form optimization over any constraint set, based on a special tensor form, to be called co-quadratic positive semidefinite tensor.

5.2.1 A Simple Motivating Example

Suppose that $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, and matrix $Q \in \mathbb{R}^{n\times n}$ is positive semidefinite. We claim that function $F(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T Q \boldsymbol{y}$ is log-quasiconvex. Indeed, for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, it is obvious that $(\boldsymbol{x} - \boldsymbol{y})^T Q(\boldsymbol{x} - \boldsymbol{y}) \geq 0$ as Q is positive semidefinite, implying that

$$\boldsymbol{x}^T Q \boldsymbol{x} + \boldsymbol{y}^T Q \boldsymbol{y} \geq 2\, \boldsymbol{x}^T Q \boldsymbol{y}.$$

The above inequality further leads to

$$\max\{\boldsymbol{x}^T Q \boldsymbol{x}, \boldsymbol{y}^T Q \boldsymbol{y}\} \geq \boldsymbol{x}^T Q \boldsymbol{y}.$$

Therefore F is log-quasiconvex. Based on the definition of log-quasiconvex, the following lemma is immediate.

Lemma 5.2.2. *For any set $S \subseteq \mathbb{R}^n$ and positive semidefinite matrix $Q \in \mathbb{R}^{n\times n}$, it follows that*

$$(L_2) \quad \max_{\boldsymbol{x} \in S} \ \boldsymbol{x}^T Q \boldsymbol{x} \ = \ \max_{\boldsymbol{x}, \boldsymbol{y} \in S} \ \boldsymbol{x}^T Q \boldsymbol{y}. \quad (R_2)$$

Recall in Section 4.3.3, we designed a polynomial-time algorithm to find a KKT solution for homogeneous polynomial optimization (H) from the solutions of multilinear polynomial optimization (T). However, it is much easier here to find an optimal solution for (L_2) if we know an optimal solution of multilinear optimization model (R_2), since the objective function $F(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T Q \boldsymbol{y}$ in Lemma 5.2.2 is log-quasiconvex. In the following, we present an extension of Lemma 5.2.2.

5.2.2 Co-Quadratic Positive Semidefinite Tensor Form

Notice that the matrix Q in Lemma 5.2.2 is positive semidefinite implying the function $\boldsymbol{x}^T Q \boldsymbol{y}$ is log-quasiconvex. In order to generalize the result of Lemma 5.2.2, let us introduce the following definition.

Definition 5.2.3. *A super-symmetric tensor form $\mathcal{F} \in \mathbb{R}^{n^{2m}}$ is called co-quadratic positive semidefinite, if its associated multilinear function*

$$F(\boldsymbol{x}_1^2\boldsymbol{x}_2^2\cdots\boldsymbol{x}_m^2) = F(\boldsymbol{x}_1,\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_m,\boldsymbol{x}_m) \geq 0 \quad \forall\, \boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_m \in \mathbb{R}^n.$$

Notice that by denoting $f(\boldsymbol{x}) = F(\boldsymbol{x}^m)$ to be the homogeneous polynomial function induced by the tensor form $\mathcal{F}$, the general positive semidefinite means that $f(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$. In particular, when $m = 1$, co-quadratic positive semidefinite is equivalent to usual positive semidefinite quadratic form, and when $m = 2$, co-quadratic positive semidefinite is equivalent to $f(\boldsymbol{x})$ being a convex function; see the following lemma.

Lemma 5.2.4. *Homogeneous quartic function $F(\boldsymbol{x}^4)$ is convex if and only if its associated super-symmetric tensor form $\mathcal{F}$ is co-quadratic positive semidefinite.*

Proof. It is straightforward to compute the Hessian matrix of $F(\boldsymbol{x}^4)$, which is $12 \cdot F(\boldsymbol{x},\boldsymbol{x},\cdot,\cdot)$. Therefore, the quartic function is convex if and only if

$$F(\boldsymbol{x},\boldsymbol{x},\cdot,\cdot) \text{ is positive semidefinite} \quad \forall\, \boldsymbol{x} \in \mathbb{R}^n,$$

which is equivalent to

$$F(\boldsymbol{x},\boldsymbol{x},\boldsymbol{y},\boldsymbol{y}) \geq 0 \quad \forall\, \boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^n.$$

The lemma is proved. □

Therefore, when $m = 2$, the usual positive semidefinite quartic form $F(\boldsymbol{x}^4)$ is not necessarily convex, while convexity implies positive semidefinite. Hence, in general, co-quadratic positive semidefiniteness is stronger than positive semidefiniteness. Interested readers are referred to Jiang *et al.* [53] for a detailed discussion on different classes of positive semidefinite quartic forms. Two examples for co-quadratic positive semidefinite tensor forms of general order are presented below.

Example 5.2.5. *The super-symmetric tensor* $\mathcal{F} = \sum_{i=1}^{k} \underbrace{\boldsymbol{a}_i \circ \boldsymbol{a}_i \circ \cdots \circ \boldsymbol{a}_i}_{2m}$ *with* $\boldsymbol{a}_i \in \mathbb{R}^n$, *is co-quadratic positive semidefinite, since its associated multilinear function*

$$F(\boldsymbol{x}_1^2 \boldsymbol{x}_2^2 \cdots \boldsymbol{x}_m^2) = \sum_{i=1}^{k} (\boldsymbol{a}_i^T \boldsymbol{x}_1)^2 (\boldsymbol{a}_i^T \boldsymbol{x}_2)^2 \cdots (\boldsymbol{a}_i^T \boldsymbol{x}_m)^2 \geq 0 \quad \forall\, \boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m \in \mathbb{R}^n.$$

Example 5.2.6. *The super-symmetric tensor form* $\mathcal{F} \in \mathbb{R}^{n^{2m}}$ *associated with homogeneous polynomial* $f(\boldsymbol{y}) = \|\boldsymbol{y}\|^{2m} = (\boldsymbol{y}^T \boldsymbol{y})^m$ *is co-quadratic positive semidefinite. Explicitly, the multilinear function induced by the tensor* $\mathcal{F}$ *is as follows:*

$$F(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_{2m}) = \frac{1}{|\sigma|} \sum_{(i_1 i_2 \cdots i_{2m}) \in \sigma} (\boldsymbol{y}_{i_1}^T \boldsymbol{y}_{i_2})(\boldsymbol{y}_{i_3}^T \boldsymbol{y}_{i_4}) \cdots (\boldsymbol{y}_{i_{2m-1}}^T \boldsymbol{y}_{i_{2m}}), \tag{5.1}$$

where σ *is the set of all permutations of* $\{1, 2, \ldots, 2m\}$, *and* $|\sigma|$ *is the total number of the permutations.*

For $m = 1$, *tensor* $\mathcal{F}$ *is actually an* $n \times n$ *identity matrix, which is positive semidefinite, implying* $\mathcal{F}$ *is co-quadratic positive semidefinite. For* $m = 2$, *we have*

$$F(\boldsymbol{x}_1, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_2) = \frac{1}{24} \left(8(\boldsymbol{x}_1^T \boldsymbol{x}_1)(\boldsymbol{x}_2^T \boldsymbol{x}_2) + 16(\boldsymbol{x}_1^T \boldsymbol{x}_2)^2\right) \geq 0 \quad \forall\, \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n.$$

For general $m \geq 3$, *it is not an easy task to directly check the nonnegativity of* $F(\boldsymbol{x}_1^2 \boldsymbol{x}_2^2 \cdots \boldsymbol{x}_m^2)$ *using* (5.1). *However, by using the so-called Hilbert's identity (see, e.g., [11]), which states that there exist vectors* $\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_k \in \mathbb{R}^n$ *such that*

$$(\boldsymbol{y}^T \boldsymbol{y})^m = \sum_{i=1}^{k} \langle \boldsymbol{c}_i, \boldsymbol{y} \rangle^{2m} \quad \forall\, \boldsymbol{y} \in \mathbb{R}^n,$$

the super-symmetric tensor $\mathcal{F}$ *can then be expressed by*

$$\mathcal{F} = \sum_{i=1}^{k} \underbrace{\boldsymbol{c}_i \circ \boldsymbol{c}_i \circ \cdots \circ \boldsymbol{c}_i}_{2m},$$

which is co-quadratic positive semidefinite as Example 5.2.5 claimed.

Remark that in the case $m = 2$, the number k is exponential in n in Hilbert's construction. In fact, it can be reduced by a polynomial function of n, see the recent work of He *et al.* [38].

5.2.3 Equivalence at Maxima

Motivated by the equivalence in Lemma 5.2.2, in this subsection we shall establish the relationship between maximization of a homogeneous co-quadratic positive semidefinite form and its multilinear function relaxation problem, which is actually *equivalent*. First, we need the following result.

