論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Modeling techniques and algorithms on conic optimization
著者(和文)	田中未来
Author(English)	Mirai TANAKA
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第9561号, 授与年月日:2014年3月26日, 学位の種別:課程博士, 審査員:中田　和秀,水野　眞治,飯島　淳一,梅室　博行,福田　光浩
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第9561号, Conferred date:2014/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Modeling techniques and algorithms on conic optimization

Mirai Tanaka

Doctoral thesis

February, 2014

Department of Industrial Engineering and Management,
Graduate School of Decision Science and Technology,
Tokyo Institute of Technology

Abstract

This thesis reviews some modeling techniques and algorithms on conic optimization. In particular, doubly nonnegative optimization problem, well-conditioned matrix approximation problems, and ship navigation problem are covered. Each of them can be formulated as a (mixed-integer) symmetric cone optimization problem, which is theoretically considered as one of the largest class of optimization problems that can be efficiently solved. In practice, however, it is difficult to solve large instances. To solve such ones, exploitation of its structure and specialized algorithms have been proposed.

One difficulty in solving doubly nonnegative optimization problem is that the problem is often degenerated and the degeneracy causes numerical errors in a solution algorithm. To overcome this difficulty, a reduction method for the problem has been investigated. The method reduces the problem into a smaller one with less degeneracy. Another difficulty is that an equivalent symmetric cone optimization problem becomes large. Hence, an inexact primal-dual path-following method has been proposed.

Symmetric cone optimization problems equivalent to well-conditioned matrix approximation problems also become large. For a basic problem, the reformulation into a univariate optimization problem and a binary search for the resulting problem has been proposed. For extended problems, a projection algorithm that employs the basic problem as a subproblem has also been invented.

Ship navigation problem can be modeled as a mixed-integer second-order cone optimization problem. Even though we can apply a general-purpose solver to this problem, it is still formidable to solve large instances since it is based on the branch-and-bound procedure. A perspective reformulation to accelerate the convergence of the solver and a specialized algorithm named the route generation algorithm has been proposed.

By reviewing these approaches and numerical results, we discuss the importance of exploitation of structures.

Acknowledgements

I would like to express the deepest appreciation to my supervisor, Prof. Kazuhide Nakata. He introduced me to the field of the symmetric cone optimzation. He also provided the best environment for my study and many oppotunities.

I also owe a special debt to my collaborators, Prof. Hayato Waki of Kyusyu University and Dr. Kazuhiro Kobayashi of National Maritime Research Institute for their valuable comments and advice.

Finally, I would like to thank to my precious one from the bottom of my heart. Her moral support has been greatly encouraging me. I could not complete this thesis without her emotional support.

Mirai Tanaka
Tokyo, February, 2014

Contents

Chapter 1

Introduction

Mathematical optimization is a method to find an optimal solution from a set of feasible solutions. It is one of the most important fields in applied mathematics. In fact, it has been applied to various fields on science and engineering such as operations research, management science, informatics, statistics, economics, mechanics and control engineering, chemistry, *etc.* Mathematical optimization has made an impact on not only academic research but also our society. In particular, it has been of great importance on utilization of data in business as we have been able to handle large data sets.

For complex decision making, we must often consider nonlinear objective function and constraints and/or discrete variables. By using nonlinearity, we can model economies of scale, diminishing marginal utility, *etc.*, and can use the product of variables. In fact, cost and utility are often considered to be a concave function of scale. By using discreteness, we can model not only number of pieces but also choices of a decision-maker. In addition, with rapid globalization of today's world, it has been necessary to solve large-scale optimization problems.

As an example, let us consider a navigation of a vehicle. Suppose a situation that we make a vehicle reach at the destination within a designated time and reduce the fuel consumption by choosing the route and controlling the speed of a vehicle. The travelling time from the start to the goal is usually represented as a nonlinear function of the speed. For the simplest situation, it is obtained by dividing the distance by the speed. In addition, the fuel consumption is often represented as a nonlinear function. In fact, the fuel consumption per unit time of a ship is known to be represented as a cubic function of the speed. Of course, the choice of the route is represented by discrete variables.

Next, we deal with other examples. We often encounter optimization problems relevant to eigenvalues, singular values, and semidefiniteness of matrices. Singular values and eigenvalues are arising from various applications. For example, numerical stability of a matrix is often measured by the largest and the smallest eigenvalues or singular values. When the stability of a matrix is often required, an optimization problem with constraints on eigenvalues or singular values arises. Semidefinite matrices are also arising in many fields of science and engineering. For example, a covariance matrix must be positive semidefinite. In addition, the stability of a differential equation is sometimes represented as the semidefiniteness of a matrix. Furthermore, the semidefinite

constraint is often used for the relaxation of difficult optimization problems. Note that semidefiniteness is closely related to eigenvalues and singular values. In fact, a matrix is positive semidefinite if and only if its minimum eigenvalue is nonnegative.

Such problems can often be represented as (mixed-integer) conic optimization problem. Conic optimization problem, defined later in mathematical term, is an linear optimization problem over a cone. Since nonlinear constraints can be often expressed as conic constraints, a number of complicated optimization problems can be modeled as conic optimization problems. The fuel consumption of a ship per unit distance is often represented as a convex quadratic function of a shipping speed. A convex quadratic function can be expressed by using the second-order cone. In addition, a semidefinite constraint is usually regarded as conic constraint since the set of semidefinite matrices forms a closed convex cone.

In particular, symmetric cone optimization problem has been considered to be one of the largest class of continuous optimization problems that we can efficiently solve it. A conic optimization is called a symmetric cone optimization when the cone in the conic constraint is a symmetric cone. For example, the cone of nonnegative vectors, the second-order cone, and the cone of semidefinite matrices are symmetric cones. To solve symmetric cone optimization problem, interior-point methods are efficient algorithm because of its polynomial time convergence. Moreover several great solvers have already been available as open-source software. We can easily utilize them to solve our instances.

In sight of practice, however, they often cannot solve large-scale instances within reasonable computational time. In fact, an optimization problem arising from the real-world usually becomes large. Thus, to efficiently solve such instances, we must elaborate an effective optimization model and an efficient algorithm.

This thesis reviews some modeling techniques and algorithms on conic optimization. Especially, a doubly nonnegative optimization problem, well-conditioned matrix approximation problems, and a ship navigation problem are covered. These problems can be formulated as (mixed-integer) symmetric cone optimization problems. The effective modeling techniques and efficient algorithms shown in later make it possible to solve large-scale instances for each problem. Therefore, they have potential to solve other real-world optimization problems as an extensions.

In this chapter, we define conic optimization problems by reviewing the history on conic optimization and we briefly introduce the latest development that includes our contribution.

1.1 Introduction to conic optimization

Linear optimization problem (LP)[1] is the most basic optimization problem. As we know, it is an optimization problem to minimize a linear objective function over polyhedral constraints. More specifically, it is formulated as

$$\left|\begin{array}{ll} \text{minimize} & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \\ & \boldsymbol{x} \geq \boldsymbol{0}, \end{array}\right. \tag{1.1}$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, and $\boldsymbol{c} \in \mathbb{R}^n$ are constants and $\boldsymbol{x} \in \mathbb{R}^n$ is a decision variable. Danzig [23] had studied it and invented the simplex method. The simplex method is an efficient algorithm to solve this problem and is still commonly used. However, Klee and Minty [52] showed an instance of LP for which the simplex method takes exponential iterations.

The first polynomial time algorithm is the ellipsoid method by Khachiyan [51]. Although it is fast algorithm in theory, it was unfortunately inefficient in practice. The first practical polynomial time algorithm was invented by Karmarkar [50]. Karmarkar's algorithm generates a sequence that converges to an optimal solution in the interior of the feasible region. Such algorithms have been called *interior-point algorithm.* From Karmarkar's breakthrough, many researchers had extensively studied interior-point methods. Kojima *et al.* [56] achieved a *primal-dual interior-point algorithm.* This algorithm solves Primal problem (1.1) and its dual below:

$$\left|\begin{array}{ll} \text{maximize} & \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} \\ \text{subject to} & \boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} + \boldsymbol{s} = \boldsymbol{c}, \\ & \boldsymbol{s} \geq \boldsymbol{0}. \end{array}\right.$$

Useful properties that the duality theorem provides have made the primal-dual interior-point algorithms efficient in theory and practice. About the details of primal-dual interior-point methods for LPs, see Wright [100] *etc.*

While many researchers had kept on studying interior-point algorithms for LP, some researchers extend LP to the space of symmetric matrices and the algorithms for it. Alizadeh [1] proposed a primal-dual interior-point method for *semidefinite optimization (SDP)*:

$$\left|\begin{array}{ll} \text{minimize} & \boldsymbol{C} \bullet \boldsymbol{X} \\ \text{subject to} & \mathcal{A}\boldsymbol{X} = \boldsymbol{b}, \\ & \boldsymbol{X} \in \mathcal{S}^n_+, \end{array}\right. \tag{1.2}$$

where $\bullet$ denotes the standard inner product on the space of n-dimensional symmetric matrices $\mathcal{S}^n$ defined by $\boldsymbol{C} \bullet \boldsymbol{X} = \mathrm{tr}(\boldsymbol{C}^{\mathrm{T}}\boldsymbol{X}) = \sum_{i=1}^n \sum_{j=1}^n C_{ij}X_{ij}$, $\mathcal{A}$:

[1] "Linear *optimization*" has also been called "linear *programming*." The author would like to use the former and the newer term in this thesis. However, the author have felt its abbreviation "LO" has not been popular. Hence, the author would like to use the classical abbreviation "LP" in this thesis.

$\mathcal{S}^n \longrightarrow \mathbb{R}^m$ is a linear mapping, *i.e.*, $\mathcal{A} : \boldsymbol{X} \longmapsto (\boldsymbol{A}_1 \bullet \boldsymbol{X}, \ldots, \boldsymbol{A}_m \bullet \boldsymbol{X})^{\mathrm{T}}$, $\boldsymbol{b} \in \mathbb{R}^m$, $\boldsymbol{C} \in \mathcal{S}^n$, and

$$\mathcal{S}^n_+ = \{\boldsymbol{X} \in \mathcal{S}^n : \boldsymbol{x}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{x} \geq 0 \ (\forall \boldsymbol{x} \in \mathbb{R}^n)\}$$

is the cone of n-dimensional positive semidefinite matrices. A dual of (1.2) is the following problem:

$$\left| \begin{array}{ll} \text{maximize} & \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} \\ \text{subject to} & \mathcal{A}^{\mathrm{T}}\boldsymbol{y} + \boldsymbol{S} = \boldsymbol{C}, \\ & \boldsymbol{S} \in \mathcal{S}^n_+, \end{array} \right. \tag{1.3}$$

where $\mathcal{A}^{\mathrm{T}} : \mathbb{R}^m \longrightarrow \mathcal{S}^n$ is the adjoint of $\mathcal{A}$, *i.e.*, $\mathcal{A}^{\mathrm{T}} : \boldsymbol{y} \longmapsto \sum_{i=1}^m y_i \boldsymbol{A}_i$. Algorithms for SDP, including primal-dual interior-point algorithms, has been extensively studied in [18, 43, 45, 55, 57, 66] *etc.*

At the same time, Nesterov and Nemirovskii [70] have achieved theory of polynomial time interior-point algorithms for a conic optimization problem over a proper cone $\mathcal{K}$, where a cone $\mathcal{K}$ is called proper if it has nonempty interior and is closed, convex, and pointed, *i.e.*, $\mathcal{K} \cap (-\mathcal{K}) = \{\boldsymbol{0}\}$. They have considered the following linear optimization problem over a proper cone $\mathcal{K}$:

$$\left| \begin{array}{ll} \text{minimize} & \langle \boldsymbol{c}, \boldsymbol{x} \rangle \\ \text{subject to} & \langle \boldsymbol{a}_i, \boldsymbol{x} \rangle = b_i \quad (i = 1, \ldots, m), \\ & \boldsymbol{x} \in \mathcal{K}, \end{array} \right. \tag{1.4}$$

where $\langle \cdot, \cdot \rangle$ is an appropriate inner product. A dual of (1.4) is the following problem:

$$\left| \begin{array}{ll} \text{maximize} & \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} \\ \text{subject to} & \sum_{i=1}^m y_i \boldsymbol{a}_i + \boldsymbol{s} = \boldsymbol{c}, \\ & \boldsymbol{s} \in \mathcal{K}^*, \end{array} \right.$$

where $\mathcal{K}^*$ is the dual cone of $\mathcal{K}$, *i.e.*, $\mathcal{K}^* = \{\boldsymbol{s} : \langle \boldsymbol{x}, \boldsymbol{s} \rangle \geq 0 \ (\forall \boldsymbol{x} \in \mathcal{K})\}$. Their theory is simplified when $\mathcal{K}$ in (1.4) is a symmetric cone, where cone $\mathcal{K}$ is called symmetric when it is self-dual and homogeneous, *i.e.*, $\mathcal{K}^* = \mathcal{K}$ and for any $\boldsymbol{x}, \boldsymbol{y} \in \mathrm{int}\,\mathcal{K}$ there exists linear transformation L such that $L(\boldsymbol{x}) = \boldsymbol{y}$ and $L(\mathcal{K}) = \mathcal{K}$. In such case, Problem (1.4) is called *symmetric cone optimization problem (SCP)*. Note that SDP is an important subclass of SCP.

Another important subclass of SCP is *second-order cone optimization problem (SOCP):*

$$\left| \begin{array}{ll} \text{minimize} & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \\ & \boldsymbol{x} \in \displaystyle\prod_{k=1}^{K} \mathcal{Q}^{n_k}, \end{array} \right. \tag{1.5}$$

where

$$\mathcal{Q}^n = \left\{ \boldsymbol{x} \in \mathbb{R}^n : x_1 \geq \sqrt{\sum_{i=2}^{n} x_i^2} \right\}$$

denotes the n-dimensional second-order cone and $\prod_{k=1}^{K} Q^{n_k}$ denotes the Cartesian product of $Q^{n_1}, \ldots, Q^{n_K}$. This cone is also called the quadratic cone or the Lorentz cone. A dual of (1.5) is below:

$$\left|\begin{array}{ll} \text{maximize} & \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} \\ \text{subject to} & \boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} + \boldsymbol{s} = \boldsymbol{c}, \\ & \boldsymbol{s} \in \displaystyle\prod_{k=1}^{K} Q^{n_k}. \end{array}\right.$$

Since a second-order cone constraint can be represent as a semidefinite constraint, an SOCP can be reformulated as an SDP. However, we can solve an SOCP more efficiently than the equivalent SDP by utilizing properties on the second-order cone such as the Euclidean Jordan algebra. Nemirovskii and Scheinberg [68] extended the Karmarkar's algorithm to SOCP. Tsuchiya [93, 94] have extended primal-dual interior-point methods to SOCP by employing the Euclidean Jordan algebra.

A symmetric cone is characterized by a Euclidean Jordan algebra. The theory of symmetric cones implies that a symmetric cone must be a direct product of the following cones:

1. The n-dimensional second-order cone Q^n,
2. The cone of $n \times n$ real symmetric semidefinite matrices $\mathcal{S}_+^n$,
3. The cone of $n \times n$ complex Hermitian semidefinite matrices,
4. The cone of $n \times n$ Hermitian semidefinite matrices with quaternion entries,
5. The cone of 3×3 Hermitian semidefinite matrices with octonion entries.

The third and fourth cone can be expressed by the semidefinite cone while the last cone cannot be expressed by the other cones. From the pragmatic viewpoint, the first two cones have importance. Hence, linear optimization problem over the direct product of the second-order cones and the semidefinite cones are called symmetric cone optimization problem in this thesis. The other three cones never appear in the remainder of this thesis. For the theory of symmetric cone, see Faraut and Korányi [34].

A number of SCP solvers have been developed by utilizing the theoretical background of SCP. These solvers enabled us to readily solve SCPs. SDPA [102] and its family [101] and CSDP [10] are popular SDP solvers. Since SOCPs are expressed as SDPs, we can consider these software as SCP solvers. SDPA and its family are the fastest solvers for SDPs and can solve large instances of SDPs since they exploits the sparsity of instances (and utilizing parallel computing). SeDuMi [83] and SDPT3 [91] can handle semidefinite constraints and second-order cone constraints. They are considered to be stable software.

1.2 Recent development and our contribution

Conic optimization has been still extensively studied. The latest developments in this area range from theory to applications. Hence, we regrettably cannot introduce all movements here. In this section, let us briefly see a couple of topics that are strongly concerned with our contribution.

1.2.1 Application to integer optimization

Optimization problems with integer variables have great importance in many fields of engineering. It has been affected recent advances in conic optimization.

It is known that SDP relaxation techniques often provide tight bounds for many NP-hard optimization problems. Hence, SDP relaxation techniques for combinatorial optimization have been studied for a long time [1, 39, 63, 73]. Recently, Burer [16] has shown that a mixed-integer quadratic optimization problem can be reformulated as an conic optimization, so-called completely positive optimization problem. Some researchers [17, 25, 87, 88, 103] have studied *doubly nonnegative optimization* that is linear optimization over the intersection of the cones of semidefinite matrices and nonnegative matrices can be employed as a relaxation problem for the completely positive optimization problem.

Another movement is the development of solvers on mixed-integer SCPs. In fact, Gurobi Optimizer, which is one of the most popular commercial solvers for integer optimization, has been able to handle a *mixed-integer SOCP* (MISOCP) from version 5, the latest version at the time of this thesis. Tanaka and Kobayashi [84] and Kobayashi and Tanaka [54] have considered an optimization problem arising from maritime affair. We call this problem a *ship navigation problem*. They modeled the problem as an MISOCP. Hence, we can apply the commercial solver to this problem.

1.2.2 Specialized algorithms

Since a number of SCP solvers have been developed, we have been able to readily solve SCPs. However, real-world instances are sometimes extremely large, even the great solvers cannot handle them. To solve such large instances, we often develop specialised algorithm by utilizing the structure of the problem.

A doubly nonnegative optimization problem discussed in [17, 25, 87, 88, 103] can be reformulated as an SDP. However, this reformulation sometimes provides a large SDP. Hence, some researchers have studied algorithms for this optimization problem in [17, 25, 87, 103]. In particular, Tanaka *et al.* [87] proposed an inexact primal-dual interior-point algorithm.

The ship navigation problem considered in [54, 84] may be handled with the commercial solver. However, algorithms for MISOCP are still developing and thus still not so fast. Thus, it cannot solve large instances. Kobayashi and Tanaka [54] have proposed an reformulation that would provide a tight

bound for this problem. Tanaka and Kobayashi [84] have proposed an efficient algorithm for this problem.

Tanaka and Nakata [86, 85] have dealt with *well-conditioned matrix approximation problems.* They are optimization problems to find a well-conditioned matrix that is the nearest to a given matrix under constraints and applicable to problems in broad areas of science and engineering, for example, statistics, finance, and signal processing. These problems can also be formulated as symmetric cone optimization problems so that we can apply the SCP solvers. To solve more efficiently them, Tanaka and Nakata [85] have proposed a reformulation of a basic problem which has only a semidefinite constraint and condition number constraint. They reformulated the problem to a univariate convex optimization problem by restricting the class of norms. It can be considered as a modeling technique. In addition, they proposed a binary search for the resulting problem. Tanaka and Nakata [86] have also considered problems with additional constraints. They proposed a projection algorithm for the problems. In their algorithm, we solve the basic problem by the binary search as a subroutine.

1.2.3 Optimization over nonsymmetric cone

Some researchers of theory have studied optimization over nonsymmetric cones in [69, 82, 103] *etc.* Recently it has been revealed that such a optimization problem has great importance. In fact, it contains the completely positive optimization and the doubly nonnegative optimization, and the copositive optimization, *i.e.*, the dual of completely positive optimization as a special case. Furthermore, it is known that geometric optimization problem [14], an entropy maximization problem [82, Section 6.2.3], and many other problems can be modeled as an optimization problem over a nonsymmetric cone.

A doubly nonnegative optimization problem arising from a combinatorial optimization is degenerate as Tanaka *et al.* [87] pointed out. Thus, it is not easy to obtain a solution with high accuracy since the degeneracy causes numerical errors. Tanaka *et al.* [87, 88] have proposed a reduction method to remove the degeneracy. The reduction method is based on facial reduction algorithms [11, 12, 71, 79, 80, 95, 96, 97]. In the reduction method, we need to solve a linear matrix inequality over the dual cone of the doubly nonnegative cone. It is known that the dual cone is the Minkowski sum of the semidefinite cone and the nonnegative cone. By utilizing this property, Tanaka *et al.* [88] proposed to a decompose the linear matrix inequality to a small SDP and LP. This can be regarded as an exploitation of the asymmetricity of the doubly nonnegative cone.