Theorem 5.2.7. *If super-symmetric tensor $\mathcal{F} \in \mathbb{R}^{n^{2m}}$ is co-quadratic positive semidefinite, then its associated multilinear function F is co-quadratic quasiconvex, i.e.,*

$$F(\boldsymbol{x}_1^2\boldsymbol{x}_2^2\cdots\boldsymbol{x}_m^2) \leq \max_{1\leq i\leq m}\{F(\boldsymbol{x}_i^{2m})\} \quad \forall\, \boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m \in \mathbb{R}^n.$$

Proof. It is sufficient to prove

$$\sum_{i=1}^{m} F(\boldsymbol{x}_i^{2m}) \geq m\, F(\boldsymbol{x}_1^2\boldsymbol{x}_2^2\cdots\boldsymbol{x}_m^2) \quad \forall\, \boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m \in \mathbb{R}^n.$$

In fact, the above inequality can be proved by induction on m. It is trivial when $m = 1$. Suppose it holds for the case m, then for the case $m+1$, we need to prove the following

$$\sum_{i=1}^{m+1} F(\boldsymbol{x}_i^{2m+2}) \geq (m+1)\, F(\boldsymbol{x}_1^2\boldsymbol{x}_2^2\cdots\boldsymbol{x}_m^2\boldsymbol{x}_{m+1}^2). \tag{5.2}$$

First, we claim that for any $\boldsymbol{z}, \boldsymbol{w} \in \mathbb{R}^n$, it holds that

$$F(\boldsymbol{z}^{2m+2}) + F(\boldsymbol{w}^{2m+2}) \geq F(\boldsymbol{z}^2\boldsymbol{w}^{2m}) + F(\boldsymbol{z}^{2m}\boldsymbol{w}^2). \tag{5.3}$$

Indeed this is because

$$\begin{aligned}
& F(\boldsymbol{z}^{2m+2}) + F(\boldsymbol{w}^{2m+2}) - F(\boldsymbol{z}^2\boldsymbol{w}^{2m}) - F(\boldsymbol{z}^{2m}\boldsymbol{w}^2) \\
= \;& \big(F(\boldsymbol{z}^2\boldsymbol{z}^{2m}) - F(\boldsymbol{z}^2\boldsymbol{w}^{2m})\big) - \big(F(\boldsymbol{w}^2\boldsymbol{z}^{2m}) - F(\boldsymbol{w}^2\boldsymbol{w}^{2m})\big) \\
= \;& F\left((\boldsymbol{z}+\boldsymbol{w})^1(\boldsymbol{z}-\boldsymbol{w})^1\boldsymbol{z}^{2m}\right) - F\left((\boldsymbol{z}+\boldsymbol{w})^1(\boldsymbol{z}-\boldsymbol{w})^1\boldsymbol{w}^{2m}\right) \\
= \;& \sum_{i=0}^{m-1}\binom{m-1}{i} F\left((\boldsymbol{z}+\boldsymbol{w})^2(\boldsymbol{z}-\boldsymbol{w})^2(\boldsymbol{z}^2)^i(\boldsymbol{w}^2)^{m-1-i}\right) \geq 0,
\end{aligned}$$

where the last inequality holds since $\mathcal{F}$ is co-quadratic positive semidefinite. Therefore by (5.3), we have

$$F(\boldsymbol{x}_j^{2m+2}) + F(\boldsymbol{x}_i^{2m+2}) \geq F(\boldsymbol{x}_j^2\boldsymbol{x}_i^{2m}) + F(\boldsymbol{x}_j^{2m}\boldsymbol{x}_i^2) \quad \forall\, 1 \leq i, j \leq m+1.$$

Summing over all $i < j$ further leads to

$$\sum_{1\leq i<j\leq m+1} \Big(F(\boldsymbol{x}_j^{2m+2}) + F(\boldsymbol{x}_i^{2m+2})\Big) \geq \sum_{1\leq i<j\leq m+1} \Big(F(\boldsymbol{x}_j^2\boldsymbol{x}_i^{2m}) + F(\boldsymbol{x}_j^{2m}\boldsymbol{x}_i^2)\Big),$$

which implies that

$$m\sum_{i=1}^{m+1} F(\boldsymbol{x}_i^{2m+2}) \geq \sum_{i=1}^{m+1}\sum_{j\neq i} F(\boldsymbol{x}_i^2\boldsymbol{x}_j^{2m}) \geq \sum_{i=1}^{m+1} mF\left(\boldsymbol{x}_i^2\prod_{j\neq i}\boldsymbol{x}_j^2\right) = m(m+1)F\left(\prod_{j=1}^{m+1}\boldsymbol{x}_j^2\right),$$

where in the second inequality, the induction assumption on m is applied. This proves (5.2). Hence, F is co-quadratic quasiconvex. □

Now, we are ready to show the equivalence result below, similar to Lemma 5.2.2.

Theorem 5.2.8. *If $\mathcal{F} \in \mathbb{R}^{n^{2m}}$ is co-quadratic positive semidefinite, then for any $S \subseteq \mathbb{R}^n$,*

$$\begin{aligned} & \max_{\boldsymbol{x}\in S} \; F(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x}}_{2m}) \quad (L_{2m}) \\ = & \max_{\boldsymbol{y}_1,\boldsymbol{y}_2,\cdots,\boldsymbol{y}_m\in S} \; F(\boldsymbol{y}_1, \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m, \boldsymbol{y}_m) \quad (M_{2m}) \\ = & \max_{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_{2m}\in S} \; F(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{2m}). \quad (R_{2m}) \end{aligned}$$

Proof. We need to prove that $v(L_{2m}) = v(M_{2m}) = v(R_{2m})$. Clearly we have $v(L_{2m}) \leq v(M_{2m}) \leq v(R_{2m})$. Besides, from Theorem 5.2.7 we know that $v(L_{2m}) \geq v(M_{2m})$ since $\mathcal{F} \in \mathbb{R}^{n^{2m}}$ is co-quadratic positive semidefinite. Therefore, $v(L_{2m}) = v(M_{2m})$.

Denote $(\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \cdots, \hat{\boldsymbol{x}}_{2m})$ to be an optimal solution of (R_{2m}), and let $\xi_1, \xi_2, \cdots, \xi_m$ be Benoulli random variables, each taking values 1 and -1 with equal probability,

satisfying $\Pi_{i=1}^{m}\xi_i = -1$. Observe that

$$\begin{aligned} 0 &\leq E\left[F\Big((\hat{\boldsymbol{x}}_1+\xi_1\hat{\boldsymbol{x}}_2)^2(\hat{\boldsymbol{x}}_3+\xi_2\hat{\boldsymbol{x}}_4)^2\cdots(\hat{\boldsymbol{x}}_{2m-1}+\xi_m\hat{\boldsymbol{x}}_{2m})^2\Big)\right] \\ &= \sum_{i_1=1}^{2}\sum_{i_2=3}^{4}\cdots\sum_{i_m=2m-1}^{2m} F(\hat{\boldsymbol{x}}_{i_1}^2\hat{\boldsymbol{x}}_{i_2}^2\cdots\hat{\boldsymbol{x}}_{i_m}^2) - 2^m F(\hat{\boldsymbol{x}}_1,\hat{\boldsymbol{x}}_2,\cdots,\hat{\boldsymbol{x}}_{2m}). \end{aligned}$$

This is because the expectation of the coefficient of each remaining term is 0, e.g., $E[\xi_1^2\xi_2\xi_3\cdots\xi_m] = -E[\xi_1] = 0$. As $\mathcal{F}$ is co-quadratic positive semidefinite, we have $F(\hat{\boldsymbol{x}}_{i_1}^2\hat{\boldsymbol{x}}_{i_2}^2\cdots\hat{\boldsymbol{x}}_{i_m}^2) \geq 0$ for any $i_1 \in \{1,2\}, i_2 \in \{3,4\},\cdots,i_m \in \{2m-1,2m\}$. Thus, there exist indices $i_1', i_2', \cdots, i_m'$, such that

$$F(\hat{\boldsymbol{x}}_{i_1'}^2\hat{\boldsymbol{x}}_{i_2'}^2\cdots\hat{\boldsymbol{x}}_{i_m'}^2) \geq F(\hat{\boldsymbol{x}}_1,\hat{\boldsymbol{x}}_2,\cdots,\hat{\boldsymbol{x}}_{2m}),$$

which implies that $v(M_{2m}) \geq v(R_{2m})$ and $(\hat{\boldsymbol{x}}_{i_1'},\hat{\boldsymbol{x}}_{i_2'},\cdots,\hat{\boldsymbol{x}}_{i_m'})$ is an optimal solution of (M_{2m}). This shows that $v(M_{2m}) = v(R_{2m})$, completing the whole proof. □

As stated in Section 4.3 and Section 5.2.1, Theorem 5.2.8 suggests an alternative way to deal with homogeneous polynomial optimization problem, say problem (L_{2m}). The procedure can be divided into two steps. The first step is to relax (L_{2m}) to multi-quadratic form optimization (M_{2m}) or multilinear form optimization (R_{2m}). One choice for solving (M_{2m}) or (R_{2m}) is to implement the MBI method presented in Chapter 2. The second step is to construct a KKT solution (or, an optimal solution) for the original problem. In this circumstance, suppose that $(\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_{2m})$ is a KKT solution (or, an optimal solution) for (R_{2m}), then we can directly find the KKT solution (or, the optimal solution) $\boldsymbol{x}_{i^*}$ for (L_{2m}), where

$$i^* = \arg\max_{i\in\{1,2,\cdots,2m\}} F(\underbrace{\boldsymbol{x}_i,\boldsymbol{x}_i,\cdots,\boldsymbol{x}_i}_{2m}), \tag{5.4}$$

as the proof of Theorem 5.2.8 suggested. The best solution for (L_{2m}) is already among the solutions $\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_{2m}$. Therefore, it is much simpler than finding KKT solutions for (H) as Algorithm KKT constructed in Section 4.3.3 of Chapter 4. This is

because the objective function of (L_{2m}) enjoys the nice property of being co-quadratic quasiconvex.

An immediate consequence of Theorem 5.2.8 is the following:

Corollary 5.2.9. *Suppose that $\mathcal{F} \in \mathbb{R}^{n^{2m}}$ is co-quadratic positive semidefinite. If integers $\lambda_i \geq 0$ $(i = 1, 2, \ldots, s)$ with $\sum_{i=1}^{s} \lambda_i = 2m$, then for any $S \subseteq \mathbb{R}^n$,*

$$\max_{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_s \in S} F(\underbrace{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_1}_{\lambda_1}, \underbrace{\boldsymbol{x}_2, \cdots, \boldsymbol{x}_2}_{\lambda_2}, \cdots, \underbrace{\boldsymbol{x}_s, \cdots, \boldsymbol{x}_s}_{\lambda_s}) = \max_{\boldsymbol{x} \in S} F(\underbrace{\boldsymbol{x}, \boldsymbol{x}, \cdots, \boldsymbol{x}}_{2m}).$$

Chapter 6

The Tucker Decomposition and Generalization

6.1 Introduction

As discussed in Chapter 4, finding the best rank-one decomposition of tensors, which is a special case of CP decomposition, is NP-hard in general. However, we may apply the MBI method to find a KKT solution for the decomposition efficiently. In this chapter, we shall investigate another form of higher-order PCA, namely, the Tucker decomposition, and demonstrate how the MBI method applies to solve the Tucker decomposition.

The Tucker decomposition can be viewed as a generalization of the CP decomposition which is the Tucker model with equal number of components in each mode. The Tucker decomposition is first introduced in 1963 by Tucker [107], which was later redefined in Levin [72] and Tucker [108, 109]. The goal of the Tucker decomposition is to decompose a tensor $\mathcal{F}$ into a core tensor multiplied by a matrix along each mode. It is related to finding the best rank-$(r_1, r_2, \cdots, r_d)$ approximation of d-th order tensors; see,

e.g., [69]. Therefore, we may treat Tucker decomposition as rank-$(r_1, r_2, \cdots, r_d)$ Tucker decomposition, or best multilinear rank-$(r_1, r_2, \cdots, r_d)$ approximation of tensors. Regarding the decomposition methods, Tucker [109] proposed three methods for solving Tucker decomposition for three-way tensors in 1966, among which the first method is sometimes referred to the "Tucker1" method, and is now better known as the higher-order singular value decomposition (HOSVD) from the work of De Lathauwer *et al.* [68]. In [68], they not only showed that the HOSVD for the tensor is a convincing generalization of SVD for the matrix case, but also proposed a truncated HOSVD that gives a suboptimal rank-$(r_1, r_2, \cdots, r_d)$ approximation of tensors, as well as suggesting good starting point for other algorithms. Analogous to the ALS method developed for computing the best rank-one approximation of tensors in Chapter 4, researchers also derived similar methods for solving the Tucker decomposition comparable to the ALS method. Kroonenberg and De Leeuw [62] developed TUCKALS3 for computing the Tucker decomposition for the three-way arrays, and a variant version TUCKALS2 for computing the Tucker2 decomposition for the three-way arrays by using two modes of the data. Later Kapteyn *et al.* [54] extended the TUCKALS3 to decompose the general d-way arrays for $d > 3$. Moreover, De Lathauwer *et al.* [69] derived a more efficient technique called higher-order orthogonal iteration (HOOI) method for computing the rank-$(r_1, r_2, \cdots, r_d)$ Tucker decomposition. The methods on how to speed up the HOOI algorithm are considered by Andersson and Bro [4]. However, as we mentioned earlier, the ALS method is not guaranteed to converge to a global minimum or a stationary point, only to a solution where the objective function ceases to decrease. On the other hand, there are convergence methods for the Tucker decomposition. Recently, Eldén and Savas [27], and Ishtev *et al.* [52] simultaneously proposed Newton-based method for computing the best multilinear rank-(r_1, r_2, r_3) approximation of tensors, the former one is called Newton-Grassmann method, and the latter

one is called differential-geometric Newton method. Both methods have quadratic local convergence property; however, the computational efforts to the Hessian are very demanding. For more information on Tucker decomposition, we refer to the excellent survey by Kolda and Bader [61].

In the Tucker decomposition, the rank-$(r_1, r_2, \cdots, r_d)$ is predetermined parameters. Therefore, the question of how to choose the rank of a Tucker model (cf. [105, 56, 43]) is quite challenging, as the problem is already NP-hard for a given $(r_1, r_2, \cdots, r_d)$. Timmerman and Kiers [105] proposed DIFFIT procedure based on optimal fit to choose the numbers of components in Tucker decomposition of three-way data. The procedure, however, is rather time-consuming. Kiers and Der Kinderen [56] revised the procedure, and computed DIFFIT on approximate fit to save computational time. Similar issue on selecting an appropriate rank has been considered in the CP decomposition; see, e.g., a consistency diagnostic named CORCONDIA designed in [16]. Moreover, the problem on how to choose a "good" number of clusters for co-clustering of gene expression data also comes up in the area of bioinformatics. As shown in Chapter 3, the co-clusters is given at the first beginning. Later Zhang *et al.* [114] derived a computational framework of co-identification that enables choosing the number of clusters.

In this chapter, we shall apply the MBI method for solving rank-$(r_1, r_2, \cdots, r_d)$ Tucker decomposition in Section 6.2. In Section 6.3, a new model for the Tucker decomposition with unspecified number of components is proposed and solved by using the MBI method. A heuristic approach is also proposed for computing the new model in this section. Some numerical results on testing the new model will be presented in Section 6.4.

6.2 Convergence of Traditional Tucker Decomposition

Traditionally, the Tucker decomposition tries to find the best approximation for a large-sized tensor by a small-sized tensor with pre-specified dimension of each mode, or equivalently, rank of each mode. Formally, the problem can be formulated as follows.

> Given a real tensor $\mathcal{F} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, find a core tensor $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_d}$ with pre-specified integers r_i with $1 \leq r_i \leq n_i$ for $i = 1, 2, \cdots, d$, that optimizes

$$
\begin{aligned}
(TD_{min}) \quad \min \quad & \|\mathcal{F} - \mathcal{C} \times_1 A^{(1)} \times_2 A^{(2)} \cdots \times_d A^{(d)}\| \\
\text{s.t.} \quad & \mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_d}, \\
& A^{(i)} \in \mathbb{R}^{n_i \times r_i} \text{ and columnwise orthogonal}, i = 1, 2, \ldots, d.
\end{aligned}
$$

Here, matrices $A^{(i)}$'s are the factor matrices. Without loss of generality, these matrices are assumed to be columnwise orthogonal. This problem can be considered as a generalization of best rank-one approximation problem discussed in Chapter 4.

For any fixed matrices $A^{(i)}$'s, if we optimize the objective function of (TD_{min}) over $\mathcal{C}$, then we have

$$\mathcal{C} = \mathcal{F} \times_1 (A^{(1)})^T \times_2 (A^{(2)})^T \cdots \times_d (A^{(d)})^T.$$

We may then remove the constraint for $\mathcal{C}$ in (TD_{min}), as the following derivation claims:

$$
\begin{aligned}
& \|\mathcal{F} - \mathcal{C} \times_1 A^{(1)} \times_2 A^{(2)} \cdots \times_d A^{(d)}\| \\
= \; & \sqrt{\|\mathcal{F}\|^2 - 2\langle \mathcal{F}, \mathcal{C} \times_1 A^{(1)} \times_2 A^{(2)} \cdots \times_d A^{(d)} \rangle + \|\mathcal{C} \times_1 A^{(1)} \times_2 A^{(2)} \cdots \times_d A^{(d)}\|^2} \\
= \; & \sqrt{\|\mathcal{F}\|^2 - 2\langle \mathcal{F} \times_1 (A^{(1)})^T \times_2 (A^{(2)})^T \cdots \times_d (A^{(d)})^T, \mathcal{C} \rangle + \|\mathcal{C}\|^2} \\
= \; & \sqrt{\|\mathcal{F}\|^2 - 2\|\mathcal{C}\|^2 + \|\mathcal{C}\|^2} \\
= \; & \sqrt{\|\mathcal{F}\|^2 - \|\mathcal{C}\|^2} \\
= \; & \sqrt{\|\mathcal{F}\|^2 - \|\mathcal{F} \times_1 (A^{(1)})^T \times_2 (A^{(2)})^T \cdots \times_d (A^{(d)})^T\|^2},
\end{aligned}
$$

where the second equality holds due to the orthogonality of matrices $A^{(i)}$ for $i = 1, 2, \ldots, d$. Remark that similar deductions are also discussed in [61, 69, 4]. Therefore, (TD_{min}) is indeed equivalent to the following maximization problem

$$\begin{aligned} (TD_{max}) \quad \max \quad & \|\mathcal{F} \times_1 (A^{(1)})^T \times_2 (A^{(2)})^T \cdots \times_d (A^{(d)})^T\| \\ \text{s.t.} \quad & A^{(i)} \in \mathbb{R}^{n_i \times r_i} \text{ and columnwise orthogonal}, i = 1, 2, \ldots, d. \end{aligned}$$

Notice that (TD_{max}) falls into the structure of the generic model (G), we may apply the MBI method to solve it once again. For this particular problem, the subproblems (G_i) in Algorithm MBI can be easily solved by the singular value decomposition (SVD). To be specific, consider subproblem

$$\begin{aligned} (G_i) \quad \max_{A^{(i)}} \quad & \|\mathcal{F} \times_1 (A^{(1)})^T \times_2 (A^{(2)})^T \cdots \times_d (A^{(d)})^T\| \\ \text{s.t.} \quad & A^{(i)} \in \mathbb{R}^{n_i \times r_i} \text{ and columnwise orthogonal.} \end{aligned}$$

We can rewrite its objective function in the matrix form as follows

$$\|(A^{(i)})^T F_{(i)} \left(A^{(d)} \otimes \cdots \otimes A^{(i+1)} \otimes A^{(i-1)} \otimes \cdots \otimes A^{(1)}\right)\|.$$

If we denote $M^{(i)} := F_{(i)} \left(A^{(d)} \otimes \cdots \otimes A^{(i+1)} \otimes A^{(i-1)} \otimes \cdots \otimes A^{(1)}\right)$, then the optimal solution for (G_i) is the r_i leading left singular vectors of the matrix $M^{(i)}$. We now present the following procedure for solving (TD_{max}) using the MBI method.

Algorithm TD1. The MBI method for the Tucker decomposition.

- **Input:** Tensor $\mathcal{F} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ and parameters $r_1, r_2, \cdots, r_d$.
- **Output:** Core tensor $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_d}$.

0 Choose a feasible solution $(A_0^{(1)}, A_0^{(2)}, \cdots, A_0^{(d)})$ and compute initial objective value $v_0 := \|\mathcal{F} \times_1 (A_0^{(1)})^T \times_2 (A_0^{(2)})^T \cdots \times_d (A_0^{(d)})^T\|$. Set $k := 0$.