1.3 Structure of this thesis

The remainder of this thesis is structured as follows: In Chapter 2 we consider the doubly nonnegative optimization. We see the reduction of problems

proposed by Tanaka *et al.* [87, 88] and the inexact primal-dual path-following algorithm for the reduced problems proposed by Tanaka *et al.* [87]. In Chapter 3 we consider the well-conditioned matrix approximation. We review the reformulation and the solution methods for a basic problem proposed by Tanaka and Nakata [85] and a solution algorithm for extended problems proposed by Tanaka and Nakata [86]. In Chapter 4 we consider the ship navigation problem. We discuss an MISOCP model derived by Tanaka and Kobayashi [84] and the reformulation proposed by Kobayashi and Tanaka [54]. We also see the solution method proposed in [84]. Chapter 5 is devoted to some concluding remarks.

1.4 Notation

In this thesis, we use the following notation: Vectors are denoted by lowercase boldface letters such as $\boldsymbol{v}$. The i-th component of this vector would be v_i. Matrices are denoted by uppercase boldface letters such as $\boldsymbol{M}$. The (i, j)-th component of this matrix would be M_{ij}. A block matrix

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix}$$

is also denoted by a MATLAB-like notation $(\boldsymbol{A}, \boldsymbol{B}; \boldsymbol{C}, \boldsymbol{D})$. The vector whose j-th element is unity and whose other elements are all zero is denoted by $\boldsymbol{e}_j$ and that whose all elements are unity is denoted by $\boldsymbol{e}$. The cone of n-dimensional symmetric nonnegative matrices is denoted by $\boldsymbol{\mathcal{N}}^n$. For a given symmetric matrix $\boldsymbol{X}$, the smallest and largest eigenvalues of it is denoted by $\lambda_{\min}(\boldsymbol{X})$ and $\lambda_{\max}(\boldsymbol{X})$. The Euclidean norm and the Frobenius norm are denoted by $\|\cdot\|_2$ and $\|\cdot\|_{\mathrm{F}}$. For given two matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$, the Hadamard product $\boldsymbol{X} \circ \boldsymbol{Y}$ is the matrix defined by $(\boldsymbol{X} \circ \boldsymbol{Y})_{ij} = X_{ij}Y_{ij}$ for all i and j. For two given matrices $\boldsymbol{P} \in \mathbb{R}^{k\times l}$ and $\boldsymbol{Q} \in \mathbb{R}^{m\times n}$, the Kronecker product $\boldsymbol{P} \otimes \boldsymbol{Q}$ is the linear mapping defined by $(\boldsymbol{P} \otimes \boldsymbol{Q})\boldsymbol{X} = \boldsymbol{Q}\boldsymbol{X}\boldsymbol{P}^{\mathrm{T}}$. For a given square matrix $\boldsymbol{X}$, the vector whose elements are the diagonal elements of it is denoted by $\mathrm{diag}(\boldsymbol{X})$. Conversely, for a given vector $\boldsymbol{x}$, the diagonal matrix whose diagonal elements are $\boldsymbol{x}$ is denoted by $\mathrm{Diag}(\boldsymbol{x})$. Extending it, for a given matrix $\boldsymbol{X}$, we define a linear mapping $\mathrm{Diag}(\boldsymbol{X})$ as $\mathrm{Diag}(\boldsymbol{X})\boldsymbol{Y} = \boldsymbol{X} \circ \boldsymbol{Y}$.

Chapter 2

Doubly nonnegative optimization

We discuss in this chapter the doubly nonnegative optimization problem. In particular, we deal with reduction methods proposed by Tanaka *et al.* [87, 88] and an inexact primal-dual path-following algorithm proposed by Tanaka *et al.* [87]. Doubly nonnegative optimization problem is often used for a relaxation problem for an NP-hard optimization problem. Especially, a doubly nonnegative relaxation for mixed-binary quadratic optimization proposed by Burer [16, 17] is highly important. We introduce them in Section 2.1. This optimization problem has often degeneracy that causes numerical errors in solution methods. Tanaka *et al.* [87, 88] have proposed reduction methods to eliminate it. We review their reduction methods in Section 2.2. Tanaka *et al.* [87] have also proposed an inexact primal-dual path-following method and preconditioners to accelerate the PSQMR method for solving large instances. These numerical techniques are shown in Section 2.3. Section 2.4 shows numerical results that implies their approaches are effective. Section 2.5 is devoted to concluding remarks.

2.1 Introduction

A symmetric matrix is called *doubly nonnegative (DNN)* if it is positive semidefinite and its all entries are nonnegative. In this chapter we consider the following linear optimization problem over the cone of DNN matrices:

$$\left|\begin{array}{lll} \text{minimize} & \boldsymbol{D} \bullet \boldsymbol{Y} & \\ \text{subject to} & \boldsymbol{B}_i \bullet \boldsymbol{Y} = b_i & (i = 1, \ldots, m), \\ & \boldsymbol{Y} \in \mathcal{S}^n_+ \cap \mathcal{N}^n, & \end{array}\right. \tag{2.1}$$

where $\boldsymbol{Y} \in \mathcal{S}^n$ is a decision variable and $b_1, \ldots, b_m \in \mathbb{R}$ and $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_m, \boldsymbol{D} \in \mathcal{S}^n$ are given constants. We call Problem (2.1) a *doubly nonnegative optimization problem*.

In recent studies [16, 17, 74, 75, 104], some approaches that utilize DNN optimization for NP-hard optimization problems are proposed. By using DNN optimization, we can obtain a tighter bound for the optimal value of the original problem than existing approaches, such as LP relaxation and SDP relaxation. Ge and Ye [38] have theoretically analyzed the tightness of a DNN relaxation.

Yoshise and Matsukawa [103] have reported a DNN relaxation for the quadratic assignment problem provided an extremely tight bound.

Especially, Burer's copositive reformulation [16] and its doubly nonnegative relaxation [16, 17] for mixed-binary quadratic optimization problem as we will see below is quite important. Let us consider the following mixed-binary quadratic optimization problem:

$$\left|\begin{array}{lll} \text{minimize} & \boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x} + 2\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} & \\ \text{subject to} & \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} = d_i & (i = 1, \ldots, m), \\ & \boldsymbol{x} \geq \boldsymbol{0}, & \\ & x_j \in \{0, 1\} & (j \in B), \end{array}\right. \tag{2.2}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is a decision variable and $\boldsymbol{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m)^{\mathrm{T}} \in \mathbb{R}^{m\times n}$, $\boldsymbol{d} = (d_1, \ldots, d_m)^{\mathrm{T}} \in \mathbb{R}^m$, $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{Q} \in \mathcal{S}^n$, and $B \subset \{1, \ldots, n\}$ are given. Many important problems arising in science and engineering can be formulated as Problem (2.2), such as the quadratic assignment problem, the (quadratic) multidimensional knapsack problem, and the standard quadratic optimization problem. In particular, the standard quadratic optimization problem itself has many important applications, for example, the maximum clique problem and portfolio selection problems. For more details on applications of the standard quadratic optimization problem, see Bomze [7]. In general, finding the global optimal value and solutions of (2.2) is known to be NP-hard. Burer [16] has shown that the optimal value of (2.2) is equal to that of the following completely positive optimization problem under a mild assumption:

$$\left|\begin{array}{lll} \text{minimize} & \boldsymbol{Q} \bullet \boldsymbol{X} + 2\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} & \\ \text{subject to} & \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} = d_i & (i = 1, \ldots, m), \\ & \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{X}\boldsymbol{a}_i = d_i^2 & (i = 1, \ldots, m), \\ & x_j = X_{jj} & (j \in B), \\ & \boldsymbol{Y} = \begin{pmatrix} 1 & \boldsymbol{x}^{\mathrm{T}} \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} \in (C^{1+n})^*, & \end{array}\right. \tag{2.3}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{X} \in \mathcal{S}^n$ are decision variables, and $(C^n)^*$ is the cone of $n \times n$ completely positive matrices defined by

$$(C^n)^* = \left\{ \sum_{i=1}^{r} \boldsymbol{v}_i \boldsymbol{v}_i^{\mathrm{T}} \in \mathcal{S}^n : r \in \mathbb{Z}_+, \boldsymbol{v}_i \geq \boldsymbol{0}\ (i = 1, \ldots, r) \right\}.$$

For more information about the completely positive matrices, see [6, 9]. However, finding the global optimal value and an optimal solution of (2.2) by solving (2.3) is still formidable although (2.3) is a convex optimization problem. In fact, checking whether or not a given matrix is completely positive has been shown by Dickinson and Gijben [31] to be NP-hard.

To find a tight lower bound of the optimal value of (2.2), Kojima and Tunçel [58] proposed a successive semidefinite relaxation for nonconvex quadratic optimization problem which contains (2.2) as a special case. Their approach

has been considered to be difficult to use in practice while it is great in theory. To achieve the same goal, Burer [16, 17] has considered to relax the completely positive constraint in (2.3) by a doubly nonnegative constraint as follows:

$$\left|\begin{array}{lll} \text{minimize} & \boldsymbol{Q} \bullet \boldsymbol{X} + 2\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} & \\ \text{subject to} & \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} = d_i & (i = 1, \ldots, m), \\ & \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{X}\boldsymbol{a}_i = d_i^2 & (i = 1, \ldots, m), \\ & x_j = X_{jj} & (j \in B), \\ & \boldsymbol{Y} = \begin{pmatrix} 1 & \boldsymbol{x}^{\mathrm{T}} \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} \in \mathcal{S}_+^{1+n} \cap \mathcal{N}^{1+n}, & \end{array}\right. \tag{2.4}$$

We call this relaxation technique a *doubly nonnegative (DNN) relaxation*.

DNN optimization problem (2.1) can be reformulated into an equivalent SDP below:

$$\left|\begin{array}{lll} \text{minimize} & \boldsymbol{D} \bullet \boldsymbol{Y} & \\ \text{subject to} & \boldsymbol{B}_i \bullet \boldsymbol{Y} = b_i & (i = 1, \ldots, m), \\ & \mathrm{vech}(\boldsymbol{Y}) = \boldsymbol{y}, & \\ & \begin{pmatrix} \boldsymbol{Y} & \boldsymbol{O} \\ \boldsymbol{O} & \mathrm{Diag}(\boldsymbol{y}) \end{pmatrix} \in \mathcal{S}_+^{n+n(n+1)/2}, & \end{array}\right. \tag{2.5}$$

where, for $\boldsymbol{Y} \in \mathcal{S}^n$, $\mathrm{vech}(\boldsymbol{Y}) \in \mathbb{R}^{n(n+1)/2}$ is the vector obtained by the lower triangular part of $\boldsymbol{Y}$. Thus, we can solve it in polynomial time to any precision, theoretically. We can also employ off-the-shelf SDP solvers [10, 83, 91, 102]. However, we encounter the following difficulty in solving Problem (2.1) in this way. When we convert Problem (2.1) with large-size into SDP (2.5), the size of equivalent SDP (2.5) often becomes too large to be handled. Indeed, the SDP has $m + n(n+1)/2$ linear equality constraints. As a result, we need to solve a linear system of equations with size $m + n(n+1)/2$ in each iteration of primal-dual interior-point methods. Thus, SDP solvers take much computation time to solve the SDP.

In addition, we also meet with another difficulty in trying to solve problem (2.4). Tanaka *et al.* [87] have mentioned that problem (2.4) is highly degenerate. Degeneracy in (2.4) causes numerical instability in primal-dual interior-point methods.

Tanaka *et al.* [87, 88] have considered to overcome these two difficulties.

Tanaka *et al.* [87] proposed a way to reduce the size of problem (2.4) by using its structure. The reduction is based on facial reduction algorithms, which have firstly been proposed by Borwein and Wolkowicz [11, 12] and later simplified by Pataki [71]. See also [79, 80, 95, 96, 97] and references therein for more details. In general, applying the algorithms we obtain a problem that is equivalent to the original problem and has interior feasible solutions. The approach proposed by Tanaka *et al.* [87] can be considered as an application of only one iteration of a facial reduction algorithm. Hence a problem reduced by using their approach may not have any interior feasible solutions. Although their approach does not remove the degeneracy completely, we may hope that the numerical stability of software for solving the resulting problem is improved in practice.

Tanaka *et al.* [87] also proposed an inexact primal-dual path-following method for the reduced problem. Their method is based on an inexact primal-dual path-following method for a convex quadratic SDP proposed by Toh [90]. It exploits the structure of the reduced DNN optimization problem without converting it into an SDP. In their method, we solve large linear systems by using the preconditioned symmetric quasi-minimal residual (PSQMR) method [36]. In their methods, we need to handle ill-conditioned linear systems. For such systems, the convergence of the PSQMR method often becomes slow. To avoid this difficulty, they extended the three preconditioners proposed in [90].

After short, Tanaka *et al.* [88] proposed a numerical reduction method for problem (2.1) in order to remove the degeneracy. Their method is also based on the facial reduction algorithm and can be regarded as an extension of the reduction method proposed in [87]. In fact, the older method cannot be applied to (2.1) other than (2.4) and reduces only the semidefinite constraints. The newer approach is applicable to a general DNN optimization problem (2.1) and reduces both the semidefinite constraint and the nonnegative constraint.

2.2 Reduction of problem

In this section, we review the numerical reduction method for DNN relaxation problem (2.4) proposed by Tanaka *et al.* [87] and that for general DNN optimization problem (2.1) proposed by Tanaka *et al.* [88]. In what follows, we assume that the set of the coefficient matrices $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_m$ in (2.1) or those in (2.4) is linearly independent without loss of generality since one can remove all linearly dependent matrices.

A matrix $\boldsymbol{Y}$ is called an *interior feasible solution* of DNN optimization problem (2.1) if $\boldsymbol{Y}$ is feasible in (2.1), $\boldsymbol{Y}$ is positive definite and all elements in $\boldsymbol{Y}$ are positive. Problem (2.1) is *degenerate* if it has no interior feasible solutions.

As Kobayashi *et al.* [53] mentioned, when we try to solve degenerate problems with a primal-dual interior-point method, we often encounter numerical trouble. Thus, we attempt to convert (2.1) or (2.4) into an equivalent problem that has interior feasible solutions. To this end, we reduce the size of the conic constraints in (2.1) or (2.4). This reduction is based on the facial reduction algorithm proposed in [97, Section 4]. The facial reduction algorithm works for a conic optimization problem and generates an equivalent conic optimization problem that has interior feasible solutions in finitely many iterations. See [11, 12, 71, 79, 80, 95, 96, 97] for more details about facial reduction algorithms in general. Although their methods can be used to make (2.1) or (2.4) less degenerate, they do not work effectively for all degenerate (2.1) or (2.4). In fact, we will see an example where the degeneracy is not be removed by their proposed method in [88] at all in Example 2.6. However, their method is effective for (2.4). We may hope that numerical stability of software for such DNN optimization problems is improved.

In their methods, we reduce DNN optimization problem (2.1) to a more compact form by using a nonzero solution to the following system associated

with (2.1):

$$\left|\begin{array}{lll} \text{find} & \boldsymbol{y} \in \mathbb{R}^m, & \\ & \boldsymbol{S} \in \mathcal{S}^n_+, & \\ & \boldsymbol{T} \in \mathcal{N}^n & \\ \text{such that} & \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} \geq 0, & \\ & \displaystyle\sum_{i=1}^{m} y_i \boldsymbol{B}_i + \boldsymbol{S} + \boldsymbol{T} = \boldsymbol{O}. & \end{array}\right. \tag{2.6}$$

It is clear that $(\boldsymbol{y}, \boldsymbol{S}, \boldsymbol{T}) = (\boldsymbol{0}, \boldsymbol{O}, \boldsymbol{O})$ is a trivial solution to (2.6). The following theorem plays an essential role in their methods.

Theorem 2.1. *We have the following relationships between DNN optimization problem* (2.1) *and its associated linear system* (2.6)*:*

(i) (2.6) *has a nonzero solution such that* $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} = 0$ *if and only if* (2.1) *is degenerate;*

(ii) If (2.6) *has a nonzero solution such that* $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} > 0$*, then* (2.1) *is infeasible.*

Proof. First, we give a proof of the only-if part of (i). Let $(\boldsymbol{y}, \boldsymbol{S}, \boldsymbol{T})$ be a nonzero solution of (2.6) such that $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} = 0$. For any feasible solution $\boldsymbol{Y}$ to (2.1), it holds that

$$\boldsymbol{Y} \bullet \boldsymbol{S} + \boldsymbol{Y} \bullet \boldsymbol{T} = -\boldsymbol{Y} \bullet \left(\sum_{i=1}^{m} y_i \boldsymbol{B}_i \right) = -\sum_{i=1}^{m} (\boldsymbol{B}_i \bullet \boldsymbol{Y}) y_i = -\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y}.$$

By employing $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} = 0$, $\boldsymbol{Y}, \boldsymbol{S} \in \mathcal{S}^n_+$, and $\boldsymbol{Y}, \boldsymbol{T} \in \mathcal{N}^n$, we obtain $\boldsymbol{Y} \bullet \boldsymbol{S} = \boldsymbol{Y} \bullet \boldsymbol{T} = 0$. Also, we have $\boldsymbol{S} \neq \boldsymbol{O}$ or $\boldsymbol{T} \neq \boldsymbol{O}$. Otherwise, we can prove $\boldsymbol{y} = \boldsymbol{0}$ from the linearly independence of the set of $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_m$. It contradicts to that $(\boldsymbol{y}, \boldsymbol{S}, \boldsymbol{T})$ is a nonzero solution. This contradicts that $(\boldsymbol{y}, \boldsymbol{S}, \boldsymbol{T})$ is nonzero. If we have $\boldsymbol{S} \neq \boldsymbol{O}$, then $\boldsymbol{Y}$ cannot be positive definite. On the other hand, if we have $\boldsymbol{T} \neq \boldsymbol{O}$, then all elements of $\boldsymbol{Y}$ cannot be positive simultaneously. Hence, (2.1) is degenerate.

We can prove the if part of (i) by using the result in [71, (2) of Theorem 3.2] or [97, Lemma 3.2]. To apply this result, we need to convert (2.1) to the so-called dual standard form. This is described in [97, Section 2]. See [71, 97] for completeness of the proof of the if part of (i).

To prove (ii), we suppose the contrary that (2.1) has a feasible solution $\boldsymbol{Y}$. Then for any solution $(\boldsymbol{y}, \boldsymbol{S}, \boldsymbol{T})$ of (2.6) such that $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} > 0$, we have $\boldsymbol{Y} \bullet \boldsymbol{S} + \boldsymbol{Y} \bullet \boldsymbol{T} = -\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y}$. Since $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} > 0$, $\boldsymbol{Y}, \boldsymbol{S} \in \mathcal{S}^n_+$ and $\boldsymbol{Y}, \boldsymbol{T} \in \mathcal{N}^n$, this equation implies the contradiction. □

In the remainder of this section, we discuss how to reduce DNN optimization problem (2.1) by using a nonzero solution $(\boldsymbol{y}, \boldsymbol{S}, \boldsymbol{T})$ to system (2.6) such that $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} = 0$. We can see from (i) in Theorem 2.1 that (2.1) is equivalent to the following DNN optimization problem:

$$\left|\begin{array}{lll} \text{minimize} & \boldsymbol{D} \bullet \boldsymbol{Y} & \\ \text{subject to} & \boldsymbol{B}_i \bullet \boldsymbol{Y} = b_i \qquad (i = 1, \ldots, m), & \\ & \boldsymbol{Y} \in \mathcal{S}^n_+ \cap \mathcal{N}^n, & \\ & \boldsymbol{Y} \bullet \boldsymbol{S} = 0, & \\ & \boldsymbol{Y} \bullet \boldsymbol{T} = 0. & \end{array}\right. \tag{2.7}$$

First, we reduce (2.7) into a more compact form by using $\boldsymbol{Y} \bullet \boldsymbol{T} = 0$. This is based on Tanaka *et al.* [88]. Let $I_+ = \{(i,j) : T_{ij} > 0\}$. It follows from $\boldsymbol{Y} \bullet \boldsymbol{T} = 0$ that $Y_{ij} = 0$ for all $(i,j) \in I_+$. Therefore, (2.7) is equivalent to

$$\left|\begin{array}{lll} \text{minimize} & \boldsymbol{D} \bullet \boldsymbol{Y} & \\ \text{subject to} & \boldsymbol{B}_i \bullet \boldsymbol{Y} = b_i & (i = 1, \ldots, m), \\ & \boldsymbol{Y} \in \mathcal{S}^n_+ \cap \mathcal{N}^n, & \\ & \boldsymbol{Y} \bullet \boldsymbol{S} = 0, & \\ & Y_{ij} = 0 & ((i,j) \in I_+). \end{array}\right. \tag{2.8}$$

Furthermore, we reduce (2.8) by using $\boldsymbol{Y} \bullet \boldsymbol{S} = 0$. This is based on [87, 97]. Since $\boldsymbol{S}$ is positive semidefinite, we can decompose it to

$$\boldsymbol{S} = \boldsymbol{R}\boldsymbol{R}^{\mathrm{T}}$$

for some full-rank $\boldsymbol{R} \in \mathbb{R}^{n \times r}$. Then there exists $\boldsymbol{L} \in \mathbb{R}^{n \times (n-r)}$ such that the matrix $\boldsymbol{V} = (\boldsymbol{L}, \boldsymbol{R})$ is nonsingular. Since $\boldsymbol{Y} \bullet (\boldsymbol{R}\boldsymbol{R}^{\mathrm{T}}) = 0$, $\boldsymbol{R}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{R} = \boldsymbol{O}$, and thus for any feasible solution $\boldsymbol{Y}$ of (2.7), we have

$$\boldsymbol{V}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{V} = \begin{pmatrix} \boldsymbol{L}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{L} & \boldsymbol{L}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{R} \\ \boldsymbol{R}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{L} & \boldsymbol{R}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{R} \end{pmatrix} = \begin{pmatrix} \boldsymbol{L}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{L} & \boldsymbol{L}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{R} \\ \boldsymbol{R}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{L} & \boldsymbol{O} \end{pmatrix}.$$

It follows from the semidefiniteness of $\boldsymbol{Y}$ that $\boldsymbol{L}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{R} = \boldsymbol{O}$ and $\boldsymbol{R}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{L} = \boldsymbol{O}$. Then we obtain

$$\boldsymbol{V}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{V} = \begin{pmatrix} \boldsymbol{X} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} \end{pmatrix},$$

for some $\boldsymbol{X} \in \mathcal{S}^{n-r}_+$. Since we have $(\boldsymbol{V}^{-1}\boldsymbol{E}\boldsymbol{V}^{-\mathrm{T}}) \bullet (\boldsymbol{V}^{\mathrm{T}}\boldsymbol{F}\boldsymbol{V}) = \boldsymbol{E} \bullet \boldsymbol{F}$ for all $\boldsymbol{E}, \boldsymbol{F} \in \mathcal{S}^n$, (2.8) is equivalent to the following optimization problem:

$$\left|\begin{array}{lll} \text{minimize} & \boldsymbol{V}^{-1}\boldsymbol{D}\boldsymbol{V}^{-\mathrm{T}} \bullet \boldsymbol{V}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{V} & \\ \text{subject to} & \boldsymbol{V}^{-1}\boldsymbol{B}_i\boldsymbol{V}^{-\mathrm{T}} \bullet \boldsymbol{V}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{V} = b_i & (i = 1, \ldots, m), \\ & \boldsymbol{V}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{V} = \begin{pmatrix} \boldsymbol{X} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} \end{pmatrix}, & \\ & \boldsymbol{X} \in \mathcal{S}^{n-r}_+, & \\ & Y_{ij} = 0 & ((i,j) \in I_+), \\ & Y_{ij} \geq 0 & ((i,j) \in I_0), \end{array}\right. \tag{2.9}$$

where $I_0 = \{(i,j) : T_{ij} = 0\}$. We define $\boldsymbol{U} = (\boldsymbol{I}_{(n-r)}, \boldsymbol{O}_{(n-r)\times r})\boldsymbol{V}^{-1}$, $\boldsymbol{C} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^{\mathrm{T}}$, and $\boldsymbol{A}_i = \boldsymbol{U}\boldsymbol{B}_i\boldsymbol{U}^{\mathrm{T}}$ for $i = 1, \ldots, m$. Then we reformulate (2.9) as follows:

$$\left|\begin{array}{lll} \text{minimize} & \boldsymbol{C} \bullet \boldsymbol{X} & \\ \text{subject to} & \boldsymbol{A}_i \bullet \boldsymbol{X} = b_i & (i = 1, \ldots, m), \\ & \boldsymbol{X} \in \mathcal{S}^{n-r}_+, & \\ & (\boldsymbol{U}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{U})_{ij} = 0 & ((i,j) \in I_+), \\ & (\boldsymbol{U}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{U})_{ij} \geq 0 & ((i,j) \in I_0). \end{array}\right. \tag{2.10}$$

One can convert (2.10) into an SDP. We remark that the resulting problem becomes less degenerate than the original (2.1) although the degeneracy may

not be removed completely. We also note that the coefficient matrices in (2.10) could be denser than those in (2.1) since $\boldsymbol{V}^{-1}$ is often dense. Furthermore, the sizes of positive semidefinite constraint and nonnegative constraints in the resulting problem (2.10) are $n - r$ and $|I_0|$, while they are n and $n(n+1)/2$ in (2.1), respectively. In this sense, (2.9) is smaller than (2.1). If $\boldsymbol{S}$ has a higher rank and/or $\boldsymbol{T}$ contains many nonzero elements, then the resulting problem (2.10) becomes small.

2.2.1 Degeneracy in Burer's doubly nonnegative relaxation problem

Let us see some properties of the feasible region of DNN relaxation problem (2.4) in this section. Specifically, we will see the degeneracy of (2.4).

We define $\boldsymbol{S}_0 \in \mathcal{S}^{1+n}_+$ as

$$\boldsymbol{S}_0 = \sum_{i=1}^{m} \begin{pmatrix} -d_i \\ \boldsymbol{a}_i \end{pmatrix} \begin{pmatrix} -d_i \\ \boldsymbol{a}_i \end{pmatrix}^{\mathrm{T}}. \tag{2.11}$$

We can prove that (2.4) has no interior feasible solutions. In fact, for any feasible solution $\boldsymbol{Y}$ of (2.4), we have

$$\boldsymbol{Y} \bullet \boldsymbol{S}_0 = \sum_{i=1}^{m} \begin{pmatrix} -d_i \\ \boldsymbol{a}_i \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} \xi & \boldsymbol{x}^{\mathrm{T}} \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} \begin{pmatrix} -d_i \\ \boldsymbol{a}_i \end{pmatrix} = \sum_{i=1}^{m} (d_i^2 \xi - 2 d_i \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{x} + \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{X} \boldsymbol{a}_i) = 0.$$

$\boldsymbol{Y}$ is not positive definite because we have $\boldsymbol{Y} \bullet \boldsymbol{S}_0 = 0$ and $\boldsymbol{S}_0$ is positive semidefinite. This implies that any feasible solution $\boldsymbol{Y}$ in (2.4) is not an interior feasible solution.

Remark 2.2. We proved the non-existence of interior feasible solutions in (2.4). In this proof, we used only the positive-semidefiniteness of $\boldsymbol{Y}$. Hence, we can prove in this way that (2.3) and the problem obtained by replacing $(C^{1+n})^*$ by a cone contained in $\mathcal{S}^{1+n}_+$ do not have any interior feasible solutions.

For (2.3), Burer [16] has proposed a reduction method to obtain more compact representation of (2.3). However, Tanaka *et al.* [87] proved that even an SDP relaxation problem for the reduced problem has no interior feasible solutions for $m \geq 2$. To this end, we give a brief introduction of Burer's reduction. In Burer's reduction, we assume that

$$\exists \boldsymbol{y} \in \mathbb{R}^m \text{ such that } \boldsymbol{\alpha} = \sum_{i=1}^{m} y_i \boldsymbol{a}_i \geq \boldsymbol{0}, \sum_{i=1}^{m} y_i d_i = 1 \tag{2.12}$$

for Problem (2.3). Burer [16] proved that under this assumption (2.3) can be

converted into the following equivalent problem:

$$\left|\begin{array}{lll} \text{minimize} & \boldsymbol{Q} \bullet \boldsymbol{X} + 2\boldsymbol{c}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\alpha} & \\ \text{subject to} & \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\alpha} = d_i & (i = 1, \ldots, m), \\ & \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{X}\boldsymbol{a}_i = d_i^2 & (i = 1, \ldots, m), \\ & \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\alpha} = 1, & \\ & (\boldsymbol{X}\boldsymbol{\alpha})_j = X_{jj} & (j \in B), \\ & \boldsymbol{X} \in (C^n)^*. & \end{array}\right. \tag{2.13}$$

This is Burer's reduction. Burer [16] has shown an example of an SDP relaxation problem for (2.13), which does have interior feasible solutions, for $m = 1$. However, the following proposition shows that (2.13) has no interior feasible solutions for $m \geq 2$.

Proposition 2.3. *For any feasible solution* $\boldsymbol{X}$ *to* (2.13) $\mathrm{rank}(\boldsymbol{X}) \leq n - m + 1$.

Proof. For the assumption (2.12), we can assume $y_1 \neq 0$ without loss of generality, and we denote

$$\boldsymbol{A}_1 = \begin{pmatrix} \boldsymbol{\alpha}^{\mathrm{T}} \\ \boldsymbol{a}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{a}_m^{\mathrm{T}} \end{pmatrix}.$$

Note that, since $y_1 \neq 0$, $\boldsymbol{A}_1$ is a full-rank matrix. Let $\boldsymbol{X}$ be a feasible solution in (2.13). Then it follows from $(C^n)^* \subset \mathcal{S}_+^n$ that $\boldsymbol{X}$ is positive semidefinite. In addition, it holds that

$$\boldsymbol{A}_1\boldsymbol{X}\boldsymbol{A}_1^{\mathrm{T}} = \begin{pmatrix} 1 & d_2 & \cdots & d_m \\ d_2 & d_2^2 & * & * \\ \vdots & * & \ddots & * \\ d_m & * & * & d_m^2 \end{pmatrix}.$$

The right hand side of this equality needs to be positive semidefinite. Therefore, the principal minor which consists of the first, i-th, and k-th rows and columns should be nonnegative, *i.e.*,

$$\det \begin{pmatrix} 1 & d_i & d_k \\ d_i & d_i^2 & z_{ik} \\ d_k & z_{ik} & d_k^2 \end{pmatrix} = -(z_{ik} - d_i d_k)^2 \geq 0.$$

This implies that $z_{ik} = d_i d_k$, so it holds that

$$\boldsymbol{A}_1\boldsymbol{X}\boldsymbol{A}_1^{\mathrm{T}} = \begin{pmatrix} 1 \\ d_2 \\ \vdots \\ d_m \end{pmatrix} \begin{pmatrix} 1 \\ d_2 \\ \vdots \\ d_m \end{pmatrix}^{\mathrm{T}}.$$

Since $\boldsymbol{A}_1$ is a full-rank matrix, we obtain

$$\operatorname{rank}(\boldsymbol{X}) \leq n - m + 1.$$

□

Remark 2.4. In this proof, we used only the positive semidefiniteness of $\boldsymbol{X}$. Therefore, we can prove in this way that any relaxation problem of (2.13) obtained by replacing $(C^n)^*$ by a cone contained in $\mathcal{S}^n_+$ has no interior feasible solutions.

2.2.2 Analytical reduction for Burer's doubly nonnegative relaxation problem

We have saw the degeneracy of Burer's DNN relaxation problem (2.4). Here we explain the reduction method proposed by Tanaka *et al.* [87]. Employing an analytical solution to System (2.6), their method reduces (2.4). Tanaka *et al.* [87] have represented (2.4) as follows:

$$\left|\begin{array}{ll}
\text{minimize} & \begin{pmatrix} 0 & \boldsymbol{c}^{\mathrm{T}} \\ \boldsymbol{c} & \boldsymbol{Q} \end{pmatrix} \bullet \begin{pmatrix} \xi & \boldsymbol{x}^{\mathrm{T}} \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} \\
\text{subject to} & \begin{pmatrix} 1 & \boldsymbol{0}^{\mathrm{T}} \\ \boldsymbol{0} & \boldsymbol{O} \end{pmatrix} \bullet \begin{pmatrix} \xi & \boldsymbol{x}^{\mathrm{T}} \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} = 1, \\
& \begin{pmatrix} 0 & \boldsymbol{a}_i^{\mathrm{T}} \\ \boldsymbol{a}_i & \boldsymbol{O} \end{pmatrix} \bullet \begin{pmatrix} \xi & \boldsymbol{x}^{\mathrm{T}} \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} = 2d_i \qquad (i = 1, \ldots, m), \\
& \begin{pmatrix} 0 & \boldsymbol{0}^{\mathrm{T}} \\ \boldsymbol{0} & \boldsymbol{a}_i\boldsymbol{a}_i^{\mathrm{T}} \end{pmatrix} \bullet \begin{pmatrix} \xi & \boldsymbol{x}^{\mathrm{T}} \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} = d_i^2 \qquad (i = 1, \ldots, m), \\
& \begin{pmatrix} 1 & -2\boldsymbol{e}_j^{\mathrm{T}} \\ -2\boldsymbol{e}_j & 4\boldsymbol{e}_j\boldsymbol{e}_j^{\mathrm{T}} \end{pmatrix} \bullet \begin{pmatrix} \xi & \boldsymbol{x}^{\mathrm{T}} \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} = 0 \quad (j \in B), \\
& \begin{pmatrix} \xi & \boldsymbol{x}^{\mathrm{T}} \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} \in \mathcal{S}^{1+n}_+ \cap \mathcal{N}^{1+n}.
\end{array}\right. \tag{2.14}$$

Note that a triple of

$$\boldsymbol{y}_{\mathrm{psd}} = \begin{pmatrix} -\boldsymbol{d}^{\mathrm{T}}\boldsymbol{d} \\ \boldsymbol{d} \\ -\boldsymbol{e} \end{pmatrix}, \tag{2.15}$$

$\boldsymbol{S}_0$ defined in (2.11), and $\boldsymbol{T}_0 = \boldsymbol{O}$ is a nontrivial solution to System (2.6) associated with (2.14). Hence, we can reduce Problem (2.14) as we seen in the previous section. From the definition of $\boldsymbol{S}_0$ defined in (2.11), we can rewrite $\boldsymbol{S}$ as $\boldsymbol{S}_0 = \boldsymbol{R}\boldsymbol{R}^{\mathrm{T}}$, where

$$\boldsymbol{R} = \begin{pmatrix} -d_1 & -d_2 & \ldots & -d_m \\ \boldsymbol{a}_1 & \boldsymbol{a}_2 & \ldots & \boldsymbol{a}_m \end{pmatrix} = \begin{pmatrix} -\boldsymbol{d}^{\mathrm{T}} \\ \boldsymbol{A}^{\mathrm{T}} \end{pmatrix}.$$

Next, we give a way to find such an $\boldsymbol{L}$ such that matrix $\boldsymbol{V} = (\boldsymbol{L}, \boldsymbol{R})$ becomes nonsingular from $\boldsymbol{A}$ and show that we can remove $2m$ linear equalities on $\boldsymbol{X}$

from the reduced problem. Applying elementary row operations to $\boldsymbol{A}$, we can factorize $\boldsymbol{A}$ as follows:

$$\boldsymbol{PA} = \tilde{\boldsymbol{U}} = \begin{pmatrix} \boldsymbol{U}_1 & \boldsymbol{M}_1 & * & * & \cdots & * & * \\ & & \boldsymbol{U}_2 & \boldsymbol{M}_2 & \cdots & * & * \\ & & & & \ddots & * & * \\ & & & & & \boldsymbol{U}_K & \boldsymbol{M}_K \end{pmatrix},$$

where blanks stand for the zero matrix of appropriate size, $\boldsymbol{P}$ is the product of elementary matrices for $\boldsymbol{A}$, $\boldsymbol{U}_k$ is an $r_k \times r_k$ upper triangle matrix and $\boldsymbol{M}_k$ is an $r_k \times c_k$ matrix. Note that $\sum_{k=1}^{K} r_k = m$ and $\sum_{k=1}^{K} c_k = n - m$. For this upper triangular matrix, we set $\tilde{\boldsymbol{L}} \in \mathbb{R}^{n\times(n-m)}$ to be

$$\tilde{\boldsymbol{L}} = \begin{pmatrix} \boldsymbol{O}_{r_1\times c_1} & & & & \\ \boldsymbol{I}_{c_1} & & & & \\ & \boldsymbol{O}_{r_2\times c_2} & & & \\ & \boldsymbol{I}_{c_2} & & & \\ & & \ddots & & \\ & & & \boldsymbol{O}_{r_K\times c_K} \\ & & & \boldsymbol{I}_{c_K} \end{pmatrix}.$$

The matrix $\tilde{\boldsymbol{V}} = (\tilde{\boldsymbol{L}}, \boldsymbol{A}^{\mathrm{T}})$ is non-singular because the matrices $(\tilde{\boldsymbol{L}}, \tilde{\boldsymbol{U}}^{\mathrm{T}})$ and $\boldsymbol{P}$ are non-singular and we have

$$\tilde{\boldsymbol{V}} = (\tilde{\boldsymbol{L}}, \boldsymbol{A}^{\mathrm{T}}) = (\tilde{\boldsymbol{L}}, \tilde{\boldsymbol{U}}^{\mathrm{T}}) \begin{pmatrix} \boldsymbol{I}_{(1+n-m)\times(1+n-m)} & \boldsymbol{O}_{(1+n-m)\times m} \\ \boldsymbol{O}_{m\times(1+n-m)} & \boldsymbol{P}^{-\mathrm{T}} \end{pmatrix}.$$

We set $\boldsymbol{L} = (1, \boldsymbol{0}_{n-m}^{\mathrm{T}}; \boldsymbol{0}_n, \tilde{\boldsymbol{L}})$. Then the matrix $\boldsymbol{V}$ is nonsingular and we have the following simple representation of $\boldsymbol{V}^{-1}$:

$$\boldsymbol{V}^{-1} = \begin{pmatrix} 1 & (\boldsymbol{0}_{n-m}^{\mathrm{T}}, \boldsymbol{d}^{\mathrm{T}})\tilde{\boldsymbol{V}}^{-1} \\ \boldsymbol{0}_n & \tilde{\boldsymbol{V}}^{-1} \end{pmatrix}.$$

It should be noted that $\tilde{\boldsymbol{V}}^{-1}\boldsymbol{a}_i = \boldsymbol{e}_{n-m+i}$ for all $i = 1, \ldots, m$. Then we can prove that

$$\begin{aligned} \boldsymbol{U}\begin{pmatrix} 1 & \boldsymbol{0}^{\mathrm{T}} \\ \boldsymbol{0} & \boldsymbol{O} \end{pmatrix}\boldsymbol{U}^{\mathrm{T}} &= \begin{pmatrix} 1 & \boldsymbol{0}^{\mathrm{T}} \\ \boldsymbol{0} & \boldsymbol{O} \end{pmatrix} = \boldsymbol{E}_{11}, \\ \boldsymbol{U}\begin{pmatrix} 0 & \boldsymbol{a}_i^{\mathrm{T}} \\ \boldsymbol{a}_i & \boldsymbol{O} \end{pmatrix}\boldsymbol{U}^{\mathrm{T}} &= \begin{pmatrix} 2d_i & \boldsymbol{0}^{\mathrm{T}} \\ \boldsymbol{0} & \boldsymbol{O} \end{pmatrix} = 2d_i\boldsymbol{E}_{11}, \\ \boldsymbol{U}\begin{pmatrix} 0 & \boldsymbol{0}^{\mathrm{T}} \\ \boldsymbol{0} & \boldsymbol{a}_i\boldsymbol{a}_i^{\mathrm{T}} \end{pmatrix}\boldsymbol{U}^{\mathrm{T}} &= \begin{pmatrix} d_i^2 & \boldsymbol{0}^{\mathrm{T}} \\ \boldsymbol{0} & \boldsymbol{O} \end{pmatrix} = d_i^2\boldsymbol{E}_{11}. \end{aligned}$$

for $i = 1, \ldots, m$, where $\boldsymbol{E}_{11} = \boldsymbol{e}_1\boldsymbol{e}_1^{\mathrm{T}}$. We can see that the second and third linear equalities in (2.16) are redundant. Let $\boldsymbol{A}_j$ ($j \in B$) be the coefficient matrix of the

fourth constraint in (2.14). Then, we obtain the following problem equivalent to (2.4):

$$\left|\begin{array}{lll} \text{minimize} & \boldsymbol{C} \bullet \boldsymbol{X} & \\ \text{subject to} & \boldsymbol{E}_{11} \bullet \boldsymbol{X} = 1, & \\ & \boldsymbol{A}_j \bullet \boldsymbol{X} = 1 & (j \in B), \\ & \boldsymbol{Y} = \boldsymbol{U}^{\mathrm{T}} \boldsymbol{X} \boldsymbol{U}, & \\ & \boldsymbol{X} \in \mathcal{S}_+^{1+n-m}, & \\ & \boldsymbol{Y} \in \mathcal{N}^{1+n}. & \end{array}\right. \tag{2.16}$$

If $\boldsymbol{e}_j$ is a column of $\tilde{\boldsymbol{L}}$, $\boldsymbol{A}_j$ is the following sparse matrix:

$$\boldsymbol{A}_j = \boldsymbol{U} \begin{pmatrix} 1 & -2\boldsymbol{e}_j^{\mathrm{T}} \\ -2\boldsymbol{e}_j & 4\boldsymbol{e}_j \boldsymbol{e}_j^{\mathrm{T}} \end{pmatrix} \boldsymbol{U}^{\mathrm{T}} = \begin{pmatrix} 1 & -2\boldsymbol{e}_j^{\mathrm{T}} \\ -2\boldsymbol{e}_j & 4\boldsymbol{e}_j \boldsymbol{e}_j^{\mathrm{T}} \end{pmatrix}$$

since we have $\tilde{\boldsymbol{V}}^{-1} \boldsymbol{e}_j = \boldsymbol{e}_j$. Otherwise, $\boldsymbol{A}_j$ may become a dense matrix.