1 For each $i = 1, 2, \ldots, d$, set $B_{k+1}^{(i)}$ to be the r_i leading left singular vectors of matrix

$$F_{(i)}\left(A_k^{(d)} \otimes \cdots \otimes A_k^{(i+1)} \otimes A_k^{(i-1)} \otimes \cdots \otimes A_k^{(1)}\right),$$

and let

$$w_{k+1}^i := \|\mathcal{F} \times_1 (A_k^{(1)})^T \cdots \times_{i-1} (A_k^{(i-1)})^T \times_i (B_{k+1}^{(i)})^T \times_{i+1} (A_k^{(i+1)})^T \cdots \times_d (A_k^{(d)})^T\|.$$

2 Let $w_{k+1} := \max_{1 \le i \le d} w_{k+1}^i$ and $i^* = \arg\max_{1 \le i \le d} w_{k+1}^i$. Let

$$\begin{aligned} A_{k+1}^{(i)} &:= A_k^{(i)}, \ \forall\ i \in \{1, 2, \cdots, d\} \backslash \{i^*\}, \\ A_{k+1}^{(i^*)} &:= B_{k+1}^{(i^*)}, \\ v_{k+1} &:= w_{k+1}. \end{aligned}$$

3 If $|v_{k+1} - v_k| < \epsilon$, stop, output $\mathcal{C} := \mathcal{F} \times_1 (A_{k+1}^{(1)})^T \times_2 (A_{k+1}^{(2)})^T \cdots \times_d (A_{k+1}^{(d)})^T$; Otherwise, set $k := k + 1$, and go to Step 1.

By the convergence property of the MBI method discussed in Chapter 2, Algorithm TD1 guarantees to converge to a stationary point for the Tucker decomposition, which is not the case for many other algorithms, e.g., HOSVD method and HOOI method.

6.3 Tucker Decomposition with Unknown Number of Components

In this section, we propose a new model for the Tucker decomposition without pre-specifying the size of the core. In fact, the size of each mode is no longer a constant number, it is a variable that also needs to be determined, which is a key issue in this model. Subsequently we propose algorithms for solving this new model as well.

6.3.1 Problem Formulation

Given a tensor $\mathcal{F} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, our goal is to find a small-sized (low rank) core tensor $\mathcal{C}$ to express $\mathcal{F}$. We are interested in determining the rank of each mode for the core tensor $\mathcal{C}$ as well as the best approximation of $\mathcal{F}$. The new Tucker decomposition model is described as follows.

Suppose the i-rank of $\mathcal{C}$ is r_i for $i = 1, 2, \ldots, d$. Clearly, we have integers $1 \leq r_i \leq n_i$ for $i = 1, 2, \ldots, d$. Unlike the general Tucker decomposition, r_i's are now decision variables which need to be determined. Let c be a given constant for the summation of all i-rank variables, i.e., $\sum_{i=1}^{d} r_i = c$, which in general prevent r_i being too large. In order to determine the allocation for r_i's in the total number c, we may be interested in solving the new Tucker decomposition problem as

$$
\begin{aligned}
(NTD_{min}) \quad \min \quad & \|\mathcal{F} - \mathcal{C} \times_1 A^{(1)} \times_2 A^{(2)} \cdots \times_d A^{(d)}\| \\
\text{s.t.} \quad & \mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_d}, \\
& A^{(i)} \in \mathbb{R}^{n_i \times r_i} \text{ and columnwise orthogonal}, i = 1, 2, \ldots, d, \\
& r_i \in \mathbb{Z}, 1 \leq r_i \leq n_i, i = 1, 2, \ldots, d, \\
& \textstyle\sum_{i=1}^{d} r_i = c.
\end{aligned}
$$

It is not an easy task to solve (NTD_{min}), since the first two constraints of (NTD_{min}) combine the block variables (e.g., $A^{(i)}$) and i-rank variables r_i together. A straightforward way is to separate these variables, and we introduce d more block variables $Y^{(i)} \in \mathbb{R}^{m_i \times m_i}$ with $m_i = \min\{n_i, c\}$ for $i = 1, 2, \ldots, d$, where

$$
Y^{(i)} = \text{Diag}\,(\boldsymbol{y}^{(i)}), \boldsymbol{y}^{(i)} \in \{0, 1\}^{m_i}, \text{ and } \sum_{j=1}^{m_i} y_j^{(i)} = r_i.
$$

Therefore, we may reformulate (NTD_{min}) by adapting the equivalence between the

Tucker decomposition models (TD_{min}) and (TD_{max}), as follows:

$$
\begin{aligned}
(NTD_{max}) \quad \max \quad & \|\mathcal{F} \times_1 \left(A^{(1)}Y^{(1)}\right)^T \times_2 \left(A^{(2)}Y^{(2)}\right)^T \cdots \times_d \left(A^{(d)}Y^{(d)}\right)^T\| \\
\text{s.t.} \quad & A^{(i)} \in \mathbb{R}^{n_i \times m_i} \text{ and columnwise orthogonal}, i = 1, 2, \ldots, d, \\
& \boldsymbol{y}^{(i)} \in \{0,1\}^{m_i},\ i = 1, 2, \ldots, d, \\
& \textstyle\sum_{j=1}^{m_i} y_j^{(i)} \geq 1,\ i = 1, 2, \ldots, d, \\
& \textstyle\sum_{i=1}^{d} \sum_{j=1}^{m_i} y_j^{(i)} = c.
\end{aligned}
$$

Remark that in (NTD_{max}) the variables $r_1, r_2, \cdots, r_d$ are already replaced by $y_j^{(i)}$. Denote $\mathcal{X} := \mathcal{F} \times_1 \left(A^{(1)}Y^{(1)}\right)^T \times_2 \left(A^{(2)}Y^{(2)}\right)^T \cdots \times_d \left(A^{(d)}Y^{(d)}\right)^T \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_d}$. If the feasible solution for block variables $Y^{(1)}, Y^{(2)}, \cdots, Y^{(d)}$ are all in the form of

$$
\hat{Y}^{(i)} = \text{Diag}\,(\underbrace{1, \cdots, 1}_{r_i}, \underbrace{0, \cdots, 0}_{m_i - r_i}),\ i = 1, 2, \ldots, d, \tag{6.1}
$$

i.e., the first r_i components of $\hat{\boldsymbol{y}}^{(i)}$ are all ones, and the rest are all zeros, then the size of the tensor $\mathcal{X}$ can be reduced to $r_1 \times r_2 \times \cdots \times r_d$ by deleting all the tails of zero components in all modes, and the rank of the tensor $\mathcal{X}$ in each mode is equal to $r_1, r_2, \cdots, r_d$, respectively. This observation, however establishes an equivalence between (NTD_{min}) and (NTD_{max}). As we shall see in the next subsection, we may force the feasible solution $Y^{(i)}$ to be in the form of equation (6.1), and then construct a core tensor with rank $r_1 \times r_2 \times \cdots \times r_d$, which is exactly the same size of the core tensor $\mathcal{C}$ to be optimized in (NTD_{min}).

6.3.2 Implementing the MBI Method on Tucker Decomposition with Unknown Number of Components

Let us now focus on the model for the Tucker decomposition with unknown number of components in the maximization form, i.e., (NTD_{max}). Recall that the separable structure of (G) in Chapter 2 is required to implement the MBI method. Therefore,

we move the equality constraint of (NTD_{max}) to its objective function. To be specific, let $\lambda \geq 0$ be a penalty parameter, and define the following penalty function model

$$\begin{aligned}(PTD) \quad \max \quad & \|\mathcal{F} \times_1 \left(A^{(1)}Y^{(1)}\right)^T \times_2 \left(A^{(2)}Y^{(2)}\right)^T \cdots \times_d \left(A^{(d)}Y^{(d)}\right)^T\|^2 \\ & -\lambda \left(\sum_{i=1}^{d}\sum_{j=1}^{m_i} y_j^{(i)} - c\right)^2 \\ \text{s.t.} \quad & A^{(i)} \in \mathbb{R}^{n_i \times m_i} \text{ and columnwise orthogonal}, i = 1, 2, \ldots, d, \\ & \boldsymbol{y}^{(i)} \in \{0,1\}^{m_i},\ i = 1, 2, \ldots, d, \\ & \textstyle\sum_{j=1}^{m_i} y_j^{(i)} \geq 1,\ i = 1, 2, \ldots, d.\end{aligned}$$

Let us denote the penalty function (i.e., the objective function of (PTD)) to be $p(\lambda, A^{(1)}, A^{(2)}, \cdots, A^{(d)}, Y^{(1)}, Y^{(2)}, \cdots, Y^{(d)})$. We are now ready to apply the MBI method for (PTD) since the block constraints are separable.

Before presenting formal algorithms for (PTD), let us first consider the subproblems which may come up in the procedure of the MBI method. Without loss of generality, we wish to optimize $A^{(1)}$ and $Y^{(1)}$ while all other block variables are fixed, i.e.,

$$\begin{aligned}(PTD_1) \quad \max \quad & \|\left(A^{(1)}Y^{(1)}\right)^T W^{(1)}\|^2 - \lambda \left(\sum_{j=1}^{m_1} y_j^{(1)} + \bar{c}_1\right)^2 \\ \text{s.t.} \quad & A^{(1)} \in \mathbb{R}^{n_1 \times m_1} \text{ and columnwise orthogonal}, \\ & \boldsymbol{y}^{(1)} \in \{0,1\}^{m_1}, \\ & \textstyle\sum_{j=1}^{m_1} y_j^{(1)} \geq 1,\end{aligned}$$

where $W^{(1)} := F_{(1)}\left(A^{(d)}Y^{(d)} \otimes A^{(d-1)}Y^{(d-1)} \otimes \cdots \otimes A^{(2)}Y^{(2)}\right)$ and $\bar{c}_1 := \sum_{i \neq 1}\sum_{j=1}^{m_i} y_j^{(i)} - c$.