Remark 2.5. To obtain a tighter bound, Burer [16] proposed to add linear equality constraints

$$\boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{X} \boldsymbol{a}_k = d_i d_k$$

for $i = 1, \ldots, m$ and $k = 1, \ldots, m$ to DNN relaxation problem (2.4). Then, we obtain the following DNN relaxation problem:

$$\left|\begin{array}{lll} \text{minimize} & \boldsymbol{Q} \bullet \boldsymbol{X} + 2\boldsymbol{c}^{\mathrm{T}} \boldsymbol{x} & \\ \text{subject to} & \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{x} = d_i & (i = 1, \ldots, m), \\ & \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{X} \boldsymbol{a}_k = d_i d_k & (i = 1, \ldots, m; k = 1, \ldots, m), \\ & x_j = X_{jj} & (j \in B), \\ & \boldsymbol{Y} = \begin{pmatrix} 1 & \boldsymbol{x}^{\mathrm{T}} \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} \in \mathcal{S}_+^{1+n} \cap \mathcal{N}^{1+n}, & \end{array}\right. \tag{2.17}$$

However, by using the discussion in this subsection, we can prove that Problem (2.4) and Problem (2.17) share the same optimal value. In fact, the coefficient matrix of the second constraint is reduced to

$$\boldsymbol{U} \begin{pmatrix} 0 & \boldsymbol{0}^{\mathrm{T}} \\ \boldsymbol{0} & (\boldsymbol{a}_i \boldsymbol{a}_k^{\mathrm{T}} + \boldsymbol{a}_k \boldsymbol{a}_i^{\mathrm{T}})/2 \end{pmatrix} \boldsymbol{U}^{\mathrm{T}} = \begin{pmatrix} d_i d_k & \boldsymbol{0}^{\mathrm{T}} \\ \boldsymbol{0} & \boldsymbol{O} \end{pmatrix} = d_i d_k \boldsymbol{E}_{11}.$$

Hence, applying the reduction method discussed in this subsection to Problem (2.17), we obtain Problem (2.16).

Preliminary experiments

Here, we show the results of preliminary experiments to determine the effectiveness of the reduction method presented in [87]. They solved the following SDPs of the form of (2.5) equivalent to (2.14) and (2.16) by using SDPA version 7.3.1 linked with GotoBLAS 2 version 1.13 with the default parameters. All experiments were executed on a computer with Red Hat Enterprise Linux 5.5, Intel Xeon CPU W3520 with 2.66 GHz (4 CPUs), and 24 GB memory.

In this experiment, we solved (2.4) and (2.16) for randomly generated (2.2) to which we added one slack variable $s \geq 0$ and one linear equality $\sum_{j \in B} x_j + s = |B|$. This introduction of the slack variable and the linear equality have been proposed in [8]. Once they have been added, (2.2) satisfies the so-called weak key condition, which has been proposed in [8] as a substitute for the so-called key condition. Although Burer [16] proved that, under the key condition, the optimal value of (2.2) is equal to that of (2.3), (2.3) and its DNN relaxation problem obtained from the resulting (2.2) often become critically large in scale. In fact, to make (2.2) satisfy the key condition, we must introduce $|B|$ slack variables $s_j \geq 0$ and $|B|$ linear equalities $x_j + s_j = |B|$ for $j \in B$.

The DIMACS errors [65] `err1` to `err6` are shown in Table 2.1. The DIMACS errors of a solution $(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{S})$ to a primal-dual pair of SDPs (1.2)–(1.3) are defined by

$$\texttt{err1} = \frac{\|\boldsymbol{b} - \mathcal{A}\boldsymbol{X}\|_2}{1 + \|\boldsymbol{b}\|_{\max}}, \qquad \texttt{err2} = \max\left\{0, \frac{-\lambda_{\min}(\boldsymbol{X})}{1 + \|\boldsymbol{b}\|_{\max}}\right\},$$

$$\texttt{err3} = \frac{\|\boldsymbol{C} - \mathcal{A}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{S}\|_{\mathrm{F}}}{1 + \|\boldsymbol{C}\|_{\max}}, \qquad \texttt{err4} = \max\left\{0, \frac{-\lambda_{\min}(\boldsymbol{S})}{1 + \|\boldsymbol{C}\|_{\max}}\right\},$$

$$\texttt{err5} = \frac{\boldsymbol{C} \bullet \boldsymbol{X} - \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y}}{1 + |\boldsymbol{C} \bullet \boldsymbol{X}| + |\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y}|}, \qquad \texttt{err6} = \frac{\boldsymbol{X} \bullet \boldsymbol{S}}{1 + |\boldsymbol{C} \bullet \boldsymbol{X}| + |\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y}|},$$

where $\|\cdot\|_{\max}$ denotes the maximum norm. If all values are sufficiently close to zero, then the solution is accurate and one can regard it as an optimal solution. In the case of SDPA, `err2` and `err4` are zero at any time so we omitted them. In most instances, `err5` and `err6`, which correspond to the duality gap, obtained by solving (2.16) are 10^2 to 10^3 times smaller than those obtained by solving (2.4). Although, for some instances, `err1` and `err3`, which correspond to primal and dual feasibility are worse, they are within the allowance. Thus, we can see that we can solve (2.16) more accurately than (2.4).

The computational time is shown in Table 2.2. In this table, "iter" means the number of iterations, "time" indicates how much time the solver takes to solve the problem in seconds, and "nnz" means the number of nonzero elements in coefficient matrices of SDPs equivalent to (2.4) or (2.16). In this table, we can see that the computation for solving (2.16) is much slower than that of (2.4). One of the reasons is considered to be that the coefficient matrices of the SDP equivalent to (2.16) corresponding to the constraint $\mathrm{vech}(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{U}) = \boldsymbol{y}$ are denser than that of the original constraint $\mathrm{vech}(\boldsymbol{Y}) = \boldsymbol{y}$.

Tab. 2.1: DIMACS errors for Problems (2.4) and (2.16)

			Problem (2.14)				Problem (2.16)			
n	m	$\lvert\mathcal{B}\rvert$	err1	err3	err5	err6	err1	err3	err5	err6
50	10	0	1.4×10^{-10}	9.8×10^{-7}	1.8×10^{-5}	1.8×10^{-5}	1.7×10^{-10}	3.6×10^{-15}	1.5×10^{-7}	1.5×10^{-7}
50	10	10	4.7×10^{-9}	8.1×10^{-7}	2.3×10^{-3}	2.3×10^{-3}	8.1×10^{-9}	2.6×10^{-8}	3.7×10^{-6}	3.9×10^{-6}
50	10	20	3.7×10^{-8}	4.8×10^{-7}	2.5×10^{-3}	2.5×10^{-3}	2.0×10^{-8}	8.5×10^{-9}	2.2×10^{-6}	2.4×10^{-6}
50	10	30	7.1×10^{-9}	2.3×10^{-7}	3.2×10^{-3}	3.3×10^{-3}	4.0×10^{-7}	2.0×10^{-9}	2.6×10^{-6}	2.7×10^{-6}
50	20	0	5.0×10^{-8}	1.1×10^{-6}	1.5×10^{-3}	1.5×10^{-3}	9.8×10^{-9}	2.2×10^{-11}	1.4×10^{-5}	1.4×10^{-5}
50	20	10	4.6×10^{-8}	1.2×10^{-6}	1.1×10^{-3}	1.1×10^{-3}	1.9×10^{-5}	5.0×10^{-11}	3.7×10^{-6}	3.8×10^{-6}
50	20	20	3.1×10^{-8}	1.3×10^{-6}	1.0×10^{-3}	1.1×10^{-3}	1.3×10^{-7}	4.3×10^{-13}	2.3×10^{-6}	2.3×10^{-6}
50	20	30	3.6×10^{-8}	2.5×10^{-11}	9.2×10^{-4}	9.8×10^{-4}	1.9×10^{-7}	1.2×10^{-13}	2.5×10^{-6}	2.5×10^{-6}
50	30	0	2.3×10^{-7}	1.3×10^{-6}	1.5×10^{-3}	1.5×10^{-3}	3.9×10^{-9}	4.7×10^{-13}	1.3×10^{-6}	1.3×10^{-6}
50	30	10	3.7×10^{-7}	1.3×10^{-6}	3.0×10^{-4}	1.3×10^{-3}	4.5×10^{-7}	9.8×10^{-13}	1.6×10^{-8}	2.8×10^{-9}
50	30	20	1.6×10^{-7}	8.8×10^{-11}	6.9×10^{-5}	7.8×10^{-4}	8.6×10^{-6}	9.0×10^{-13}	4.5×10^{-8}	7.8×10^{-8}
50	30	30	7.0×10^{-8}	1.8×10^{-10}	-1.4×10^{-4}	3.1×10^{-4}	8.2×10^{-7}	2.1×10^{-12}	9.5×10^{-7}	9.5×10^{-7}

Tab. 2.2: Computational time for Problems (2.4) and (2.16)

			Problem (2.14)			Problem (2.16)		
n	m	$\|B\|$	iter	time [sec.]	nnz	iter	time	nnz
50	10	0	25	2.53	29928	29	29.76	128673
50	10	10	24	2.72	30266	32	38.78	167347
50	10	20	27	3.10	30646	33	40.00	169064
50	10	30	29	3.42	31226	36	43.79	169114
50	20	0	29	3.36	55928	30	36.35	241918
50	20	10	30	3.82	56266	38	50.59	273652
50	20	20	32	4.18	56646	39	52.03	283262
50	20	30	34	4.48	57226	39	52.98	284669
50	30	0	30	3.94	81928	27	17.76	232663
50	30	10	38	5.48	82266	40	27.77	251657
50	30	20	42	6.15	82646	39	27.27	256067
50	30	30	50	7.45	83226	41	28.67	260477

2.2.3 Numerical reduction for general doubly nonnegative optimization problem

Tanaka *et al.* [88] have proposed a way to numerically find a nonzero solution of (2.6). Let us see their method. To find such a solution $(\boldsymbol{y}, \boldsymbol{S}, \boldsymbol{T})$ of (2.6), we find a feasible solution to the following problem:

$$\left| \begin{array}{ll} \text{maximize} & \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} \\ \text{subject to} & \displaystyle\sum_{i=1}^{m} y_i \boldsymbol{A}_i + \boldsymbol{S} + \boldsymbol{T} = \boldsymbol{O}, \\ & \boldsymbol{S} \in \mathcal{S}^n_+, \\ & \boldsymbol{T} \in \mathcal{N}^n. \end{array} \right. \tag{2.18}$$

Since $(\boldsymbol{y}, \boldsymbol{S}, \boldsymbol{T}) = (\boldsymbol{0}, \boldsymbol{O}, \boldsymbol{O})$ is a feasible solution of (2.18), the optimal value is nonnegative. If the optimal value is positive, then it follows from (ii) of Theorem 2.1 that (2.1) is infeasible. Since we can reformulate (2.18) to an SDP, we may obtain a nonzero solution by applying SDP solvers. However, its size is almost the same as that of an SDP equivalent to (2.1), so that finding a nonzero solution to (2.6) is as computationally expensive as solving (2.1).

In their method, we decompose (2.18) to the following two optimization problems to find a nonzero solution of (2.18):

$$\left| \begin{array}{ll} \text{maximize} & \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} \\ \text{subject to} & \displaystyle\sum_{i=1}^{m} y_i \boldsymbol{A}_i + \boldsymbol{T} = \boldsymbol{O}, \\ & \boldsymbol{T} \in \mathcal{N}^n \end{array} \right. \tag{2.19}$$

and

$$\left|\begin{array}{ll} \text{maximize} & \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} \\ \text{subject to} & \displaystyle\sum_{i=1}^{m} y_i \boldsymbol{A}_i + \boldsymbol{S} = \boldsymbol{O}, \\ & \boldsymbol{S} \in \mathcal{S}^n_+. \end{array}\right. \tag{2.20}$$

We remark that LP (2.19) and SDP (2.20) are much smaller than an SDP equivalent to (2.18). We observe that for any feasible solutions $(\boldsymbol{y}_{\mathrm{psd}}, \boldsymbol{S})$ to (2.19) and $(\boldsymbol{y}_{\mathrm{nn}}, \boldsymbol{T})$ to (2.20), $(\boldsymbol{y}_{\mathrm{psd}} + \boldsymbol{y}_{\mathrm{nn}}, \boldsymbol{S}, \boldsymbol{T})$ is also a feasible solution to (2.6). Consequently, we obtain a nonzero solution of (2.18) by finding nonzero optimal solutions of $(\boldsymbol{y}_{\mathrm{psd}}, \boldsymbol{S})$ to (2.19) and/or $(\boldsymbol{y}_{\mathrm{nn}}, \boldsymbol{T})$ to (2.20).

If a nonzero solution to LP (2.19) exists, then we can obtain the densest solution of (2.19) with interior-point methods since the central path of an LP converges to the strong complementary solution, *i.e.*, for optimal solution $(\boldsymbol{y}^*, \boldsymbol{T}^*)$ and $\boldsymbol{Y}^*$ of (2.19) and its dual, we have $Y^*_{ij} T^*_{ij} = 0$ and $Y^*_{ij} + T^*_{ij} > 0$ for all (i, j). See Wright [100] for more details on interior-point methods for LPs.

We can obtain the optimal solution $\boldsymbol{S}$ to SDP (2.20) that has the largest rank in the optimal solutions with an interior-point method since it also converges the analytical center of the optimal set. See de Klerk [26] for more details about interior-point methods for SDPs. As we have seen in the previous section, Tanaka *et al.* [87] pointed out that SDP (2.20) obtained from DNN relaxation problem (2.14) always has an analytical feasible solution. They reduced the semidefinite constraint in (2.1) by using the analytical feasible solution of (2.20), instead of using a solution obtained by SDP software. We use the same approach to find a nonzero solution to SDP (2.20) in the preliminary experiments we will see later.

Even if (2.1) is degenerate or infeasible, *i.e.*, Problem (2.18) has nonzero feasible solutions, we may not be able to construct such nonzero solutions to (2.18) from nonzero solutions to (2.19) and (2.20). We present such an example.

Example 2.6. In DNN optimization problem (2.1), we take

$$\boldsymbol{A}_{\mathrm{psd}} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \qquad \boldsymbol{A}_{\mathrm{nn}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

and $\boldsymbol{A} = \boldsymbol{A}_{\mathrm{psd}} + \boldsymbol{A}_{\mathrm{nn}}$. Also, we take $b \in \mathbb{R}$ and $\boldsymbol{C} \in \mathcal{S}^3$ arbitrarily. From $\pm\boldsymbol{A} \notin \mathcal{S}^n_+$ and $\pm\boldsymbol{A} \notin \mathcal{N}^n$, we can see that problems (2.19) and (2.20) have no nonzero solutions. However, (2.18) has a nonzero solution $(-1, \boldsymbol{A}_{\mathrm{psd}}, \boldsymbol{A}_{\mathrm{nn}})$.

Remark 2.7. Unlike SDP (2.20), we do not have any analytical solutions to LP (2.19). Since the solution obtained by LP software contains numerical errors occurred in floating-point computation, it may not be exact. In the numerical experiments shown in later, we use the following numerical remedy: Let $(\boldsymbol{y}^\dagger, \boldsymbol{T}^\dagger)$ and $\boldsymbol{Y}^\dagger$ be solutions of (2.19) and its dual obtained by LP software with an interior-point method. Then since they are approximation of optimal solutions of (2.19) and its dual, they satisfy $Y^\dagger_{ij} T^\dagger_{ij} \simeq 0$ and $Y^\dagger_{ij}, T^\dagger_{ij} > 0$ for all (i, j). For such solutions, we set I_0 and I_+ as $I_+ = \{(i, j) : Y^\dagger_{ij} < T^\dagger_{ij}\}$ and $I_0 = \{(i, j) : T^\dagger_{ij} < Y^\dagger_{ij}\}$.

Remark 2.8. We can generalize the method discussed in this subsection to the following linear optimization problem over the intersection of multiple closed convex cones $\mathcal{K}_k$ with $k = 1, \ldots, K$:

$$\left|\begin{array}{lll} \text{minimize} & \langle \boldsymbol{c}, \boldsymbol{x} \rangle & \\ \text{subject to} & \langle \boldsymbol{a}_i, \boldsymbol{x} \rangle = b_i & (i = 1, \ldots, m), \\ & \boldsymbol{x} \in \mathcal{K}_k & (k = 1, \ldots, K). \end{array}\right. \tag{2.21}$$

Note that this problem includes problem (2.1) as a special case. For problem (2.21), under a mild assumption, we can formulate the following problem corresponding to (2.6) as follows:

$$\left|\begin{array}{lll} \text{find} & \boldsymbol{y} \in \mathbb{R}^m, & \\ & \boldsymbol{s}_k \in \mathcal{K}_k^* & (k = 1, \ldots, K) \\ \text{such that} & \boldsymbol{b}^{\mathrm{T}} \boldsymbol{y} \geq 0, & \\ & \displaystyle\sum_{i=1}^{m} \boldsymbol{a}_i y_i + \sum_{k=1}^{K} \boldsymbol{s}_k = \boldsymbol{0}, & \end{array}\right.$$

where $\mathcal{K}_k^*$ is the dual cone of $\mathcal{K}_k$. We can decompose this system in a way similar to the method discussed in this subsection.

Preliminary experiments

Tanaka *et al.* [88] verified the effectiveness of the method. To this end, they solved the following three types of SDPs of the form of (2.5) equivalent to the Burer's DNN relaxation for the quadratic assignment problem with SDPA:

original DNN optimization problem (2.1),

only psd the reduced problem (2.10) by using a solution $(\boldsymbol{y}_{\mathrm{psd}}, \boldsymbol{S}_0, \boldsymbol{O})$ to System (2.6) defined by (2.11) and (2.15),

psd + nn the reduced problem (2.10) by using a solution $(\boldsymbol{y}_{\mathrm{psd}} + \boldsymbol{y}_{\mathrm{nn}}, \boldsymbol{S}_0, \boldsymbol{T}_0)$ to System (2.6) where $(\boldsymbol{y}_{\mathrm{nn}}, \boldsymbol{T}_0)$ is an optimal solution to Problem (2.19) corresponding to Problem (2.17) computed by an interior-point method.

They used the instances of the quadratic assignment problem in QAPLIB [19]. They solved them with SDPA 7.3.6 linked with OpenBLAS 0.2.4 with the default parameters. They executed all experiments on a computer with Intel Xeon CPU E5530 with 2.40 GHz (8 CPUs) and 24 GB memory and used 32 threads. We observed from preliminary experiments that the sets of coefficient matrices in the resulting SDP obtained by "psd + nn" was linearly dependent. Therefore, we removed all redundant constraints corresponding to linearly dependent matrices from the SDPs.

Tables 2.3–2.4 depicts the accuracy of the obtained solutions and the computational time. Entire results are shown in [88]. In all the tables, "instance" and "type" denote the names of instance and the types of DNN optimization

problems. "dim psd", "dim nn", and "dim eq" denote the size of positive semidefinite variable matrices, the number of nonnegative constraints, and the number of linear equality constraints, respectively. Note that the size of linear equation systems which SDPA solves at each iteration, so-called Schur complement equation, is equal to "dim eq." `err1` to `err6` denotes DIMACS errors [65]. "time" denotes how much time were spent before SDPA stopped.

We observe in the two tables that for "original" and "only psd", SDPA could not reduce DIMACS errors sufficiently for all instances. Especially, `err5` and `err6` were still large for all instances. In contrast, for "psd + nn", SDPA returned more accurate solutions than "original" and "only psd" in all instances.

Tab. 2.3: Numerical results for small instances of "taiXXa" in QAPLIB [19].

instance	type	dim psd	dim nn	dim eq	err1	err3	err5	err6	time
tai5a	original	26	351	431	1.4×10^{-7}	2.9×10^{-11}	4.1×10^{-2}	4.7×10^{-2}	0.12
	only psd	17	351	377	6.4×10^{-7}	1.4×10^{-11}	5.5×10^{-2}	5.9×10^{-2}	0.11
	psd + nn	17	251	327	1.5×10^{-8}	1.6×10^{-13}	1.0×10^{-7}	9.6×10^{-8}	0.11
tai6a	original	37	703	817	2.9×10^{-7}	3.8×10^{-11}	6.4×10^{-2}	7.1×10^{-2}	0.40
	only psd	26	703	740	1.7×10^{-6}	2.3×10^{-11}	6.5×10^{-2}	7.1×10^{-2}	0.38
	psd + nn	26	523	668	3.2×10^{-10}	3.4×10^{-13}	5.3×10^{-8}	5.3×10^{-8}	0.46
tai7a	original	50	1275	1429	4.6×10^{-7}	4.6×10^{-11}	7.5×10^{-2}	8.2×10^{-2}	1.42
	only psd	37	1275	1325	1.9×10^{-6}	7.0×10^{-11}	3.7×10^{-2}	4.1×10^{-2}	1.97
	psd + nn	37	981	1227	5.4×10^{-9}	5.2×10^{-13}	1.8×10^{-7}	1.8×10^{-7}	1.62
tai8a	original	65	2145	2345	3.2×10^{-6}	1.7×10^{-10}	1.4×10^{-1}	1.7×10^{-1}	3.80
	only psd	50	2145	2210	5.5×10^{-6}	2.3×10^{-11}	6.1×10^{-2}	7.0×10^{-2}	5.06
	psd + nn	50	1697	2082	1.4×10^{-8}	2.9×10^{-13}	1.1×10^{-7}	1.1×10^{-7}	6.09
tai9a	original	82	3403	3655	1.4×10^{-6}	5.9×10^{-11}	8.9×10^{-2}	1.1×10^{-1}	11.65
	only psd	65	3403	3485	9.2×10^{-6}	3.0×10^{-11}	6.7×10^{-2}	8.3×10^{-2}	13.82
	psd + nn	65	2755	3323	4.2×10^{-9}	6.2×10^{-13}	9.2×10^{-7}	9.2×10^{-7}	19.71
tai10a	original	101	5151	5461	1.6×10^{-6}	8.4×10^{-11}	6.3×10^{-2}	7.9×10^{-2}	27.39
	only psd	82	5151	5252	8.2×10^{-6}	2.0×10^{-11}	3.7×10^{-2}	4.8×10^{-2}	38.14
	psd + nn	82	4251	5052	2.7×10^{-8}	6.0×10^{-13}	4.3×10^{-7}	4.3×10^{-7}	47.44
tai11a	original	122	7503	7877	2.8×10^{-6}	8.2×10^{-11}	6.7×10^{-2}	8.9×10^{-2}	55.47
	only psd	101	7503	7625	1.7×10^{-5}	3.0×10^{-11}	3.8×10^{-2}	4.9×10^{-2}	78.53
	psd + nn	101	6293	7383	1.2×10^{-8}	2.6×10^{-13}	2.2×10^{-7}	2.2×10^{-7}	120.36

Tab. 2.4: Numerical results for large instances of "taiXXa" in QAPLIB [19].

instance	type	dim psd	dim nn	dim eq	`err1`	`err3`	`err5`	`err6`	time
tai12a	original	145	10585	11029	2.8×10^{-6}	6.4×10^{-11}	6.5×10^{-2}	8.2×10^{-2}	124.30
	only psd	122	10585	10730	2.5×10^{-5}	3.1×10^{-11}	1.1×10^{-1}	1.2×10^{-1}	178.99
	psd + nn	122	9001	10442	2.0×10^{-9}	1.2×10^{-12}	1.1×10^{-6}	1.1×10^{-6}	220.68
tai13a	original	170	14535	15055	4.8×10^{-6}	1.2×10^{-10}	1.2×10^{-1}	1.5×10^{-1}	223.44
	only psd	145	14535	14705	3.7×10^{-5}	5.9×10^{-11}	1.3×10^{-1}	1.5×10^{-1}	281.18
	psd + nn	145	12507	14367	6.7×10^{-9}	6.6×10^{-13}	3.3×10^{-8}	3.3×10^{-8}	628.34
tai14a	original	197	19503	20105	2.9×10^{-6}	1.1×10^{-10}	1.1×10^{-1}	1.3×10^{-1}	442.23
	only psd	170	19503	19700	5.9×10^{-5}	1.8×10^{-11}	1.9×10^{-1}	2.1×10^{-1}	506.92
	psd + nn	170	16955	19308	4.8×10^{-9}	4.5×10^{-13}	1.3×10^{-7}	1.3×10^{-7}	1169.63
tai15a	original	226	25651	26341	1.5×10^{-5}	1.1×10^{-10}	2.2×10^{-1}	2.8×10^{-1}	447.18
	only psd	197	25651	25877	4.5×10^{-5}	2.8×10^{-10}	1.7×10^{-1}	2.0×10^{-1}	1012.57
	psd + nn	197	22501	25427	4.1×10^{-9}	4.8×10^{-13}	4.0×10^{-8}	3.9×10^{-8}	2231.11
tai16a	original	257	33153	33937	8.4×10^{-6}	2.6×10^{-10}	2.6×10^{-1}	3.0×10^{-1}	879.28
	only psd	226	33153	33410	5.4×10^{-5}	8.0×10^{-11}	2.1×10^{-1}	2.4×10^{-1}	1043.34
	psd + nn	226	29313	32898	2.4×10^{-8}	2.5×10^{-13}	6.6×10^{-8}	6.8×10^{-8}	2529.61
tai17a	original	290	42195	43079	1.1×10^{-5}	1.1×10^{-10}	2.2×10^{-1}	2.7×10^{-1}	1904.90
	only psd	257	42195	42485	7.0×10^{-5}	3.9×10^{-11}	2.9×10^{-1}	3.2×10^{-1}	1889.13
	psd + nn	257	37571	41907	5.4×10^{-8}	4.1×10^{-13}	2.7×10^{-8}	2.9×10^{-8}	4906.94

2.3 Inexact primal-dual path-following method

Let us see in this section an inexact primal-dual path-following methods for the DNN optimization problem reduced by a solution $(\boldsymbol{y}, \boldsymbol{S}, \boldsymbol{O})$ to System (2.6) proposed by Tanaka *et al.* [87]. Tanaka *et al.* [87] considered the problem below

$$\left|\begin{array}{ll} \text{minimize} & \boldsymbol{C} \bullet \boldsymbol{X} \\ \text{subject to} & \mathcal{A}\boldsymbol{X} = \boldsymbol{b}, \\ & \boldsymbol{Y} - \boldsymbol{U}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{U} = \boldsymbol{O}, \\ & \boldsymbol{X} \in \mathcal{S}_{+}^{n-r}, \\ & \boldsymbol{Y} \in \mathcal{N}^{n} \end{array}\right. \tag{2.22}$$

and its dual:

$$\left|\begin{array}{ll} \text{maximize} & \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} \\ \text{subject to} & \mathcal{A}^{\mathrm{T}}\boldsymbol{y} + \boldsymbol{S} + \boldsymbol{U}\boldsymbol{T}\boldsymbol{U}^{\mathrm{T}} = \boldsymbol{C}, \\ & \boldsymbol{S} \in \mathcal{S}_{+}^{n-r}, \\ & \boldsymbol{T} \in \mathcal{N}^{n}, \end{array}\right. \tag{2.23}$$

where $\mathcal{A} : \boldsymbol{X} \longmapsto (\boldsymbol{A}_1 \bullet \boldsymbol{X}, \ldots, \boldsymbol{A}_m \bullet \boldsymbol{X})^{\mathrm{T}}$ is a linear mapping and $\mathcal{A}^{\mathrm{T}} : \boldsymbol{y} \longmapsto \sum_{i=1}^{m} y_i \boldsymbol{A}_i$ is the adjoint of $\mathcal{A}$. We define a central path for (2.22) and (2.23) defined by

$$\left\{(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{y}, \boldsymbol{S}, \boldsymbol{T}) : \begin{array}{rrr} \mathcal{A}\boldsymbol{X} = \boldsymbol{b},\ \boldsymbol{Y} - \boldsymbol{U}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{U} = \boldsymbol{O}, & \boldsymbol{X} \in \operatorname{int}\mathcal{S}_{+}^{n-r}, & \boldsymbol{Y} \in \operatorname{int}\mathcal{N}^{n}, \\ \mathcal{A}^{\mathrm{T}}\boldsymbol{y} + \boldsymbol{S} + \boldsymbol{U}\boldsymbol{T}\boldsymbol{U}^{\mathrm{T}} = \boldsymbol{C}, & \boldsymbol{S} \in \operatorname{int}\mathcal{S}_{+}^{n-r}, & \boldsymbol{T} \in \operatorname{int}\mathcal{N}^{n}, \\ \mu > 0, & \boldsymbol{X}\boldsymbol{S} = \mu w \boldsymbol{I}, & \boldsymbol{Y} \circ \boldsymbol{T} = \mu \boldsymbol{E} \end{array}\right\},$$

where $w > 0$ is a fixed weight parameter. The central path plays the following important role in an interior-point algorithm: It is a smooth curve parameterized by μ and it converges to one of the optimal solutions as $\mu \downarrow 0$. Thus, by tracing this curve numerically, we can get the optimal solution.

We describe their inexact primal-dual path-following method for Problems (2.22)–(2.23) in Algorithm 2.1. In commonly used primal-dual interior-point methods, we solve the scaled Newton system *exactly* with a direct solver to compute a search direction. In the cases of DNN optimization, however, this is difficult because of the computational cost. Thus, we solve it *approximately* with an iterative solver in the computation of a search direction.

Remark 2.9. Problems (2.22)–(2.23) correspond to the reduced problem (2.10) by using a solution $(\boldsymbol{y}_{\mathrm{psd}}, \boldsymbol{S}_0, \boldsymbol{O})$ to System (2.6). In other words, the problems that are considered in this section are obtained by the reduction of only the semidefinite constraint in Problem (2.1). A primal-dual pair obtained by the reduction of both conic constraints can be represented

$$\left|\begin{array}{ll} \text{minimize} & \boldsymbol{C} \bullet \boldsymbol{X} \\ \text{subject to} & \mathcal{A}\boldsymbol{X} = \boldsymbol{b}, \\ & \mathcal{U}_0\boldsymbol{X} = \boldsymbol{0}, \\ & \mathcal{U}_{+}\boldsymbol{X} = \boldsymbol{x}_{+}, \\ & \boldsymbol{X} \in \mathcal{S}_{+}^{n-r}, \\ & \boldsymbol{x}_{+} \geq \boldsymbol{0} \end{array}\right. \tag{2.24}$$

Algorithm 2.1 Inexact path-following method for problems (2.22)–(2.23)

Set parameters $\sigma \in (0,1)$ and $\gamma \in (0,1)$.
Choose initial point $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{y}, \boldsymbol{S}, \boldsymbol{T}) \in \operatorname{int} \mathcal{S}_+^{n-r} \times \operatorname{int} \mathcal{N}^n \times \mathbb{R}^m \times \operatorname{int} \mathcal{S}_+^{n-r} \times \operatorname{int} \mathcal{N}^n$
while $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{y}, \boldsymbol{S}, \boldsymbol{T})$ is not approximately optimal **do**
    Let $(\Delta\boldsymbol{X}, \Delta\boldsymbol{Y}, \Delta\boldsymbol{y}, \Delta\boldsymbol{S}, \Delta\boldsymbol{T})$ be an *approximate* solution to a scaled Newton system
    $\alpha_{\mathrm{p}} := \min\{\max\{\alpha : \boldsymbol{X} + \alpha\Delta\boldsymbol{X} \in \mathcal{S}_+^{n-r}\}, \max\{\alpha : \boldsymbol{Y} + \alpha\Delta\boldsymbol{Y} \in \mathcal{N}^n\}\}$
    $\alpha_{\mathrm{d}} := \min\{\max\{\alpha : \boldsymbol{S} + \alpha\Delta\boldsymbol{S} \in \mathcal{S}_+^{n-r}\}, \max\{\alpha : \boldsymbol{T} + \alpha\Delta\boldsymbol{T} \in \mathcal{N}^n\}\}$
    $\alpha_{\mathrm{p}} := \min\{1, \gamma\alpha_{\mathrm{p}}\}$
    $\alpha_{\mathrm{d}} := \min\{1, \gamma\alpha_{\mathrm{d}}\}$
    $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{y}, \boldsymbol{S}, \boldsymbol{T}) := (\boldsymbol{X} + \alpha_{\mathrm{p}}\Delta\boldsymbol{X}, \boldsymbol{Y} + \alpha_{\mathrm{p}}\Delta\boldsymbol{Y}, \boldsymbol{y} + \alpha_{\mathrm{d}}\Delta\boldsymbol{y}, \boldsymbol{S} + \alpha_{\mathrm{d}}\Delta\boldsymbol{S}, \boldsymbol{T} + \alpha_{\mathrm{d}}\Delta\boldsymbol{T})$
end while

and

$$\left|\begin{array}{ll} \text{maximize} & \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} \\ \text{subject to} & \mathcal{A}^{\mathrm{T}}\boldsymbol{y} + \boldsymbol{S} + \mathcal{U}_0^{\mathrm{T}}\boldsymbol{t}_0 + \mathcal{U}_+^{\mathrm{T}}\boldsymbol{t}_+ = \boldsymbol{C}, \\ & \boldsymbol{S} \in \mathcal{S}_+^{n-r}, \\ & \boldsymbol{t}_+ \geq \boldsymbol{0} \end{array}\right. \tag{2.25}$$

for linear mappings $\mathcal{U}_0 : \mathcal{S}^n \longrightarrow \mathbb{R}^{|I_+|}$ and $\mathcal{U}_+ : \mathcal{S}^n \longrightarrow \mathbb{R}^{|I_0|}$. A central path for Problems (2.24)–(2.25) is written as

$$\left\{ (\boldsymbol{X}, \boldsymbol{x}_+, \boldsymbol{y}, \boldsymbol{S}, \boldsymbol{t}_0, \boldsymbol{t}_+) : \begin{array}{c} \mathcal{A}\boldsymbol{X} = \boldsymbol{b},\ \mathcal{U}_0\boldsymbol{X} = \boldsymbol{0},\ \mathcal{U}_+\boldsymbol{X} = \boldsymbol{x}_+, \\ \mathcal{A}^{\mathrm{T}}\boldsymbol{y} + \boldsymbol{S} + \mathcal{U}_0^{\mathrm{T}}\boldsymbol{t}_0 + \mathcal{U}_+^{\mathrm{T}}\boldsymbol{t}_+ = \boldsymbol{C}, \\ \boldsymbol{X} \in \operatorname{int} \mathcal{S}_+^{n-r},\ \boldsymbol{S} \in \operatorname{int} \mathcal{S}_+^{n-r}, \\ \boldsymbol{x}_+ > \boldsymbol{0},\ \boldsymbol{t}_+ > \boldsymbol{0}, \\ \mu > 0,\ \boldsymbol{X}\boldsymbol{S} = \mu w \boldsymbol{I},\ \boldsymbol{x}_+ \circ \boldsymbol{t}_+ = \mu \boldsymbol{e} \end{array} \right\}.$$

2.3.1 Computation of search direction

In this subsection, we mention the computation of a search direction. To compute a search direction, we solve a linear system, which is a so-called augmented system. The size of the augmented system is $(n-r)^2 + m$. By contrast, the size of the Schur equation that we solve in the commonly used primal-dual interior-point algorithm for SDP equivalent to (2.22) and its dual is $n(n+1)/2 + m$. Thus, we solve smaller linear systems at each iteration in the inexact primal-dual path-following method. However, direct solvers still cannot solve them with a reasonable computational cost, so by applying an iterative solver, we attempt to reduce the computational cost.

First, we introduce the augmented system, which we must solve to compute a search direction. Let $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{y}, \boldsymbol{S}, \boldsymbol{T})$ and $(\Delta\boldsymbol{X}, \Delta\boldsymbol{Y}, \Delta\boldsymbol{y}, \Delta\boldsymbol{S}, \Delta\boldsymbol{T})$ be a current point and a search direction, respectively. They satisfy the following scaled Newton

system:

$$\mathcal{A}\Delta\boldsymbol{X} = \boldsymbol{r}_{\mathrm{p}}, \tag{2.26}$$
$$\Delta\boldsymbol{Y} - \boldsymbol{U}^{\mathrm{T}}\Delta\boldsymbol{X}\boldsymbol{U} = \boldsymbol{R}_{\mathrm{p}}, \tag{2.27}$$
$$\mathcal{A}^{\mathrm{T}}\Delta\boldsymbol{y} + \Delta\boldsymbol{S} + \boldsymbol{U}\Delta\boldsymbol{T}\boldsymbol{U}^{\mathrm{T}} = \boldsymbol{R}_{\mathrm{d}}, \tag{2.28}$$
$$\mathcal{E}\Delta\boldsymbol{X} + \mathcal{F}\Delta\boldsymbol{S} = \boldsymbol{R}_{\mathrm{psd}}, \tag{2.29}$$
$$\boldsymbol{T} \circ \Delta\boldsymbol{Y} + \boldsymbol{Y} \circ \Delta\boldsymbol{T} = \boldsymbol{R}_{\mathrm{nn}}, \tag{2.30}$$

where

$$\begin{aligned}
\boldsymbol{r}_{\mathrm{p}} &= \boldsymbol{b} - \mathcal{A}\boldsymbol{X}, \\
\boldsymbol{R}_{\mathrm{p}} &= \boldsymbol{U}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{U} - \boldsymbol{Y}, \\
\boldsymbol{R}_{\mathrm{d}} &= \boldsymbol{C} - \mathcal{A}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{S} - \boldsymbol{U}\boldsymbol{T}\boldsymbol{U}^{\mathrm{T}}, \\
\boldsymbol{R}_{\mathrm{psd}} &= \sigma\mu w\boldsymbol{I} - \mathcal{H}_{\boldsymbol{P}}(\boldsymbol{X}\boldsymbol{S}), \\
\boldsymbol{R}_{\mathrm{nn}} &= \sigma\mu\boldsymbol{E} - \boldsymbol{Y} \circ \boldsymbol{T},
\end{aligned}$$

and $\mathcal{E}, \mathcal{F}$ are linear operators that depend on the scaling operator $\mathcal{H}_{\boldsymbol{P}}$ (for more details, see [67, 89]). Eliminating $\Delta\boldsymbol{Y}$, $\Delta\boldsymbol{S}$, and $\Delta\boldsymbol{T}$, we obtain the following system:

$$\begin{pmatrix} -\mathcal{H} & \mathcal{A}^{\mathrm{T}} \\ \mathcal{A} & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \Delta\boldsymbol{X} \\ \Delta\boldsymbol{y} \end{pmatrix} = \begin{pmatrix} \boldsymbol{R}_{\mathrm{a}} \\ \boldsymbol{r}_{\mathrm{p}} \end{pmatrix}, \tag{2.31}$$

where

$$\begin{aligned}
\mathcal{H} &= \mathcal{F}^{-1}\mathcal{E} + (\boldsymbol{U} \otimes \boldsymbol{U})\,\mathrm{Diag}(\boldsymbol{Y}^{(-1)} \circ \boldsymbol{T})(\boldsymbol{U} \otimes \boldsymbol{U})^{\mathrm{T}}, \\
\boldsymbol{R}_{\mathrm{a}} &= \boldsymbol{R}_{\mathrm{d}} - \mathcal{F}^{-1}\boldsymbol{R}_{\mathrm{psd}} - \boldsymbol{U}(\boldsymbol{Y}^{(-1)} \circ \boldsymbol{R}_{\mathrm{nn}})\boldsymbol{U}^{\mathrm{T}} + \boldsymbol{U}(\boldsymbol{Y}^{(-1)} \circ \boldsymbol{T} \circ \boldsymbol{R}_{\mathrm{p}})\boldsymbol{U}^{\mathrm{T}}.
\end{aligned}$$

This linear system is called the *augmented system* and the coefficient matrix is called the *augmented matrix.* For the NT direction, we have $\mathcal{F}^{-1}\mathcal{E} = \boldsymbol{W}^{-1} \otimes \boldsymbol{W}^{-1}$, where $\boldsymbol{W}$ is the unique matrix that satisfies $\boldsymbol{W}\boldsymbol{S}\boldsymbol{W} = \boldsymbol{X}$. For the HKM direction, we have $\mathcal{E}^{-1}\mathcal{F} = \boldsymbol{X} \odot \boldsymbol{S}^{-1}$ and, for the dual HKM direction, we have $\mathcal{F}^{-1}\mathcal{E} = \boldsymbol{S} \odot \boldsymbol{X}^{-1}$, where $\boldsymbol{P} \odot \boldsymbol{Q} : \mathbb{R}^{n\times n} \longrightarrow \mathbb{R}^{n\times n}$ denotes the symmetrized Kronecker product defined by $\boldsymbol{X} \longmapsto (\boldsymbol{Q}\boldsymbol{X}\boldsymbol{P}^{\mathrm{T}} + \boldsymbol{P}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{Q}^{\mathrm{T}})/2$. In this paper, we consider only the case of the NT direction since it is easy to construct preconditioners for the augmented system. Eliminating $\Delta\boldsymbol{X}$ from (2.31), we get the following Schur equation with size m:

$$\mathcal{A}\mathcal{H}^{-1}\mathcal{A}^{\mathrm{T}}\Delta\boldsymbol{y} = \boldsymbol{r}_{\mathrm{p}} + \mathcal{A}\mathcal{H}^{-1}\boldsymbol{R}_{\mathrm{a}}. \tag{2.32}$$

When we solve SDPs or convex quadratic SDPs with a special structure [92], we usually solve a Schur equation like (2.32). In this case, however, the computation of $\mathcal{H}^{-1}$ involves a huge cost, so that, we cannot solve the Schur equation (2.32) unless we pay a huge cost. Therefore, we solve the augmented system (2.31) as in the case of general QSDPs [90]. When we solve the SDPs equivalent to (2.22) with commonly used primal-dual interior-point methods,

we have to solve a Schur equation with size $n(n+1)/2+m$ to compute a search direction at each iteration. In their method, it is only necessary to solve the augmented system (2.31) with size $(n-r)^2+m$.