The above model can be handled in the following way. First, as optimizing $A^{(1)}$ is irrelevant to the penalty term, we may get the optimal solution $\bar{A}^{(1)}$ as the m_1 leading left singular vectors of matrix $W^{(1)}$ using SVD. Let $V_1 = (\bar{A}^{(1)})^T W^{(1)}$, and denote $w_j\,(j = 1, 2, \ldots, m_l)$ to be the square of 2-norm of the j-th row vector for matrix V_1. Next, we search for the optimal $Y^{(1)}$ for given optimal $A^{(1)}$. Observe that $\left(y_j^{(1)}\right)^2 = y_j^{(1)}$

for $j = 1, 2, \ldots, m_1$, this issue can be casted in solving the following problem

$$\begin{array}{lll} (PTD_{\boldsymbol{y}^{(1)}}) & \max & -\lambda \left(\sum_{j=1}^{m_1} y_j^{(1)}\right)^2 + \hat{\boldsymbol{c}}^T \boldsymbol{y}^{(1)} \\ & \text{s.t.} & \boldsymbol{y}^{(1)} \in \{0,1\}^{m_1}, \\ & & \sum_{j=1}^{m_1} y_j^{(1)} \geq 1, \end{array}$$

where $\hat{\boldsymbol{c}} = \boldsymbol{w} - 2\lambda\, \bar{c}_1\, \boldsymbol{e}$ with $\boldsymbol{e} = (1, 1, \cdots, 1)^T$ and $\boldsymbol{w} = (w_1, w_2, \cdots, w_{m_1})^T$. Although $(PTD_{\boldsymbol{y}^{(1)}})$ looks like combinatorial, it is solvable in polynomial-time. This is because the possible values for $\sum_{j=1}^{m_1} y_j^{(1)}$ are in the set of $\{1, 2, \cdots, m_1\}$, and for any fixed $\sum_{j=1}^{m_1} y_j^{(1)}$, deciding optimal $\boldsymbol{y}^{(1)}$ can be done by the greedy algorithm. Thus we only need to do m_1 trials and pick the best solution.

From the above discussions, we know that the optimal solution $Y^{(1)}$ will automatically satisfy the formation (6.1), due to the method of selecting optimal $A^{(1)}$ of (PTD_1), and we can update $A^{(1)}$ and $Y^{(1)}$ simultaneously in the Maximum Improvement step of the MBI method for a given penalty parameter λ. To summarize, the whole procedure for solving (NTD_{max}) using the MBI method and penalty function method (cf. [12, 102]) is as follows.

Algorithm TD2.

The MBI method for the Tucker decomposition with unknown number of components.

- **Input:** Tensor $\mathcal{F} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ and parameters $c > 0, \sigma > 0$.
- **Output:** Core tensor $\mathcal{C} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_d}$.

0 Choose a penalty parameter $\lambda_0 > 0$ and a feasible solution $(A_0^{(1)}, A_0^{(2)}, \cdots, A_0^{(d)}, Y_0^{(1)}, Y_0^{(2)}, \cdots, Y_0^{(d)})$ with $A_0^{(i)} \in \mathbb{R}^{n_i \times m_i}$ and $\boldsymbol{y}_0^{(i)} \in \{0,1\}^{m_i}$ for $i = 1, 2, \ldots, d$.

Compute initial objective value $v_0 := p(\lambda, A_0^{(1)}, A_0^{(2)}, \cdots A_0^{(d)}, Y_0^{(1)}, Y_0^{(2)}, \cdots, Y_0^{(d)})$.

Set $k := 0, l := 0$.

1 For each $i = 1, 2, \ldots, d$, solve

$$\begin{aligned}(PTD_i) \quad \max \quad & p(\lambda_l, A_k^{(1)}, \cdots, A_k^{(i-1)}, A^{(i)}, A_k^{(i+1)}, \cdots, A_k^{(d)}, \\ & Y_k^{(1)}, \cdots, Y_k^{(i-1)}, Y^{(i)}, Y_k^{(i+1)}, \cdots, Y_k^{(d)}) \\ \text{s.t.} \quad & A^{(i)} \in \mathbb{R}^{n_i \times m_i} \text{ and columnwise orthogonal}, \\ & \boldsymbol{y}^{(i)} \in \{0,1\}^{m_i}, \\ & \textstyle\sum_{j=1}^{m_i} y_j^{(i)} \geq 1.\end{aligned}$$

Denote its optimal solution to be $\left(A_{k+1}^{(i)}, Y_{k+1}^{(i)}\right)$, and compute

$$\begin{aligned}w_{k+1}^i \quad := \quad & p(\lambda_k, A_k^{(1)}, \cdots, A_k^{(i-1)}, A_{k+1}^{(i)}, A_k^{(i+1)}, \cdots, A_k^{(d)}, \\ & Y_k^{(1)}, \cdots, Y_k^{(i-1)}, Y_{k+1}^{(i)}, Y_k^{(i+1)}, \cdots, Y_k^{(d)}).\end{aligned}$$

2 Let $w_{k+1} := \max_{1 \leq i \leq d} w_{k+1}^i$ and $i^* = \arg\max_{1 \leq i \leq d} w_{k+1}^i$, and let

$$\begin{aligned}& A_{k+1}^{(i)} := A_k^{(i)}, \quad Y_{k+1}^{(i)} := Y_k^{(i)}, \ \forall\ i \in \{1, 2, \cdots, d\} \backslash \{i^*\}, \\ & A_{k+1}^{(i^*)} := A_{k+1}^{(i^*)}, \quad Y_{k+1}^{(i^*)} := Y_{k+1}^{(i^*)}, \\ & v_{k+1} := w_{k+1}.\end{aligned}$$

3 If $|v_{k+1} - v_k| < \epsilon_1$, go to Step 4; Otherwise, set $k := k + 1$, and go to Step 1.

4 If $\lambda_l\, q(Y_{k+1}^{(1)}, Y_{k+1}^{(2)}, \cdots, Y_{k+1}^{(d)}) < \epsilon_2$, stop and output core tensor $\mathcal{C}$; otherwise, set $\lambda_{l+1} := \sigma \lambda_l$ and $l := l + 1$, and go to Step 1.

We remark that when the algorithm stops, we can shrink the size of core tensor to $(r_1, r_2, \cdots, r_d)$, where r_i is the number of non-zero elements in $Y_{k+1}^{(i)}$.

6.3.3 A Heuristic Approach

In order to save computational efforts, in this subsection we present a heuristic approach to the Tucker decomposition with unknown number of components. We know that if each i-rank ($i = 1, 2, \ldots, d$) of core tensor $\mathcal{C}$ is given, then the problem becomes the traditional Tucker decomposition discussed in Section 6.2, which can be solved by the MBI method. Besides, as we discussed in Section 6.3, the key issue is about how to allocate the constant number c to each i-rank of the core tensor. Thus, the optimal value of (TD_{max}) can be considered as a function of $(r_1, r_2, \cdots, r_d)$. Here, the heuristic approach on distributing the constant number c is again based on the MBI method. Basically, to allocate the i-rank r_i $(i = 1, 2, \ldots, d)$, there are two alternative routines: one is decreasing rank strategy (i.e., we start from large number r_i's and decrease them until $\sum_{i=1}^{d} r_i = c$); the other is increasing rank strategy (i.e., we start from small number r_i's and increase them until $\sum_{i=1}^{d} r_i = c$). The following heuristic approach is based on the decreasing rank strategy.

Algorithm TD3. A heuristic approach.

- **Input:** Tensor $\mathcal{F} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$.
- **Output:** ranks $r_1, r_2, \cdots, r_d$ and core tensor.

0 Choose initial ranks $r_1, r_2, \cdots, r_d$ with $\sum_{i=1}^{d} r_i > c$.

1 Use the MBI method to solve (TD_{max}), and let

$$g(r_1, r_2, \cdots, r_d) := \|\mathcal{F} \times_1 (\hat{A}^{(1)})^T \times_2 (\hat{A}^{(2)})^T \cdots \times_d (\hat{A}^{(d)})^T\|$$

when the MBI method converges to the solution $(\hat{A}^{(1)}, \hat{A}^{(2)}, \cdots, \hat{A}^{(d)})$.

2 For each $i = 1, 2, \ldots, d$, delete the r_i-th column of matrix $\hat{A}^{(i)}$, and compute

$$g(r_1, \cdots, r_{i-1}, s_i, r_{i+1}, \cdots, r_d)$$

with $s_i = r_i - 1$. Let $i^* := \arg\max_{1 \le i \le d} g(r_1, \cdots, r_{i-1}, s_i, r_{i+1}, \cdots, r_d)$.

Update $r_{i^*} := r_{i^*} - 1$.

Repeat this procedure until $\sum_{i=1}^{d} r_i = c$.

3 Use the MBI method to solve (TD_{max}) with $(r_1, r_2, \cdots, r_d)$ satisfying $\sum_{i=1}^{d} r_i = c$.

6.4 Numerical Experiments

In this section, we shall present some preliminary test results for the algorithms proposed for solving the Tucker decomposition with unknown number of components. All the computations are conducted using the same computer as the one conducting the numerical experiments in Section 4.4 of Chapter 4. The supporting software is MATLAB 7.8.0 (R2009a) as a platform. We use MATLAB Tensor Toolbox Version 2.4 [8] whenever tensor operations are called, and also use it to solve the Tucker decomposition with given rank-$(r_1, r_2, \cdots, r_d)$, where the embedded method is the ALS method and the (termination) precision is set to be 10^{-4} by default.

The parameter σ of Algorithm TD2 is set to be 2. The starting points $(A_0^{(1)}, A_0^{(2)}, \cdots, A_0^{(d)})$ are randomly generated, and the starting points $Y_0^{(i)}$ for $i = 1, 2, \ldots, d$, are all set to be $\mathrm{Diag}\,(1, \underbrace{0, 0, \cdots, 0}_{m_i - 1})$. The (termination) precisions are all set to be 10^{-4} in this algorithm.

The initial ranks of Algorithm TD3 are set to be $r_i = \min(n_i, c)$, for $i = 1, 2, \ldots, d$. The starting points $(A_0^{(1)}, A_0^{(2)}, \cdots, A_0^{(d)})$ for solving $(TD_{\max})$ in Step *1* of Algorithm TD3 are randomly generated when using the MBI method and the (termination)

precision is set to be 10^{-2} , while in Step *3* of this algorithm the starting points $(A_0^{(1)}, A_0^{(2)}, \cdots, A_0^{(d)})$ for solving $(TD_{\max})$ is obtained from Step *2* of this algorithm and the (termination) precision is set to be 10^{-4} for the MBI method.