Unfortunately, to solve the augmented system (2.31), direct solvers still require $O([(n-r)^2+m]^3)$ of computational time and $O([(n-r)^2+m]^2)$ of memory since $\mathcal{H}$ is dense. Thus, when we solve medium-sized or large instances, direct solvers are not reasonable choices. Therefore, we solve the augmented system (2.31) by means of an iterative solver. Toh [90] has shown that the preconditioned symmetric quasi-minimal residual (PSQMR) method, which is a Krylov subspace method proposed by Freund and Nachtigal [36] solves the augmented system $(-(\mathcal{F}^{-1}\mathcal{E}+\mathcal{Q}), \mathcal{A}^{\mathrm{T}}; \mathcal{A}, \boldsymbol{O})$ efficiently in an inexact primal-dual path-following method for the following QSDP:

$$\left|\begin{array}{ll} \text{minimize} & \dfrac{1}{2}\boldsymbol{X}\bullet\mathcal{Q}\boldsymbol{X}+\boldsymbol{C}\bullet\boldsymbol{X} \\ \text{subject to} & \mathcal{A}\boldsymbol{X}=\boldsymbol{b}, \\ & \boldsymbol{X}\in\mathcal{S}^n_+, \end{array}\right.$$

where $\mathcal{Q}$ is a given positive semidefinite operator. Note that the coefficient matrix of the augmented system (2.31) is the matrix replacing $\mathcal{Q}$ in that of QSDPs with $(\boldsymbol{U}\otimes\boldsymbol{U})\,\mathrm{Diag}(\boldsymbol{Y}^{(-1)}\circ\boldsymbol{T})(\boldsymbol{U}\otimes\boldsymbol{U})^{\mathrm{T}}$. Thus, the PSQMR method can be considered to be an effective approach. At each iteration of the algorithm, we compute the product of the augmented matrix and a vector. In this case, we do not need to store $\mathcal{H}$ in memory since we can compute the product by the following formula:

$$\begin{pmatrix} -\mathcal{H} & \mathcal{A}^{\mathrm{T}} \\ \mathcal{A} & \boldsymbol{O} \end{pmatrix}\begin{pmatrix} \mathcal{Q} \\ \boldsymbol{q} \end{pmatrix}=\begin{pmatrix} -\boldsymbol{W}^{-1}\mathcal{Q}\boldsymbol{W}^{-1}-\boldsymbol{U}[\boldsymbol{Y}^{(-1)}\circ\boldsymbol{T}\circ(\boldsymbol{U}^{\mathrm{T}}\mathcal{Q}\boldsymbol{U})]\boldsymbol{U}^{\mathrm{T}}+\mathcal{A}^{\mathrm{T}}\boldsymbol{q} \\ \mathcal{A}\mathcal{Q} \end{pmatrix}.$$

The computation of the right hand side requires $O(n^2(n-r)+m(n-r)^2)$ of computational time and $O(n^2+m)$ of additional memory. Thus, if the iterative method terminates quickly, we can obtain a solution of the augmented system (2.31) faster with lower memory cost.

The disadvantage of using the iterative solver is that it is impossible to solve the augmented system accurately. More precisely, the residuals of the scaled Newton system (2.26)–(2.30) would be nonzero. In particular, if the residuals of (2.26), (2.27) and (2.28) are nonzero, then $\|\boldsymbol{r}_{\mathrm{p}}\|$, $\|\boldsymbol{R}_{\mathrm{p}}\|_{\mathrm{F}}$, and $\|\boldsymbol{R}_{\mathrm{d}}\|_{\mathrm{F}}$ could increase. If this occurs, primal-dual path-following methods might not converge. To avoid this undesirable situation, in their algorithm, we adjust the search directions to satisfy (2.26), (2.27), and (2.28) accurately. We can compute a search direction that satisfies (2.26), (2.27), and (2.28) accurately by Algorithm 2.2 proposed in [87]. Note that it requires some computation to obtain a search direction that satisfies (2.26) accurately as we see below. We can obtain a search direction that satisfies (2.26), by solving the following least squares problem for example:

$$\left|\begin{array}{ll} \text{minimize} & \|\Delta\boldsymbol{X}_{\mathrm{adj}}-\Delta\boldsymbol{X}\|_{\mathrm{F}}^2 \\ \text{subject to} & \mathcal{A}\Delta\boldsymbol{X}_{\mathrm{adj}}=\boldsymbol{r}_{\mathrm{p}}. \end{array}\right.$$

This adjustment is based on Kojima *et al.* [57]. The optimal solution can be computed easily by the following formula:

$$\Delta \boldsymbol{X}^{*}_{\mathrm{adj}} = \Delta \boldsymbol{X} + \mathcal{A}^{\mathrm{T}}(\mathcal{A}\mathcal{A}^{\mathrm{T}})^{-1}(\boldsymbol{r}_{\mathrm{p}} - \mathcal{A}\Delta \boldsymbol{X}). \tag{2.33}$$

Since $\mathcal{A}\mathcal{A}^{\mathrm{T}} \in \mathcal{S}^{m}_{+}$ stays constant, we need not compute its Cholesky factorization at each iteration. Moreover, when $\mathcal{A}$ is sparse, we can expect the Cholesky factorization of $\mathcal{A}\mathcal{A}^{\mathrm{T}}$ to become sparse too. Therefore, the computation of (2.33) is much easier than the computation of $(\Delta \boldsymbol{X}, \Delta \boldsymbol{y})$. Although the residual of (2.29) is not equal to zero, we do not need to solve (2.29) and (2.30) as accurately as (2.26), (2.27), and (2.28) since they are linear approximations of nonlinear equalities.

Algorithm 2.2 Computation of a search direction

1: Solve System (2.31) via the iterative solver and obtain $(\Delta \boldsymbol{X}, \Delta \boldsymbol{y})$
2: By substituting $\Delta \boldsymbol{X}$ into (2.33), compute $\Delta \boldsymbol{X}_{\mathrm{adj}}$
3: By substituting $\Delta \boldsymbol{X}_{\mathrm{adj}}$ into $\Delta \boldsymbol{X}$ in (2.27), compute $\Delta \boldsymbol{Y}$
4: By substituting $\Delta \boldsymbol{Y}$ into (2.30), compute $\Delta \boldsymbol{T}$
5: By substituting $\Delta \boldsymbol{y}$ and $\Delta \boldsymbol{Y}$ into (2.28), compute $\Delta \boldsymbol{S}$

Remark 2.10. We can derive a computation method for an inexact primal-dual path-following method for Problems (2.24)–(2.25). The corresponding augmented matrix is

$$\begin{pmatrix} -\mathcal{H} & \mathcal{A}^{\mathrm{T}} & \mathcal{U}_0 \\ \mathcal{A} & \boldsymbol{O} & \boldsymbol{O} \\ \mathcal{U}_0 & \boldsymbol{O} & \boldsymbol{O} \end{pmatrix},$$

where

$$\mathcal{H} = \mathcal{F}^{-1}\mathcal{E} + \mathcal{U}_{+}^{\mathrm{T}} \operatorname{Diag}(\boldsymbol{x}_{+}^{(-1)} \circ \boldsymbol{t}_{+})\mathcal{U}_{+}.$$

However, no effective preconditioners for this matrix have been obtained at the time of this thesis.

2.3.2 Preconditioning for augmented matrix

Let us see in this subsection three preconditioners for the augmented matrix proposed by Tanaka *et al.* [87] to accelerate convergence of the PSQMR method. The augmented matrix becomes ill-conditioned as the iteration of the path-following method becomes close to an optimal solution. Generally, the convergence rate of Krylov subspace methods depends on the condition number of the coefficient matrix. Thus, to overcome the difficulty, we must use appropriate preconditioners for the augmented matrix.

The unpreconditioned PSQMR method is equivalent to the minimal residual (MINRES) method [72]. Although the MINRES method allows only symmetric positive definite preconditioners, the PSQMR method allows symmetric indefinite preconditioners that can be effective preconditioners. Actually, Toh [90]

has proposed three effective symmetric indefinite preconditioners for the augmented matrix of QSDPs. Recall that the augmented system (2.31) has a similar structure to that of QSDPs. Therefore, we extend the preconditioning strategies for the augmented matrix of QSDPs proposed by Toh [90]. In these preconditioning strategies, we approximate $\mathcal{H}$ in the $(1,1)$-block of the augmented matrix by a simpler operator for which the inverse is easy to compute. It is worth noting that, although the condition number of $\boldsymbol{Q}$ in QSDPs stays constant, the condition number of $(\boldsymbol{U} \otimes \boldsymbol{U})\,\mathrm{Diag}(\boldsymbol{Y}^{(-1)} \circ \boldsymbol{T})(\boldsymbol{U} \otimes \boldsymbol{U})^{\mathrm{T}}$ becomes large. Thus, we must approximate $\mathcal{H}$ more appropriately.

Let $\mathcal{K}$ be an approximate operator of $\mathcal{H}$ for which $\mathcal{K}^{-1}$ is easy to compute. The preconditioners proposed below have the form of $(-\mathcal{K}, \mathcal{A}^{\mathrm{T}}; \mathcal{A}, \boldsymbol{O})$. Thus, for given $(\boldsymbol{R}; \boldsymbol{r}) \in \mathcal{S}^{n-r} \times \mathbb{R}^m$, we can compute $(\boldsymbol{Q}; \boldsymbol{q}) = (-\mathcal{K}, \mathcal{A}^{\mathrm{T}}; \mathcal{A}, \boldsymbol{O})^{-1}(\boldsymbol{R}; \boldsymbol{r})$ efficiently by Algorithm 2.3.

Algorithm 2.3 Preconditioning for System (2.31)

1: $\boldsymbol{B} := \mathcal{A}\mathcal{K}^{-1}\mathcal{A}^{\mathrm{T}}$
2: $\boldsymbol{s} := \mathcal{A}\mathcal{K}^{-1}\boldsymbol{R} + \boldsymbol{r}$
3: $\boldsymbol{q} := \boldsymbol{B}^{-1}\boldsymbol{s}$
4: $\boldsymbol{Q} := \mathcal{K}^{-1}(\mathcal{A}^{\mathrm{T}}\boldsymbol{q} - \boldsymbol{R})$

Hereinafter, we will see the three preconditioners $\mathcal{V}^*$, $\boldsymbol{\Psi}^*$, and $\boldsymbol{\Sigma}$ depending on the approximation of $\mathcal{H}$. To construct the two preconditioners $\mathcal{V}^*$ and $\boldsymbol{\Psi}^*$, we use the nearest Kronecker product approximation. To construct the last preconditioner $\boldsymbol{\Sigma}$, we use a diagonal scaling.

Nearest Kronecker product approach

First, we introduce the nearest Kronecker product approach. This preconditioner is an extension of $\mathcal{V}$ proposed by Toh [90]. Here, we represent $\mathcal{H}$ as the sum of $1+n$ Kronecker products and approximate it as one Kronecker product. Thus, the preconditioner $\mathcal{V}^*$ has the form:

$$\mathcal{V}^* = \begin{pmatrix} -\boldsymbol{V} \otimes \boldsymbol{V} & \mathcal{A}^{\mathrm{T}} \\ \mathcal{A} & \boldsymbol{O} \end{pmatrix},$$

where $\boldsymbol{V} \otimes \boldsymbol{V}$ is an approximation of $\mathcal{H}$.

In this approach, we use the nearest Kronecker product approximation:

$$\min_{\boldsymbol{V}_{\mathrm{L}} \in \mathbb{R}^{n \times n}, \boldsymbol{V}_{\mathrm{R}} \in \mathbb{R}^{m \times m}} \left\| \sum_{p=1}^{q} \boldsymbol{G}_p \otimes \boldsymbol{K}_p - \boldsymbol{V}_{\mathrm{L}} \otimes \boldsymbol{V}_{\mathrm{R}} \right\|_{\mathrm{F}}^2, \tag{2.34}$$

where $\boldsymbol{G}_1, \ldots, \boldsymbol{G}_q \in \mathbb{R}^{n \times n}$ and $\boldsymbol{K}_1, \ldots, \boldsymbol{K}_q \in \mathbb{R}^{m \times m}$ are given matrices. It has been shown by Langville and Stewart [59] that the optimal solution of (2.34) takes the form $\boldsymbol{V}_{\mathrm{L}} = \sum_{p=1}^{q} \alpha_p \boldsymbol{G}_p$ and $\boldsymbol{V}_{\mathrm{R}} = \sum_{p=1}^{q} \beta_p \boldsymbol{K}_p$, where $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is the optimal solution

of the following nonlinear optimization problem:

$$\min_{\alpha,\beta\in\mathbb{R}^q} \operatorname{tr}(\boldsymbol{G}\boldsymbol{K}) - 2\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{K}\boldsymbol{\beta} + (\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{\alpha})(\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\beta}), \tag{2.35}$$

where $\boldsymbol{G}, \boldsymbol{K} \in \mathcal{S}_+^q$ are the matrices defined by $G_{ij} = \boldsymbol{G}_i \bullet \boldsymbol{G}_j$ and $K_{ij} = \boldsymbol{K}_i \bullet \boldsymbol{K}_j$. Moreover, it has also been shown by Toh *et al.* [92] that the optimal solution of the nonconvex optimization problem (2.35) $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is the pair of the left and right eigenvectors of $\boldsymbol{G}\boldsymbol{K}$ corresponding to the maximum eigenvalue of $\boldsymbol{G}\boldsymbol{K}$.

To apply this approximation, we represent $\mathcal{H}$ as a sum of Kronecker product. To get this representation, we apply spectral decomposition to $\boldsymbol{Y}^{(-1)} \circ \boldsymbol{T}$ as follows:

$$\boldsymbol{Y}^{(-1)} \circ \boldsymbol{T} = \sum_{i=1}^{n} \lambda_i \boldsymbol{q}_i \boldsymbol{q}_i^{\mathrm{T}}.$$

Using the eigenvalues and eigenvectors, we obtain the following approximation:

$$\begin{aligned}
\mathcal{H} &= \boldsymbol{W}^{-1} \otimes \boldsymbol{W}^{-1} + (\boldsymbol{U} \otimes \boldsymbol{U}) \left(\sum_{i=1}^{n} \sqrt{\sigma_i \lambda_i} \operatorname{Diag}(\boldsymbol{q}_i) \otimes \sigma_i \sqrt{\sigma_i \lambda_i} \operatorname{Diag}(\boldsymbol{q}_i) \right) (\boldsymbol{U} \otimes \boldsymbol{U})^{\mathrm{T}} \\
&= \boldsymbol{W}^{-1} \otimes \boldsymbol{W}^{-1} + \sum_{i=1}^{n} \sqrt{\sigma_i \lambda_i} \boldsymbol{U} \operatorname{Diag}(\boldsymbol{q}_i) \boldsymbol{U}^{\mathrm{T}} \otimes \sigma_i \sqrt{\sigma_i \lambda_i} \boldsymbol{U} \operatorname{Diag}(\boldsymbol{q}_i) \boldsymbol{U}^{\mathrm{T}} \\
&\simeq \boldsymbol{V} \otimes \boldsymbol{V},
\end{aligned}$$

where

$$\sigma_i = \begin{cases} +1 & \text{if } \lambda_i > 0, \\ -1 & \text{otherwise if } \lambda_i < 0, \\ 0 & \text{otherwise.} \end{cases}$$

Note that we have the nearest Kronecker product such that $\boldsymbol{V} = \boldsymbol{V}_{\mathrm{L}}^* = \boldsymbol{V}_{\mathrm{R}}^* \in \operatorname{int} \mathcal{S}_+^{n-r}$. We show two properties about the nearerst Kronecker product approximation that hold in the special cases that we will consider below. One property holds when there is only a difference in sign between $\boldsymbol{G}_p$ and $\boldsymbol{K}_p$.

Theorem 2.11. *For $\boldsymbol{K}_p, \boldsymbol{G}_p \in \mathbb{R}^{n\times n}$ with $p = 1, \ldots, q$, we suppose $\boldsymbol{K}_p = \sigma_p \boldsymbol{G}_p$ for some $\boldsymbol{\sigma} \in \{+1, -1\}^q$. Then, if $(\boldsymbol{V}_{\mathrm{L}}^*, \boldsymbol{V}_{\mathrm{R}}^*)$ denotes an optimal solution of* (2.34)*, it holds that $\boldsymbol{V}_{\mathrm{R}}^* = c\boldsymbol{V}_{\mathrm{L}}^*$ for some constant c.*

Proof. For all entries of $\boldsymbol{K}$, we have $K_{ij} = \sigma_i \sigma_j \boldsymbol{G}_i \bullet \boldsymbol{G}_j$. Now, let $\boldsymbol{\Sigma} = \operatorname{Diag}(\boldsymbol{\sigma})$. Since $\boldsymbol{K} = \boldsymbol{\Sigma}\boldsymbol{G}\boldsymbol{\Sigma}$, it holds that $\boldsymbol{G}\boldsymbol{K} = (\boldsymbol{G}\boldsymbol{\Sigma})^2$. Therefore, we can see that $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ is the pair of the left and right eigenvectors of $\boldsymbol{G}\boldsymbol{\Sigma}$ corresponding to the eigenvalue whose absolute value is the maximum. Below, λ denotes this eigenvalue. Since $\boldsymbol{\beta}^*$ is a right eigenvector corresponding to λ, *i.e.*,

$$\boldsymbol{G}\boldsymbol{\Sigma}\boldsymbol{\beta}^* = \lambda\boldsymbol{\beta}^*,$$

it holds that

$$(\boldsymbol{\Sigma}\boldsymbol{\beta}^*)^{\mathrm{T}}\boldsymbol{G}\boldsymbol{\Sigma} = \lambda(\boldsymbol{\Sigma}\boldsymbol{\beta}^*)^{\mathrm{T}}.$$

This implies that $\Sigma\beta^*$ is the left eigenvector corresponding to λ. Therefore,

$$\alpha^* = c\Sigma\beta^* = c\sigma \circ \beta^*,$$

where $c = 1/(\beta^*)^{\mathrm{T}}\Sigma\beta^*$. Finally, we obtain

$$V_{\mathrm{L}}^* = \sum_{p=1}^{q} \alpha_p^* G_p = c \sum_{p=1}^{q} \beta_p^* K_p = cV_{\mathrm{R}}^*.$$

□

The other property implies that when the sum of Kronecker products is positive definite, the nearest Kronecker product is also positive definite.

Theorem 2.12. *For $K_p \in \mathcal{S}^m$ and $G_p \in \mathcal{S}^n$ with $p = 1, \ldots, q$, we suppose that $\sum_{p=1}^{q} G_p \otimes K_p$ is positive definite. Then, the nearest Kronecker product $V_{\mathrm{L}}^* \otimes V_{\mathrm{R}}^*$ is positive definite.*

Proof. Let $\mathcal{F} = \sum_{p=1}^{q} G_p \otimes K_p$ and $V_{\mathrm{L}} \otimes V_{\mathrm{R}}$ be the nearest Kronecker product. Applying the spectral decomposition, we obtain $V_{\mathrm{L}} = P\Lambda P^{\mathrm{T}}$ and $V_{\mathrm{R}} = Q\Sigma Q^{\mathrm{T}}$, where P and Q are orthogonal matrices and Λ and Σ are the diagonal matrices whose diagonal elements are the eigenvalues of V_{L} and V_{R}. Let Λ_+ and Λ_- be the diagonal matrices whose diagonal elements are nonpositive and nonnegative elements of Λ replaced by 0 and $V_{\mathrm{L}+} = P\Lambda_+ P^{\mathrm{T}}$ and $V_{\mathrm{L}-} = -P\Lambda_- P^{\mathrm{T}}$, respectively. Thus, we can decompose V as the difference between two semidefinite matrices $V_{\mathrm{L}} = V_{\mathrm{L}+} - V_{\mathrm{L}-}$. In a similar way, we can decompose $V_{\mathrm{R}} = V_{\mathrm{R}+} - V_{\mathrm{R}-}$. With this decomposition, the residual norm of the approximation is represented as follows:

$$\begin{aligned} &\|\mathcal{F} - V_{\mathrm{L}} \otimes V_{\mathrm{R}}\|_{\mathrm{F}}^2 \\ &= \|\mathcal{F}\|_{\mathrm{F}}^2 - 2\mathcal{F} \bullet [(V_{\mathrm{L}+} - V_{\mathrm{L}-}) \otimes (V_{\mathrm{R}+} - V_{\mathrm{R}-})] + \operatorname{tr}(V_{\mathrm{L}+}^2 + V_{\mathrm{L}-}^2) \operatorname{tr}(V_{\mathrm{R}+}^2 + V_{\mathrm{R}-}^2). \end{aligned}$$

In what follows, we show that unless both V_{L} and V_{R} are positive definite (or negative definite), there exists another matrix that has smaller residual norms. Let Λ_0 and be the diagonal matrix whose diagonal elements are zero and one corresponding to nonzero and zero diagonal elements of Λ, respectively, and $V_{\mathrm{L}0} = P\Lambda_0 P^{\mathrm{T}}$. Moreover, we define $V_{\mathrm{R}0}$ similarly. Now we consider $V_{\mathrm{L}}(\alpha) = V_{\mathrm{L}+} + V_{\mathrm{L}-} + \alpha V_{\mathrm{L}0}$ and $V_{\mathrm{R}}(\alpha) = V_{\mathrm{R}+} + V_{\mathrm{R}-} + \alpha V_{\mathrm{R}0}$. When we approximate $\mathcal{F}$ with $V_{\mathrm{L}}(\alpha) \otimes V_{\mathrm{R}}(\alpha)$, the residual norm is as follows:

$$\begin{aligned} &\|\mathcal{F} - V_{\mathrm{L}}(\alpha) \otimes V_{\mathrm{R}}(\alpha)\|_{\mathrm{F}}^2 \\ &\quad = \|\mathcal{F}\|_{\mathrm{F}}^2 - 2\mathcal{F} \bullet [(V_{\mathrm{L}+} + V_{\mathrm{L}-} + \alpha V_{\mathrm{L}0}) \otimes (V_{\mathrm{R}+} - V_{\mathrm{R}-} + \alpha V_{\mathrm{R}0})] \\ &\qquad\qquad + \operatorname{tr}(V_{\mathrm{L}+}^2 + V_{\mathrm{L}-}^2 + \alpha^2 V_{\mathrm{L}0}^2) \operatorname{tr}(V_{\mathrm{R}+}^2 + V_{\mathrm{R}-}^2 + \alpha^2 V_{\mathrm{R}0}^2). \end{aligned}$$

Let $f(\alpha) = \|\mathcal{F} - V_{\mathrm{L}} \otimes V_{\mathrm{R}}\|_{\mathrm{F}}^2 - \|\mathcal{F} - V_{\mathrm{L}}(\alpha) \otimes V_{\mathrm{R}}(\alpha)\|_{\mathrm{F}}^2$. Then,

$$f(0) = 4\mathcal{F} \bullet (V_{\mathrm{L}+} \otimes V_{\mathrm{R}-} + V_{\mathrm{L}-} \otimes V_{\mathrm{R}+}) \geq 0.$$

Note that the equality holds if and only if both $\boldsymbol{V}_{\mathrm{L}}$ and $\boldsymbol{V}_{\mathrm{R}}$ are positive semidefinite (or negative semidefinite). In such cases, we consider the derivative below:

$$f'(0) = 2\mathcal{F} \bullet [(\boldsymbol{V}_{\mathrm{L}+} + \boldsymbol{V}_{\mathrm{L}-}) \otimes \boldsymbol{V}_{\mathrm{R}0} + \boldsymbol{V}_{\mathrm{L}0} \otimes (\boldsymbol{V}_{\mathrm{R}+} + \boldsymbol{V}_{\mathrm{R}-})] \geq 0.$$

The equality holds if and only if both $\boldsymbol{V}_{\mathrm{L}}$ and $\boldsymbol{V}_{\mathrm{R}}$ are zero matrices or positive definite (or negative definite). In the case that $\boldsymbol{V}_{\mathrm{L}}$, $\boldsymbol{V}_{\mathrm{R}}$ are zero matrices. However, when we set $\alpha^* = \sqrt{\mathrm{tr}(\mathcal{F})/mn}$, then we obtain

$$f(\alpha^*) = \frac{\mathrm{tr}(\mathcal{F})}{mn} > 0.$$

As we mentioned above, unless both $\boldsymbol{V}_{\mathrm{L}}$ and $\boldsymbol{V}_{\mathrm{R}}$ are positive definite (or negative definite), there exists a nearer Kronecker product. Therefore, we conclude that the nearest Kronecker product is positive definite. □

From the positive definiteness of $\boldsymbol{W}^{-1}$ and the positivity of $\boldsymbol{Y}^{(-1)} \circ \boldsymbol{T}$, $\mathcal{H}$ is positive definite. From Theorem 2.12, the nearest Kronecker product of $\mathcal{H}$ is also positive definite. Moreover, since the sum of the Kronecker product representation of $\mathcal{H}$ satisfies the assumption of Theorem 2.11, $\boldsymbol{V}_{\mathrm{R}}^* = c\boldsymbol{V}_{\mathrm{L}}^*$ for some $c > 0$. Therefore, there is a nearest Kronecker product such that $\boldsymbol{V}_{\mathrm{L}}^* = \boldsymbol{V}_{\mathrm{R}}^*$.

Block diagonal and the nearest Kronecker product approach

The next preconditioner is based on the preconditioner $\boldsymbol{\Psi}$ proposed by Toh [90]. In this preconditioning strategy, we partition the (1, 1)-block of the augmented matrix according to the eigenvalues of $\boldsymbol{W}^{-1}$. As mentioned in [90], $\boldsymbol{W}^{-1}$ has l small eigenvalues of order $\Theta(\sqrt{\mu})$ and $n - r - l$ large eigenvalues of order $\Theta(1/\sqrt{\mu})$ where l denote the rank of $\boldsymbol{X}$ at the maximal complementary solution. Thus, we decompose $\boldsymbol{W}^{-1}$ as follows:

$$\boldsymbol{W}^{-1} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{\mathrm{T}} = \boldsymbol{P}_1\boldsymbol{D}_1\boldsymbol{P}_1^{\mathrm{T}} + \boldsymbol{P}_2\boldsymbol{D}_2\boldsymbol{P}_2^{\mathrm{T}},$$

where $\boldsymbol{D}_1 = \mathrm{Diag}(\boldsymbol{d}_1) \in \mathrm{int}\,\mathcal{S}_+^l$ and $\boldsymbol{D}_2 = \mathrm{Diag}(\boldsymbol{d}_2) \in \mathrm{int}\,\mathcal{S}_+^{n-r-l}$ are the diagonal matrices whose diagonal elements are small and large eigenvalues of $\boldsymbol{W}^{-1}$, respectively, and $\boldsymbol{P}_1 \in \mathbb{R}^{(n-r)\times l}$ and $\boldsymbol{P}_2 \in \mathbb{R}^{(n-r)\times(n-r-l)}$ are the matrices whose columns are the eigenvectors of $\boldsymbol{W}^{-1}$ corresponding to $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$, respectively. Note that we have $\boldsymbol{D} = \mathrm{Diag}((\boldsymbol{d}_1; \boldsymbol{d}_2))$, $\boldsymbol{P} = (\boldsymbol{P}_1, \boldsymbol{P}_2)$. With this notation, we can represent $\mathcal{H}$ as follows:

$$\mathcal{H} = (\boldsymbol{P} \otimes \boldsymbol{P}) \left[\boldsymbol{D} \otimes \boldsymbol{D} + \sum_{i=1}^{n} \sqrt{\sigma_i\lambda_i}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U}\,\mathrm{Diag}(\boldsymbol{q}_i)\boldsymbol{U}^{\mathrm{T}}\boldsymbol{P} \otimes \sigma_i \sqrt{\sigma_i\lambda_i}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U}\,\mathrm{Diag}(\boldsymbol{q}_i)\boldsymbol{U}^{\mathrm{T}}\boldsymbol{P} \right] (\boldsymbol{P} \otimes \boldsymbol{P})^{\mathrm{T}}.$$

Here, we apply a block the diagonal approximation as follows:

$$\begin{aligned}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U}\,\mathrm{Diag}(\boldsymbol{q}_i)\boldsymbol{U}^{\mathrm{T}}\boldsymbol{P} &= \begin{pmatrix}\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{U}\,\mathrm{Diag}(\boldsymbol{q}_i)\boldsymbol{U}^{\mathrm{T}}\boldsymbol{P}_1 & \boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{U}\,\mathrm{Diag}(\boldsymbol{q}_i)\boldsymbol{U}^{\mathrm{T}}\boldsymbol{P}_2\\ \boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{U}\,\mathrm{Diag}(\boldsymbol{q}_i)\boldsymbol{U}^{\mathrm{T}}\boldsymbol{P}_1 & \boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{U}\,\mathrm{Diag}(\boldsymbol{q}_i)\boldsymbol{U}^{\mathrm{T}}\boldsymbol{P}_2\end{pmatrix}\\ &\simeq \begin{pmatrix}\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{U}\,\mathrm{Diag}(\boldsymbol{q}_i)\boldsymbol{U}^{\mathrm{T}}\boldsymbol{P}_1 & \boldsymbol{O}\\ \boldsymbol{O} & \boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{U}\,\mathrm{Diag}(\boldsymbol{q}_i)\boldsymbol{U}^{\mathrm{T}}\boldsymbol{P}_2\end{pmatrix}\end{aligned}$$

for $i = 1, \ldots, n$. In addition, we apply the nearest Kronecker product approximation as follows:

$$\begin{aligned}\hat{\mathcal{H}}_1^* &= \boldsymbol{V}_1 \otimes \boldsymbol{V}_1 \simeq \boldsymbol{D}_1 \otimes \boldsymbol{D}_1 + \sum_{i=1}^{n} \boldsymbol{Z}_{i1} \otimes \sigma_i \boldsymbol{Z}_{i1},\\ \hat{\mathcal{H}}_2^* &= \boldsymbol{V}_{2\mathrm{L}} \otimes \boldsymbol{V}_{2\mathrm{R}} \simeq \boldsymbol{D}_1 \otimes \boldsymbol{D}_2 + \sum_{i=1}^{n} \boldsymbol{Z}_{i1} \otimes \sigma_i \boldsymbol{Z}_{i2},\\ \hat{\mathcal{H}}_3^* &= \boldsymbol{V}_3 \otimes \boldsymbol{V}_3 \simeq \boldsymbol{D}_2 \otimes \boldsymbol{D}_2 + \sum_{i=1}^{n} \boldsymbol{Z}_{i2} \otimes \sigma_i \boldsymbol{Z}_{i2},\end{aligned}$$

where

$$\begin{aligned}\boldsymbol{Z}_{i1} &= \sqrt{\sigma_i \lambda_i}\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{U}\,\mathrm{Diag}(\boldsymbol{q}_i)\boldsymbol{U}^{\mathrm{T}}\boldsymbol{P}_1,\\ \boldsymbol{Z}_{i2} &= \sqrt{\sigma_i \lambda_i}\boldsymbol{P}_2^{\mathrm{T}}\boldsymbol{U}\,\mathrm{Diag}(\boldsymbol{q}_i)\boldsymbol{U}^{\mathrm{T}}\boldsymbol{P}_2\end{aligned}$$

for $i = 1, \ldots, n$. Using these operators, we construct an approximation operator of $\mathcal{H}$ as follows:

$$\hat{\mathcal{H}}^* : \begin{pmatrix}\boldsymbol{M}_{11} & \boldsymbol{M}_{21}^{\mathrm{T}}\\ \boldsymbol{M}_{21} & \boldsymbol{M}_{22}\end{pmatrix} \longmapsto \begin{pmatrix}\hat{\mathcal{H}}_1^*(\boldsymbol{M}_{11}) & [\hat{\mathcal{H}}_2^*(\boldsymbol{M}_{21})]^{\mathrm{T}}\\ \hat{\mathcal{H}}_2^*(\boldsymbol{M}_{21}) & \hat{\mathcal{H}}_3^*(\boldsymbol{M}_{22})\end{pmatrix},$$

where $\boldsymbol{M}_{11} \in \mathcal{S}^l$, $\boldsymbol{M}_{21} \in \mathbb{R}^{(n-r-l)\times l}$, and $\boldsymbol{M}_{22} \in \mathcal{S}^{n-r-l}$. Note that, since $\hat{\mathcal{H}}_1^*$, $\hat{\mathcal{H}}_2^*$, and $\hat{\mathcal{H}}_3^*$ are positive definite, $\hat{\mathcal{H}}^* : \mathcal{S}^{n-r} \longrightarrow \mathcal{S}^{n-r}$ is also positive definite. Consequently, we obtain the following preconditioner:

$$\boldsymbol{\Psi}^* = \begin{pmatrix}-(\boldsymbol{P}\otimes\boldsymbol{P})\hat{\mathcal{H}}^*(\boldsymbol{P}\otimes\boldsymbol{P})^{\mathrm{T}} & \mathcal{A}^{\mathrm{T}}\\ \mathcal{A} & \boldsymbol{O}\end{pmatrix}.$$

Diagonal scaling approach

The last preconditioner is a diagonal scaling approach. We can regard this preconditioner as an extension of the preconditioners $\boldsymbol{\Phi}_{\pm}$ proposed by Toh [90]. We can construct block diagonal preconditioners similar to $\boldsymbol{\Phi}_{\pm}$. However, such preconditioners were not efficient numerically in some pilot studies.

In this preconditioning strategy, we use the spectral decomposition $\boldsymbol{W}^{-1} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{\mathrm{T}}$ and represent $\mathcal{H}$ as follows:

$$\mathcal{H} = (\boldsymbol{P}\otimes\boldsymbol{P})\Big[\boldsymbol{D}\otimes\boldsymbol{D} + (\boldsymbol{P}\otimes\boldsymbol{P})^{\mathrm{T}}(\boldsymbol{U}\otimes\boldsymbol{U})\,\mathrm{Diag}(\boldsymbol{Y}^{(-1)}\circ\boldsymbol{T})(\boldsymbol{U}\otimes\boldsymbol{U})^{\mathrm{T}}(\boldsymbol{P}\otimes\boldsymbol{P})\Big](\boldsymbol{P}\otimes\boldsymbol{P})^{\mathrm{T}}.$$

Then, we approximate the inside of the square brackets with the diagonal operator as follows:

$$\begin{aligned}&\boldsymbol{D}\otimes\boldsymbol{D}+(\boldsymbol{P}\otimes\boldsymbol{P})^{\mathrm{T}}(\boldsymbol{U}\otimes\boldsymbol{U})\operatorname{Diag}(\boldsymbol{Y}^{(-1)}\circ\boldsymbol{T})(\boldsymbol{U}\otimes\boldsymbol{U})^{\mathrm{T}}(\boldsymbol{P}\otimes\boldsymbol{P})\\ \simeq\ &\boldsymbol{D}\otimes\boldsymbol{D}+\operatorname{Diag}(\boldsymbol{\Delta})=\operatorname{Diag}[\operatorname{diag}(\boldsymbol{D})\operatorname{diag}(\boldsymbol{D})^{\mathrm{T}}+\boldsymbol{\Delta}].\end{aligned}$$

Note that since $\boldsymbol{D}\otimes\boldsymbol{D}$ is diagonal, we approximated only the second term. In this approximation, the following proposition is useful.