For the given tensor $\mathcal{F}$ and the computed rank-$(r_1, r_2, \cdots, r_d)$ approximation $\hat{\mathcal{F}}$, the *relative square error* (RSE) of the approximation is defined as $\|\mathcal{F} - \hat{\mathcal{F}}\|/\|\mathcal{F}\|$. Here, we use another term *fit* to measure the closeness of the approximation, which is equal to $1 - \|\mathcal{F} - \hat{\mathcal{F}}\|/\|\mathcal{F}\|$. The following list of abbreviations refers to the results summarized in the tables to follow:

Time:	cpu seconds to solve the instance;
ALS:	the command *tucker_als* in MATLAB Tensor Toolbox Version 2.4;
TD2-fit:	compute the fit by using the ALS method with the ranks obtained from Algorithm TD2;
TD3-fit:	compute the fit by using the ALS method with the ranks obtained from Algorithm TD3;
mean(fit):	average fit by running the ALS method 10 times with randomly generated feasible ranks.

We shall test two real three-way array datasets, one is from the Enron e-mail corpus (the dataset was provided to us by Professor Nikos Sidiropoulos), to be called Dataset 1, whose size is $184 \times 184 \times 44$; the other one is 3D Arabidopsis gene expression data (this dataset was provided to us by Professor Xiuzhen Huang), whose size is $2369 \times 6 \times 9$, to be called Dataset 2.

The initial penalty parameter for testing Dataset 1 for Algorithm TD2 is set to be $\lambda_0 = 20$, while it is set to be $\lambda_0 = 100$ for testing Dataset 2. The results of our experiments are summarized in Tables 6.1 and 6.2 for Dataset 1 and Dataset 2,

Table 6.1: Numerical results for Dataset 1

c	Algorithm TD2			Algorithm TD3			ALS		
	(r_1, r_2, r_3)	Time	fit	(r_1, r_2, r_3)	Time	fit	TD2-fit	TD3-fit	mean(fit)
3	$(1, 1, 1)$	0.9963	17.78%	$(1, 1, 1)$	1.8489	18.58%	17.77%	17.77%	17.77%
5	$(2, 2, 1)$	5.7216	19.93%	$(2, 2, 1)$	3.2380	22.74%	19.94%	19.94%	18.81%
10	$(3, 4, 3)$	9.7469	29.68%	$(4, 4, 2)$	2.0878	33.14%	29.68%	27.16%	20.81%
20	$(8, 8, 4)$	20.0316	38.13%	$(8, 9, 3)$	4.6761	43.89%	37.90%	36.47%	25.89%
30	$(12, 12, 6)$	34.5752	42.87%	$(12, 14, 4)$	17.3214	49.74%	42.57%	41.58%	33.09%

Table 6.2: Numerical results for Dataset 2

c	Algorithm TD2			Algorithm TD3			ALS		
	(r_1, r_2, r_3)	Time	fit	(r_1, r_2, r_3)	Time	fit	TD2-fit	TD3-fit	mean(fit)
3	$(1, 1, 1)$	0.6079	85.85%	$(1, 1, 1)$	4.0984	85.96%	85.85%	85.85%	85.85%
5	$(2, 1, 2)$	13.9852	86.66%	$(2, 1, 2)$	7.6941	87.73%	86.74%	86.74%	86.01%
10	$(4, 2, 4)$	14.9092	89.81%	$(3, 5, 2)$	8.2450	94.03%	89.82%	88.26%	86.61%
15	$(7, 3, 5)$	15.793	91.94%	$(6, 5, 4)$	8.4225	95.60%	91.97%	91.47%	89.15%
20	$(10, 4, 6)$	11.8375	93.60%	$(8, 3, 9)$	4.0474	93.25%	93.60%	92.61%	92.71%

respectively. As we observe, the numerical results show the excellent performance of Algorithms TD2 and TD3. Based on these two tables, we observe that the information of ranks r_1, r_2, r_3 provided by Algorithms TD2 and TD3 are more suitable than the randomly generated feasible three-tuple ranks for the ALS method in terms of the fit. Moreover, we observe that Algorithm TD2 appears to work better in combination with the ALS method than Algorithms TD3 does.

Chapter 7

Conclusion and Recent Developments

This thesis is based on an analysis of a specific block-variable improvement method, called the Maximum Block Improvement (MBI) method, for solving generic models. Our MBI method guarantees to converge to a stationary solution, while all other existing algorithms do not have that property (e.g., the BCD method and the ALS method mentioned in Chapter 2, the HOSVD method and the HOOI method mentioned in Chapter 6). Another advantage is that our proposed MBI method can be applied to solve any optimization model with separate block constraints. Due to this flexibility, we are able to apply the method to various applications. One direct application is found in bioinformatics, where we propose a new framework for co-clustering of high-dimensional gene expression data based on tensor optimization model and then apply the MBI method to solve the problem. Then, we focus on some particular polynomial optimization problems. Some equivalence results between the multilinear function optimization models and their homogeneous polynomial optimization counterparts are established, which makes it possible to apply the MBI method to solve the homoge-

neous polynomial optimization models. Hence, the MBI method is capable of finding the best rank-one approximation of a given tensor, whether it is super-symmetric or not. Moreover, the MBI method can be used to find the best multilinear rank approximation of a higher-order tensor under the framework of Tucker decomposition. In the case that the rank information of Tucker decomposition is not given, we devise a model of Tucker decomposition with unknown number of components, where the ranks will be determined when the MBI method stops. Our numerical experiments show that the MBI method is actually very effective for solving those practical problems, and it typically produces high quality solutions.

The results presented in this thesis are mostly based on our research papers [19, 115, 20, 21]. To be specific, Chapter 2 and Chapter 4 are mainly based on [19], Chapter 3 is mainly based on [115], Chapter 5 is mainly based on [20], and Chapter 6 is mainly based on [21]. Since the MBI method can handle generic models if the block variables are separable in the constraints, it creates the opportunities for solving many other practical problems. In fact, there are some follow-up studies to explore the potential capabilities of the MBI method. Very recently, A. Aubry *et al.* [7] present a cognitive approach to design phase-only modulated waveforms sharing a desired ambiguity function in the research area of radar systems, and propose algorithms based on our MBI method to solve a quartic polynomial complex-valued optimization problem. Besides, in the research work of gene expression data, Zhang *et al.* [114] set up a computational framework of co-identification that enables clustering to be multi-dimensional and adaptive (as former approaches restrict single elements, e.g., a gene or a time point, to participate in one cluster or co-cluster), which will facilitate further functional analysis of complex biological systems. And the embedded solving technique for their model is still based on the MBI method. Furthermore, the MBI method can also be applied to solve other related problems, e.g., nonnegative matrix factorization, nonnegative

rank-one tensor decomposition, nonnegative tucker decomposition. In the future, we will study possible extensions of the MBI method and the applications, and understand better why the MBI method and its variants work so well in practice. The potential of the approach is very promising indeed.

Bibliography

[1] A.A. Ahmadi, A. Olshevsky, P.A. Parrilo, and J.N. Tsitsiklis, *NP-hardness of deciding convexity of quartic polynomials and related problems*, Preprint, 2010; also available online from http://arxiv.org/abs/1012.1908v1 59

[2] W. Ai, Y. Huang, and S. Zhang, *New results on Hermitian mtrix rank-one decomposition*, Mathematical Programming, Series A, 128, 253 – 283, 2011. 41

[3] A.A. Alizadeh, M.B. Eisen, R.E Davis, C. Ma, I.S. Lossos, A. Rosenwald, J.C. Boldrick, H. Sabet, T. Tran, X. Yu, J.I. Powell, L. Yang, G. E. Marti, T. Moore, J. Jr Hudson, L. Lu, D.B. Lewis, R. Tibshirani, G. Sherlock, W.C. Chan, T.C. Greiner, D.D. Weisenburger, J.O. Armitage, R. Warnke, R. Levy, W. Wilson, M.R. Grever, J.C. Byrd, D. Botstein, P.O. Brown, and L.M. Staudt, *Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling*, Nature, 403(6769), 503 – 511, 2000. 27

[4] C.A. Andersson, and R. Bro, *Improving the speed of multi-way algorithms: Part I. Tucker3*, Chemometrics and Intelligent Laboratory Systems, 42(1), 93 – 103, 1998. 69, 72

[5] M.F. Anjos and J.B. Lasserre (Eds.), *Handbook of Semidefinite, Conic and Polynomial Optimization*, Springer, New York, 2011, ISBN 978-1-4614-0768-3. 3

[6] C.J. Appellof and E.R. Davidson, *Strategies for analyzing data from video fluorometric monitoring of liquid chromatographic effluents*, Analytical Chemistry, 53, 2053 – 2056, 1981. 4

[7] A. Aubry, A. De Maio, B. Jiang, and S. Zhang, *A cognitive approach for ambiguity function shaping*, in Proceedings of The Seventh IEEE Sensor Array and Multichannel Signal Processing Workshop, 2012. 84

[8] B.W. Bader and T.G. Kolda, *Matlab Tensor Toolbox Version 2.4*, 2010; http://csmr.ca.sandia.gov/~tgkolda/TensorToolbox 47, 80

[9] S. Banach, *Über homogene polynome in* (L^2), Studia Mathematica, 7, 36 – 44, 1938. 40

[10] A. Barmpoutis, B. Jian, B.C. Vemuri, and T.M. Shepherd, *Symmetric Positive 4th Order Tensors and Their Estimation from Diffusion Weighted MRI*, Lecture Notes in Computer Science, 4584, N. Karssemijer and B. Lelieveldt (eds.), Springer, New York, 308 – 319, 2007. 1

[11] A. Barvinok, *A Course in Convexity*, Graduate Studies in Mathematics, 54, American Mathematical Society, Providence, Rhode Island, 2002. 63

[12] D.P. Bertsekas, *Nonlinear Programming*, second edition, Athena Scientific, Belmont, MA, 1999. 13, 17, 52, 77

[13] G. Blekherman, *Convex forms that are not sums of squares*, Preprint, 2009; also available online from http://arxiv.org/abs/0910.0656 59

[14] G. Blekherman, P. Parrilo, and R. Thomas, *Semidefinite Optimization and Convex Algebraic Geometry*, Forthcoming book in the MOS-SIAM Series on Optimization, 2011; also available online from http://www.math.washington.edu/~thomas/frg/book.html 59

[15] J. Bochnak and J. Siciak, *Polynomials and multilinear mappings in topological vector spaces*, Studia Mathematica, 39, 59 – 76, 1971. 40

[16] R. Bro, and H.A.L. Kiers, *A new efficient method for determining the number of components in PARAFAC models*, Journal of Chemometrics, 17(5), 274 – 286, 2003. 5, 70

[17] P.H. Calamai and J.J. Moré, *Projected gradient methods for linearly constrained problems*, Mathematical Programming, 39, 93 – 116, 1987. 52

[18] J.D. Carroll and J.J. Chang, *Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition*, Psychometrika, 35, 283 – 319, 1970. 3

[19] B. Chen, S. He, Z. Li, and S. Zhang, *Maximum block improvement and polynomial optimization*, SIAM Journal on Optimization, 22(1), 87 – 107, 2012. 84

[20] B. Chen, S. He, Z. Li, and S. Zhang, *Logarithmically quasiconvex optimization*, working paper, 2012. 84

[21] B. Chen, Z. Li, and S. Zhang, *Tucker decomposition with unknown number of components*, working paper, 2012. 84

[22] Y. Cheng and G.M. Church, *Biclustering of expression data*, in Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology, 8, 93 – 103, 2000. 20, 27, 31

[23] H. Cho, I.S. Dhillon, Y. Guan, and S. Sra, *Minimum sum-squared residue Co-clustering of gene expression data*, in Proceedings of The fourth SIAM International Conference on Data Mining, 114 – 125, 2004. 20, 29

[24] G. Dahl, J.M. Leinaas, J. Myrheim, and E. Ovrum, *A tensor product matrix approximation problem in quantum physics*, Linear Algebra and Its Applications, 420, 711 – 725, 2007. 1

[25] P. D′haeseleer, *How does gene expression clustering work?* Nature Biotechnology, 23(12), 1499 – 1501, 2005. 20

[26] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein, *Cluster analysis and display of genome-wide expression patterns*, in Proceedings of the National Academy of Sciences of the United States of America, 95, 14863 – 14868, 1998. 20

[27] L. Eldén and B. Savas, *A Newton-Grassmann method for computing the best multi-linear rank-*$(r1, r2, r3)$ *approximation of a tensor*, SIAM Journal on Matrix Analysis and Applications, 31, 248 – 271, 2009. 69

[28] B.S. Everitt, S. Landau, and M. Leese, *Cluster Analysis*, London: Arnold, 2001. 20

[29] A. Ghosh, E. Tsigaridas, M. Descoteaux, P. Comon, B. Mourrain, and R. Deriche, *A polynomial based approach to extract the maxima of an antipodally symmetric spherical function and its application to extract fiber directions from the orientation distribution function in diffusion MRI*, in Proceedings of the Computational Diffusion MRI Workshop (CDMRI'08), New York, 2008. 1, 35, 55

[30] E.F. Gonzalez and Y. Zhang, *Accelerating the Lee-Seung algorithm for nonnegative matrix factorization*, Preprint, March 3, 2005. 14

[31] M. Grant and S. Boyd, *CVX: Matlab Software for Disciplined Convex Programming, version 1.2*, 2010; http://cvxr.com/cvx 47

[32] L. Grippo and M. Sciandrone, *On the convergence of the block nonlinear Gauss-Seidel method under convex constraints*, Operations Research Letters, 26, 127 – 136, 2000. 13

[33] L. Gurvits, *Classical deterministic complexity of Edmonds' problem and quantum entanglement*, in Proceedings of the 35th ACM Symposium on Theory of Computing, 10 – 19, ACM, New York, 2003. 1

[34] R.A. Harshman, *Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis*, UCLA Working Papers in Phonetics, 16, 1 – 84, 1970; also available online from http://publish.uwo.ca/$\sim$harshman/wpppfac0.pdf 3

[35] J.A. Hartigan, *Direct clustering of a data matrix*, Journal of the American Statistical Association, 67(337), 123 – 129, 1972. 20

[36] J.A. Hartigan, *Clustering Algorithms*, New York: John Wiley & Sons, 1975. 20

[37] J. Hastad, *Tensor rank is NP-complete*, Journal of Algorithms, 11, 644 – 654, 1990. 4, 10

[38] S. He, B. Jiang, Z. Li, S. Zhang, *On representation of moments tensor and Hilbert's identity*, working paper, 2011. 63

[39] S. He, M. Li, S. Zhang and Z.Q. Luo, *A nonconvergent example for the Iterative Water-filling algorithm*, Numerical Algebra, Control and Optimization, 1, 147 – 150, 2011. 13

[40] S. He, Z. Li, S. Zhang, *Approximation algorithms for homogeneous polynomial optimization with quadratic constraints*, Mathematical Programming, Series B., 125, 353 – 383, 2010. 2, 35, 37

[41] S. He, Z. Li, and S. Zhang, *General constrained polynomial optimization: an approximation approach*, Technical Report SEEM2009-06, Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, 2009. 2, 35, 37, 51

[42] S. He, Z. Li, and S. Zhang, *Approximation algorithms for discrete polynomial optimization*, Technical Report SEEM2010-01, Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, 2010. 2, 37, 44

[43] Z. He, A. Cichocki and S. Xie, *Efficient method for Tucker3 model selection*, Electronics Letters, 45, 805 – 806, 2009. 70

[44] J.W. Helton and J. Nie, *Semidefinite representation of convex sets*, Mathematical Programming, Series A., 122, 21 – 64, 2010. 59

[45] D. Henrion and J.B. Lasserre, *GloptiPoly: Global optimization over polynomials with Matlab and SeDuMi*, ACM Transactions on Mathematical Software, 29, 165 – 194, 2003. 3

[46] D. Henrion, J.B. Lasserre, and J. Loefberg, *GloptiPoly 3: Moments, optimization and semidefinite programming*, Optimization Methods and Software, 24, 761 – 779, 2009. 3, 47

[47] D. Hilbert, *Über die darstellung definiter formen als summe von formenquadraten*, Mathematische Annalen, 32, 342 – 350, 1888. 59

[48] C.J. Hillar and L.H. Lim, *Most tensor problems are NP hard*, Preprint, 2010; also available online from http://www.stat.uchicago.edu/~lekheng/work/np.pdf 1

[49] J.-B. Hiriart-Urruty, *A new series of conjectures and open questions in optimization and matrix analysis*, ESAIM: Control, Optimisation and Calculus of Variations, 15, 454 – 470, 2009. 40

[50] S. Hochreite, U. Bodenhofer, M. Heusel, A. Mayr, A. Mitterecker, A. Kasim, T. Khamiakova, S.V. Sanden, D. Lin, W. Talloen, L. Bijnens, H.W.H. Ghlmann, Z. Shkedy, D. Clevert, *FABIA: factor analysis for bicluster acquisition*, Bioinformatics, 26(12), 1520 – 1527, 2010. 20

[51] P.M.J. Hof, C. Scherer and P.S.C. Heuberger, *Model-Based Control: Bridging Rigorous Theory and Advanced Technology, Part 1*, 49 – 68, Springer, New York, 2009. 1, 35

[52] M. Ishteva, L. De Lathauwer, P.A. Absil, and S. Van Huffel, *Differential-geometric Newton method for the best rank-*(R_1, R_2, R_3) *approximation of tensors*, Numerical Algorithms, 51(2), 179 – 194, 2009. 69

[53] B. Jiang, Z. Li and S. Zhang, *On cones of nonnegative quartic forms*, Preprint, 2011; also available online from http://www.menet.umn.edu/~zhangs/Reports/2012_JLZ.pdf 59, 62

[54] A. Kapteyn, H. Neudecker, and T. Wansbeek, *An approach to n-mode components analysis*, Psychometrika, 51, 269 – 275, 1986. 69

[55] S. Khot and A. Naor, *Linear equations modulo 2 and the L_1 diameter of convex bodies*, the 48th Annual IEEE Symposium on Foundations of Computer Science, 318 – 328, 2007. 37

[56] H.A.L. Kiers and A. Der Kinderen, *A fast method for choosing the numbers of components in Tucker3 analysis*, British Journal of Mathematical and Statistical Psychology, 56, 119 – 125, 2003. 5, 70

[57] E. de Klerk, *The complexity of optimizing over a simplex, hypercube or sphere: a short survey*, Central European Journal of Operations Research, 16, 111 – 125, 2008. 1

[58] E. de Klerk, M. Laurent, and P.A. Parrilo, *A PTAS for the minimization of polynomials of fixed degree over the simplex*, Theoretical Computer Science, 261, 210 – 225, 2006. 1

[59] E. Kofidis and P.A. Regalia, *On the best Rank-1 approximation of higher order supersymmetric tensors*, SIAM Journal on Matrix Analysis and Applications, 23(3), 863 – 884, 2002. 4, 35, 36, 41, 54

[60] T.G. Kolda, *Multilinear operators for higher-order decompositions*, Tech. Report SAND2006-2081, Sandia National Laboratories, Albuquerque, NM, Livermore, CA, 2006. 9

[61] T.G. Kolda and B.W. Bader, *Tensor decompositions and applications*, SIAM Review, 51, 455 – 500, 2009. 4, 5, 6, 9, 10, 11, 15, 70, 72

[62] P.M. Kroonenberg and J. De Leeuw, *Principal component analysis of three-mode data by means of alternating least squares algorithms*, Psychometrika, 45, 69 – 97, 1980. 4, 69

[63] J.B. Lasserre, *Global optimization with polynomials and the problem of moments*, SIAM Journal on Optimization, 11, 796 – 817, 2001. 3

[64] J.B. Lasserre, *Polynomials nonnegative on a grid and discrete representations*, Transactions of the American Mathematical Society, 354, 631 – 649, 2001. 3

[65] J.B. Lasserre, *Convergent SDP relaxations in polynomial optimization with sparsity*, SIAM Journal on Optimization, 17(3), 822 – 843, 2006. 3

[66] J.B. Lasserre, *A sum of squares approximation of nonnegative polynomials*, SIAM Review, 49(4), 651 – 669, 2007. 3

[67] L. De Lathauwer, *Signal processing based on multilinear algebra*, Ph.D. thesis, K.U. Leuven, E.E. Dept.-ESAT, Belgium, 1997. 10

[68] L. De Lathauwer, B. De Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM Journal on Matrix Analysis and Applications, 21, 1253 – 1278, 2000. 4, 6, 9, 69

[69] L. De Lathauwer, B. De Moor, and J. Vandewalle, *On the best rank-1 and rank-$(R_1, R_2, \cdots, R_N)$ approximation of higher-order tensors*, SIAM Journal on Matrix Analysis and Applications, 21(4), 1324 – 1342, 2000. 4, 14, 36, 41, 69, 72

[70] L. De Lathauwer, and D. Nion, *Decompositions of a higher-order tensor in block terms —- Part III: Alternating least squares algorithms*, SIAM Journal on Matrix Analysis and Applications, 30(3), 1067 – 1083, 2008. 14

[71] D. Leibovici and R. Sabatier, *A singular value decomposition of a k-way array for a principal component analysis of multiway data, pta-k*, Linear Algebra and its Applications, 269, 307 – 329, 1998. 4

[72] J. Levin, *Three-mode Factor Analysis*, Ph.D. thesis, University of Illinois, Urbana, 1963. 68

[73] Z. Li, *Polynomial Optimization Problems—Approximation Algorithms and Applications*, Ph.D. Thesis, The Chinese Univesrity of Hong Kong, June 2011. 2, 41, 44

[74] Z. Li, S. He, and S. Zhang, *Approximation Methods for Polynomial Optimization: Models, Algorithms, and Applications*, Series: SpringerBriefs in Optimization, Springer, New York, 2012.

[75] C. Ling, J. Nie, L. Qi, and Y. Ye, *Biquadratic optimization over unit spheres and semidefinite programming Relaxations*, SIAM Journal on Optimization, 20, 1286 – 1310, 2009. 2, 45

[76] Z. Luo and J. Pang, *Analysis of Iterative Waterfilling algorithm for multiuser power control in digital subscriber lines*, EURASIP Journal on Applied Signal Processing, 2006, 1 – 10, 2006. 13, 14

[77] Z. Luo, J.F. Sturm, and S. Zhang, *Multivariate nonnegative quadratic mappings*, SIAM Journal on Optimization, 14, 1140 – 1162, 2004. 59

[78] Z. Luo and S. Zhang, *A semidefinite relaxation scheme for multivariate quartic polynomial optimization with quadratic constraints*, SIAM Journal on Optimization, 20, 1716 – 1736, 2010. 2

[79] S.C. Madeira, and A.L. Oliveira, *Biclustering algorithms for biological data analysis: A survey*, IEEE/ACM Transactions on Computational Biology and Bioinformatics, 1(1), 24 – 45, 2004. 20

[80] B. Mourrain and J.P. Pavone, *Subdivision methods for solving polynomial equations*, Journal of Symbolic Computation, 44, 292 – 306, 2009. 36

[81] B. Mourrain and P. Trébuchet, *Generalized normal forms and polynomial system solving*. In M. Kauers, eds., in Proceedings of the International Symposium on Symbolic and Algebraic Computation, ACM Press, New York, 253 – 260, 2005. 36

[82] Yu. Nesterov, *Random walk in a simplex and quadratic optimization over convex polytopes*, CORE Discussion Paper, UCL, Louvain-la-Neuve, Belgium, 2003. 1

[83] Yu. Nesterov and A. Nemirovski, *Interior-point polynomial algorithms in convex programming*, SIAM Studies in Applied Mathematics, 1994. 40

[84] Network-Enabled Optimization Systems (NEOS) Server for Optimization, `http://www.neos-server.org/neos`. 2

[85] G. Ni and Y. Wang, *On the best rank-1 approximation to higher-order symmetric tensors*, Mathematical and Computer Modelling, 46, 1345 – 1352, 2007. 43

[86] D. Nion and L. De Lathauwer, *An enhanced line search scheme for complex-valued tensor decompositions*, Application in DS-CDMA, Signal Processing, 88, 749 – 755, 2008. 36

[87] P.A. Parrilo, *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*, Ph.D. Dissertation, California Institute of Technology, 2000. 3

[88] P.A. Parrilo, *Semidefinite programming relaxations for semialgebraic problems*, Mathematical Programming, Series B., 96, 293 – 320, 2003. 3

[89] M.J.D. Powell, *On search directions for minimization algorithms*, Mathematical Programming, 4, 193 – 201, 1973. 13

[90] S. Prajna, A. Papachristodoulou, P.A. Parrilo, *Introducing SOSTOOLS: a general purpose sum of squares programming solver*, in Proceedings of the 41st IEEE Conference on Decision and Control, Las Vegas, USA, 2002. 2

[91] L. Qi, *Eigenvalues of a real supersymmetric tensor,* Journal of Symbolic Computation, 40, 1302 – 1324, 2005. 1, 35, 42

[92] L. Qi, *Eigenvalues and invariants of tensors*, Journal of Mathematical Analysis and Applications, 325, 1363 – 1377, 2007. 35, 42

[93] L. Qi, *The best rank-one approximation ratio of a tensor space*, SIAM Journal on Matrix Analysis and Applications, 32(2), 430 – 442, 2011. 4, 43, 44

[94] L. Qi and K.L. Teo, *Multivariate polynomial minimization and its application in signal processing*, Journal of Global Optimization, 26, 419 – 433, 2003. 1

[95] L. Qi, F. Wang, and Y. Wang, *Z-eigenvalue methods for a global polynomial optimization problem*, Mathematical Programming, Series A., 118, 301 – 316, 2009. 36, 52

[96] M. Rajih, P. Comon, and R.A. Harshman, *Enhanced line search: A novel method to accelerate PARAFAC*, SIAM Journal on Matrix Analysis and Applications, 30(3), 1128 – 1147, 2008. 14, 36

[97] A.M.C. So, *Deterministic approximation algorithms for sphere constrained homogeneous polynomial optimization problems*, Mathematical Programming, Series B., 129, 357 – 382, 2011. 2, 45

[98] S. Soare, J.W. Yoon, and O. Cazacu, *On the use of homogeneous polynomials to develop anisotropic yield functions with applications to sheet forming*, International Journal of Plasticity, 24, 915 – 944, 2008. 1

[99] J.F. Sturm, *SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones*, Optimization Methods and Software, 11 & 12, 625 – 653, 1999. 3

[100] J.F. Sturm and S. Zhang, *On cones of nonnegative quadratic functions*, Mathematics of Operations Research, 28, 246 – 267, 2003. 43, 59

[101] J. Supper, M. Strauch, D. Wanke, K. Harter, A. Zell, *EDISA: extracting biclusters from multiple time-series of gene expression profiles*, BMC Bioinformatics, 8, 334 – 347, 2007. 32

[102] W. Sun, Y. Yuan, *Optimization Theory and Methods: Nonlinear Programming*, Springer Optimization and Its Applicationss, Volume 1, Springer, 2006. 77

[103] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E.S. Lander, and T.R. Golub, *Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation*, in Proceedings of the National Academy of Sciences of the United States of America, 96(6), 2907 – 2912, 1999. 20

[104] S. Tavazoie, J.D. Hughes, M.J. Campbell, R.J. Cho, and G.M. Church, *Systematic determination of genetic network architecture*, Nature Genetics, 22(3), 281 – 285, 1999. 20, 27, 30, 31

[105] M.E. Timmerman, H.A.L. Kiers, *Three-mode principal components analysis: Choosing the numbers of components and sensitivity to local optima*, British Journal of Mathematical and Statistical Psychology, 53, 1 – 16, 2000. 5, 70

[106] P. Tseng, *Convergence of a block coordinate descent method for nondifferentiable minimization*, Journal of Optimizatoin Theory and Applications, 109, 475 – 494, 2001. 12, 13, 14

[107] L.R. Tucker, *Implications of factor analysis of three-way matrices for measurement of change*, in Problems in Measuring Change, C. W. Harris, eds., University of Wisconsin Press, 122 – 137, 1963. 4, 68

[108] L.R. Tucker, *The extension of factor analysis to three-dimensional matrices*, in Contributions to Mathematical Psychology, H. Gulliksen and N. Frederiksen, eds., Holt, Rinehardt, & Winston, New York, 1964. 4, 68

[109] L.R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31, 279 – 311, 1966. 4, 68, 69

[110] H. Waki, S. Kim, M. Kojima, M. Muramatsu, and H. Sugimoto, *Algorithm 883: SparsePOP: A sparse semidefinite programming relaxation of polynomial optimization problems*, ACM Transactions on Mathematical Software, 35(2), 1 – 13, 2008. 3

[111] Y. Wang and L. Qi, *On the successive supersymmetric rank-1 decomposition of higher-order supersymmetric tensors*, Numerical Linear Algebra with Applications, 14, 503 – 519, 2007. 36, 52

[112] S. Wright, *Accelerated Block-Coordinate relaxation for regularized optimization*, SIAM Journal on Optimization, 22, 159 – 186, 2012. 13

[113] Y. Ye and S. Zhang, *New results on quadratic minimization*, SIAM Journal on Optimization, 14, 245 – 267, 2003. 43

[114] S. Zhang, K. Wang, C. Ashby, B. Chen and X. Huang, *A unified adaptive Co-identification analysis for multi-dimensional gene expression data*, working paper, 2012. 21, 70, 84

[115] S. Zhang, K. Wang, B. Chen and X. Huang, *A New Framework for Co-Clustering of Gene Expression Data*, M. Loog et al. eds., PRIB2011, Lecture Notes in Computer Science, 7036, Springer, New York, 1 – 12, 2011. 84

[116] T. Zhang and G.H. Golub, *Rank-one approximation to high order tensors*, SIAM Journal on Matrix Analysis and Applications, 23, 534 – 550, 2001. 35, 41