Proposition 2.13. *For a matrix $\boldsymbol{A}\in\mathbb{R}^{m\times n}$ and a vector $\boldsymbol{b}\in\mathbb{R}^{n}$, it hold that*

$$\operatorname{diag}[\boldsymbol{A}\operatorname{Diag}(\boldsymbol{b})\boldsymbol{A}^{\mathrm{T}}]=\boldsymbol{A}^{(2)}\boldsymbol{b}.$$

Proof. Let $\boldsymbol{A}=(\boldsymbol{a}_1,\dots,\boldsymbol{a}_m)^{\mathrm{T}}$, then we obtain

$$\operatorname{diag}[\boldsymbol{A}\operatorname{Diag}(\boldsymbol{b})\boldsymbol{A}^{\mathrm{T}}]=\begin{pmatrix}\boldsymbol{a}_1^{\mathrm{T}}(\boldsymbol{b}\circ\boldsymbol{a}_1)\\ \vdots\\ \boldsymbol{a}_m^{\mathrm{T}}(\boldsymbol{b}\circ\boldsymbol{a}_m)\end{pmatrix}=\begin{pmatrix}(\boldsymbol{a}_1^{(2)})^{\mathrm{T}}\boldsymbol{b}\\ \vdots\\ (\boldsymbol{a}_m^{(2)})^{\mathrm{T}}\boldsymbol{b}\end{pmatrix}=\boldsymbol{A}^{(2)}\boldsymbol{b}.$$

□

This proposition enables us to compute the approximation as follows:

$$\begin{aligned}&(\boldsymbol{P}\otimes\boldsymbol{P})^{\mathrm{T}}(\boldsymbol{U}\otimes\boldsymbol{U})\operatorname{Diag}(\boldsymbol{Y}^{(-1)}\circ\boldsymbol{T})(\boldsymbol{U}\otimes\boldsymbol{U})^{\mathrm{T}}(\boldsymbol{P}\otimes\boldsymbol{P})\\ =\ &(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U}\otimes\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U})\operatorname{Diag}(\boldsymbol{Y}^{(-1)}\circ\boldsymbol{T})(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U}\otimes\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U})^{\mathrm{T}}\\ \simeq\ &(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U}\otimes\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U})^{(2)}(\boldsymbol{Y}^{(-1)}\circ\boldsymbol{T})=[(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U})^{(2)}\otimes(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U})^{(2)}](\boldsymbol{Y}^{(-1)}\circ\boldsymbol{T})\\ =\ &(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U})^{(2)}(\boldsymbol{Y}^{(-1)}\circ\boldsymbol{T})(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U})^{(2)\,\mathrm{T}}=\boldsymbol{\Delta}.\end{aligned}$$

Therefore, we obtain the following preconditioner:

$$\boldsymbol{\Sigma}=\begin{pmatrix}-(\boldsymbol{P}\otimes\boldsymbol{P})\operatorname{Diag}[\operatorname{diag}(\boldsymbol{D})\operatorname{diag}(\boldsymbol{D})^{\mathrm{T}}+\boldsymbol{\Delta}](\boldsymbol{P}\otimes\boldsymbol{P})^{\mathrm{T}} & \mathcal{A}^{\mathrm{T}}\\ \mathcal{A} & \boldsymbol{O}\end{pmatrix}.$$

2.3.3 Implementation details

Tanaka *et al.* [87] have implemented their algorithm in MATLAB version 7.11 and solved multiple instances on the computer used in the previous preliminary experiments. In this subsection, we mention their implementation details of the inexact primal-dual path-following method. First, we describe the implementation of outer iteration of the algorithm. They took an initial point of the algorithm as follows:

$$\boldsymbol{X}=10^3\sqrt{w}\boldsymbol{I},\qquad \boldsymbol{Y}=10^3\boldsymbol{E},\qquad \boldsymbol{y}=\boldsymbol{0},\qquad \boldsymbol{S}=10^3\sqrt{w}\boldsymbol{I},\qquad \boldsymbol{T}=10^3\boldsymbol{E},$$

where $\boldsymbol{E} = \boldsymbol{e}\boldsymbol{e}^{\mathrm{T}}$ and we set the weight parameter to $w = 1 + n - m$. They stopped the algorithm when the relative errors below

$$\frac{\sqrt{\|\boldsymbol{r}_{\mathrm{P}}\|_2^2 + \|\boldsymbol{R}_{\mathrm{P}}\|_{\mathrm{F}}^2}}{1 + \|\boldsymbol{b}\|_{\max}}, \qquad \frac{\|\boldsymbol{R}_{\mathrm{d}}\|_{\mathrm{F}}}{1 + \|\boldsymbol{C}\|_{\max}}, \qquad \frac{\boldsymbol{X} \bullet \boldsymbol{S} + \boldsymbol{Y} \bullet \boldsymbol{T}}{1 + |\boldsymbol{C} \bullet \boldsymbol{X}| + |\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y}|}$$

were less than 10^{-6}. Note that these relative errors correspond to `err1`, `err3`, and `err6` in DIMACS errors, respectively. In the computation of a search direction, they computed μ according to the current parameter

$$\mu = (w\boldsymbol{X} \bullet \boldsymbol{S} + \boldsymbol{Y} \bullet \boldsymbol{T})/[w^2(n - r) + n^2],$$

which is the minimizer of

$$\min_{\mu > 0} \left\{ \|\boldsymbol{X}\boldsymbol{S} - w\mu\boldsymbol{I}\|_{\mathrm{F}}^2 + \|\boldsymbol{Y} \circ \boldsymbol{T} - \mu\boldsymbol{E}\|_{\mathrm{F}}^2 \right\}$$

and we set the parameter $\sigma = 0.3$. Moreover, in the computation of a step length, they set the parameter $\gamma = 0.7$ and if the step lengths of both primal and dual were shorter than 10^{-4}, they stopped the outer iteration.

Next, we describe their implementation of the PSQMR method. They started the algorithm from an initial point $(\Delta\boldsymbol{X}, \Delta\boldsymbol{y}) = (\boldsymbol{O}, \boldsymbol{0})$ and stopped it when

$$\|\boldsymbol{R}\|_{\mathrm{F}} \le \min\left\{ \max\left\{ 10^{-6}(1 + \|\boldsymbol{C}\|_{\infty}), 10^{-2}\|\boldsymbol{R}_{\mathrm{d}}\|_{\mathrm{F}} \right\}, 10^{-2}\|\mathcal{F}^{-1}\boldsymbol{R}_{\mathrm{psd}}\|_{\mathrm{F}} \right\},$$
$$\|\boldsymbol{r}\| \le \max\left\{ 10^{-6}(1 + \|\boldsymbol{b}\|_{\infty}), 10^{-2}\|\boldsymbol{r}_{\mathrm{P}}\| \right\},$$

where $(\boldsymbol{R}, \boldsymbol{r})$ is a residual of the augmented system. They also set the maximum number of iterations of the PSQMR method to 1000 and they restarted the PSQMR method when its number of iterations reached the maximum number of iterations or it was stopped by numerical problems before an accurate solution was obtained. They quit restarting when the number of the restarts exceeded 100 or the total number of iterations exceeded 20000. In such cases, they also stopped the outer iteration. This condition implies that the augmented system (2.31) becomes very ill-conditioned and we cannot obtain an accurate solution unless we allow the PSQMR method to take more iterations.

2.3.4 Preliminary experiments

We show the results of preliminary experiments executed by Tanaka *et al.* [87] to determine whether the algorithm is robust and can be efficient. To this end, they solved (2.16) for the quadratic multidimensional knapsack problem and a portfolio selection problem with the inexact primal-dual path-following algorithm.

They used the following preconditioning strategies:

(P0) No preconditioner was used.

(P1) $\mathcal{V}^*$ was used at every iteration.

(P2) When `err6` > 1, $\mathcal{V}^*$ was used. Otherwise, $\boldsymbol{\Psi}^*$ was used.

(P3) When `err6` > 1, $\mathcal{V}^*$ was used. Otherwise, $\boldsymbol{\Sigma}$ was used.

The reason is they saw that the preconditioners $\boldsymbol{\Psi}^*$ and $\boldsymbol{\Sigma}$ are not effective when the current point is far from optimal from their pilot studies.

First, we will see the results for the quadratic multidimensional knapsack problem

$$\left|\begin{array}{lll} \text{maximize} & \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} & \\ \text{subject to} & \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} \leq b_i & (i = 1, \ldots, m), \\ & \boldsymbol{x} \geq \boldsymbol{0}, & \\ & x_j \in \{0, 1\} & (j = 1, \ldots, n), \end{array}\right. \tag{2.36}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is a decision variable. Note that this problem satisfies the weak key condition [8] since all $\boldsymbol{a}_i$ and $\boldsymbol{b}$ are nonnegative vectors.

The DIMACS errors, the number of iterations of the outer iteration, and computational time for a solution to Problem (2.16) corresponding to Problem (2.36) are given in Table 2.5. In this table, "precond" means the preconditioning strategy that they used. Without preconditioners, they could not obtain feasible solutions for any instances. When they used appropriate preconditioners, in all instances, the feasibilities `err1` and `err3` were almost zero and the duality gaps `err5` and `err6` were around 10^{-4}. These results show that their algorithm gave solutions with moderately high accuracy for Problem (2.16) obtained from Problem (2.36). Comparing the preconditioning strategies, we can see that Strategy (P3) is the most effective, on average.

The relationship between the relative duality gap $(\boldsymbol{X} \bullet \boldsymbol{S} + \boldsymbol{Y} \bullet \boldsymbol{T})/(1 + |\boldsymbol{C} \bullet \boldsymbol{X}^*| + |\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y}^*|)$ and the number of iterations of the PSQMR method for the instance of $n = 50$, $m = 20$ is shown in Figure 2.1, where $\boldsymbol{C} \bullet \boldsymbol{X}^*$ and $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y}^*$ are the optimal values of (2.22) and (2.23), respectively. Each point in this figure indicates a pair of a duality gap and the number of iterations of the PSQMR method to solve the augmented system at each iteration. In the case of (P0), we can see that the iteration did not progress. By contrast, we can see that, when they used appropriate preconditioners, after the convergence of the PSQMR method accelerated. Thus, their algorithm could obtain solutions with moderately high accuracy. In this instance, when they used Strategy (P2), they obtained the most accurate solution but it took many iterations and much time (see also Table 2.5 again).

Next, we consider the following portfolio selection problem:

$$\left|\begin{array}{lll} \text{minimize} & \boldsymbol{x}^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{x} - 2(\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{x})^2 & \\ \text{subject to} & \boldsymbol{e}^{\mathrm{T}}\boldsymbol{x} = 1, & \\ & \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} \leq b_i & (i = 1, \ldots, m), \\ & \boldsymbol{x} \geq 0, & \end{array}\right. \tag{2.37}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are an expectation vector and a covariance matrix of the rate of return of assets, respectively, $\boldsymbol{e}$ is the vector whose elements are all one, and all

$\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} \leq b_i$ are randomly generated constraints. Note that, especially when $m = 0$, (2.37) becomes the standard quadratic optimization problem [7].

The DIMACS errors, the number of iterations, and computational time for a solution to Problem (2.16) corresponding to Problem (2.37) are given in Table 2.6. Without any preconditioners, they could not obtain feasible solutions except for the instance with $m = 0$. For $m = 0$, they could obtain a feasible solution but it was far from the optimal in contrast to the solutions obtained with other strategies. When they used appropriate preconditioners, all of the DIMACS errors were close to zero for all instances. These results indicate that their algorithm gives highly accurate solutions for Problem (2.16) corresponding to Problem (2.37). There were no large differences in computational time among any of the preconditioning strategies.

The relationship between the relative duality gap and the number of iterations of the PSQMR method for the case of $n = 50$, $m = 20$ is shown in Figure 2.2. Without any preconditioners, the PSQMR method failed to converge in an early iteration. By contrast, with appropriate preconditioners, their algorithm behaved just like the case of Problem (2.36). However, in this instance, the number of iterations of the PSQMR method was smaller than 10^3. Thus, we can say that the preconditioners were very effective for Problem (2.16) obtained from Problem (2.37). When they used Strategy (P2), the computational time was longest but the number of iterations of the PSQMR method is smallest. Therefore, this strategy may be a good choice for finding more-accurate solutions.

Tab. 2.5: Results for Problem (2.16) corresponding to Problem (2.36)

n	m	precond	`err1`	`err3`	`err5`	`err6`	iter	time
50	10	(P0)	$6.7 \times 10^{+8}$	$7.5 \times 10^{+1}$	-1.0×10^{0}	$4.3 \times 10^{+1}$	6	7.73
		(P1)	2.3×10^{-11}	8.4×10^{-16}	1.0×10^{-4}	1.0×10^{-4}	55	44.77
		(P2)	2.1×10^{-11}	9.8×10^{-16}	2.2×10^{-3}	2.2×10^{-3}	49	46.50
		(P3)	3.3×10^{-11}	1.2×10^{-15}	3.6×10^{-5}	3.6×10^{-5}	57	46.97
50	20	(P0)	$3.4 \times 10^{+9}$	$1.0 \times 10^{+1}$	-1.0×10^{0}	$3.6 \times 10^{+1}$	4	36.06
		(P1)	2.1×10^{-10}	1.0×10^{-15}	1.1×10^{-4}	1.1×10^{-4}	62	79.31
		(P2)	3.6×10^{-10}	8.2×10^{-16}	1.3×10^{-5}	1.3×10^{-5}	82	205.44
		(P3)	2.7×10^{-10}	8.3×10^{-16}	3.2×10^{-5}	3.2×10^{-5}	65	50.47
50	30	(P0)	$8.2 \times 10^{+7}$	$2.9 \times 10^{+2}$	-1.0×10^{0}	$1.5 \times 10^{+2}$	8	29.50
		(P1)	9.4×10^{-12}	4.1×10^{-16}	6.1×10^{-4}	6.1×10^{-4}	62	54.99
		(P2)	7.1×10^{-12}	4.2×10^{-16}	3.3×10^{-5}	3.3×10^{-5}	70	156.66
		(P3)	8.8×10^{-12}	2.4×10^{-16}	3.0×10^{-4}	3.0×10^{-4}	64	43.36
50	40	(P0)	$3.9 \times 10^{+8}$	$1.1 \times 10^{+2}$	1.0×10^{0}	$5.8 \times 10^{+1}$	1	12.52
		(P1)	5.0×10^{-11}	4.1×10^{-15}	4.2×10^{-4}	4.2×10^{-4}	65	55.61
		(P2)	3.7×10^{-11}	3.9×10^{-15}	3.7×10^{-3}	3.7×10^{-3}	57	52.16
		(P3)	3.1×10^{-11}	4.1×10^{-15}	4.1×10^{-5}	4.1×10^{-5}	71	69.66

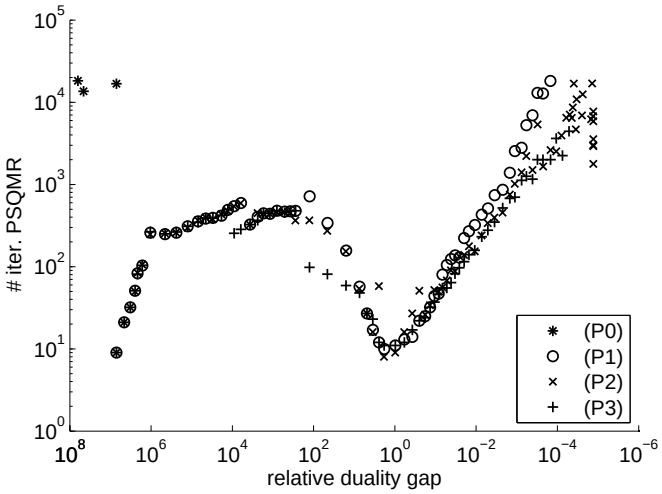

Fig. 2.1: Results for Problem (2.16) corresponding to Problem (2.36) with $n = 50$ and $m = 20$

Tab. 2.6: Results for Problem (2.16) corresponding to Problem (2.37)

n	m	precond	err1	err3	err5	err6	iter	time
50	0	(P0)	2.0×10^{-16}	1.2×10^{-15}	2.1×10^{-2}	2.1×10^{-2}	40	26.26
		(P1)	5.1×10^{-17}	1.0×10^{-15}	8.0×10^{-7}	8.0×10^{-7}	65	12.32
		(P2)	4.7×10^{-16}	1.0×10^{-15}	8.0×10^{-7}	8.0×10^{-7}	65	8.75
		(P3)	2.0×10^{-16}	1.0×10^{-15}	8.0×10^{-7}	8.0×10^{-7}	65	5.21
50	10	(P0)	$1.2 \times 10^{+8}$	8.7×10^{0}	1.0×10^{0}	$1.2 \times 10^{+1}$	5	13.20
		(P1)	7.8×10^{-15}	3.6×10^{-15}	4.2×10^{-7}	4.2×10^{-7}	73	14.32
		(P2)	1.5×10^{-14}	3.6×10^{-15}	4.2×10^{-7}	4.2×10^{-7}	73	14.59
		(P3)	1.3×10^{-14}	3.6×10^{-15}	4.3×10^{-7}	4.3×10^{-7}	73	12.55
50	20	(P0)	$5.4 \times 10^{+3}$	9.2×10^{-12}	1.0×10^{0}	$1.8 \times 10^{+3}$	8	20.16
		(P1)	2.8×10^{-16}	3.7×10^{-15}	9.3×10^{-7}	9.3×10^{-7}	89	11.78
		(P2)	1.9×10^{-16}	3.7×10^{-15}	9.3×10^{-7}	9.3×10^{-7}	89	12.61
		(P3)	1.5×10^{-16}	3.7×10^{-15}	9.3×10^{-7}	9.3×10^{-7}	89	11.82
50	30	(P0)	$9.5 \times 10^{+7}$	6.4×10^{0}	1.0×10^{0}	$2.6 \times 10^{+1}$	2	10.04
		(P1)	6.8×10^{-15}	9.8×10^{-15}	3.7×10^{-7}	3.7×10^{-7}	97	18.86
		(P2)	7.4×10^{-15}	9.8×10^{-15}	3.8×10^{-7}	3.8×10^{-7}	97	19.13
		(P3)	9.8×10^{-15}	9.8×10^{-15}	4.2×10^{-7}	4.2×10^{-7}	97	16.72
50	40	(P0)	$3.7 \times 10^{+6}$	$1.4 \times 10^{+1}$	1.0×10^{0}	$2.3 \times 10^{+2}$	13	30.17
		(P1)	6.6×10^{-16}	2.9×10^{-15}	8.7×10^{-7}	8.7×10^{-7}	117	22.36
		(P2)	7.5×10^{-16}	2.9×10^{-15}	8.8×10^{-7}	8.8×10^{-7}	117	22.94
		(P3)	8.5×10^{-16}	2.9×10^{-15}	9.8×10^{-7}	9.8×10^{-7}	117	17.64

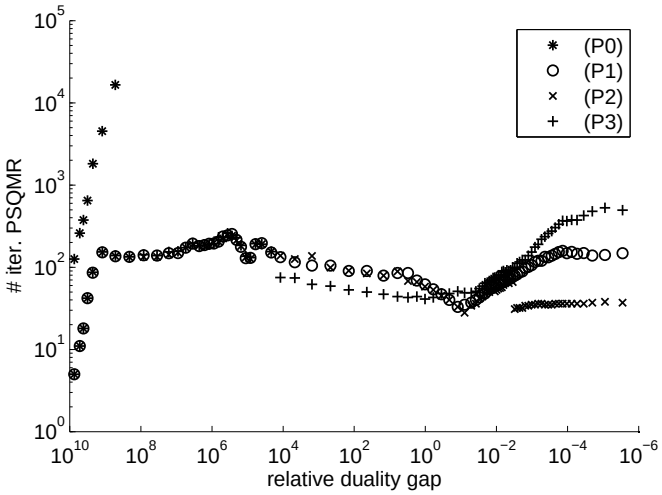

Fig. 2.2: Results for Problem (2.16) corresponding to Problem (2.37) with $n = 50$ and $m = 20$

2.4 Numerical results

In this section, we describe the numerical experiments conducted to verify the efficiency of the approach by Tanaka *et al.* [87]. To this end, they solved the DNN relaxation problems for Problem (2.36) and Problem (2.37). We compared the following three solution methods:

(S1) SDPs equivalent to (2.4) were solved with SDPA.

(S2) SDPs equivalent to (2.16) were solved with SDPA.

(S3) (2.16) were solved with the inexact primal-dual interior-point algorithm.

In these experiments, we limited the computational time to 20000 seconds.

The DIMACS errors `err1`, `err3`, `err5`, `err6`, the numbers of iterations "iter", and computational time in seconds "time" for Problem (2.36) are given in Tables 2.7–2.8. In this table, "solver" means the solution method that they used and "precond" means the preconditioner that we used in (S3). When we used (S2), SDPA ran out of time in most instances, except for the instance $n = 100$, $m = 20$. In this case, although the solution by (S2) is the most accurate, it takes awfully long amount of time. Thus, SDPA cannot quickly solve SDPs equivalent to (2.16) in this manner. When they used (S1) and (S3), they obtained solutions in most instances. In the case where m was small, (S1) was faster than (S3) and the optimalities `err5` and `err6` of the solutions by (S1) are as small as those obtained by (S3). However, when m was large, SDPA required a long computational time. In particular, when $n = 150$, $m = 90$, SDPA ran out of time. By contrast, m did not contribute to computational time of their algorithm.

Moreover, the feasibilities `err1` and `err3` of the solutions produced by (S1) were larger than those by (S3).

The results of solving DNN relaxation problems for Problem (2.37) are given in Tables 2.9–2.10. In the instance $n = 100$, $m = 0$, all the solvers obtained solutions with high accuracy. (S2) was slower than the other solvers and (S3) was as fast as (S1). In the instances with $n = 100$, $m = 20$ and $n = 150$, $m = 0$, all solution methods obtained solutions. The DIMACS errors of the solutions were all sufficiently small and there were only small differences in accuracy in these instance. However, (S3) was faster than (S1) and (S2). In the instances with $n = 100$, $m = 60$ and $n = 150$, $m = 30$, (S1) and (S3) obtained solutions but (S2) ran out of time. The DIMACS errors of the solution obtained by (S1) were larger than those by (S3), and (S1) was much slower than (S3). In the instances with $n = 150$, $m = 60$ and $n = 150$, $m = 90$, only (S3) obtained solutions. The solutions obtained by (S3) was with highly accurate. Moreover, we can see that, for (2.16) of (2.37), (P3) was the most effective preconditioner from the viewpoint of computation time.

Tab. 2.7: Results for DNN relaxation problem for Problem (2.36)

n	m	sol	precond	err1	err3	err5	err6	time
100	20	(S1)	—	1.9×10^{-7}	2.3×10^{-6}	5.1×10^{-4}	5.2×10^{-4}	197.76
		(S2)	—	9.9×10^{-4}	1.3×10^{-12}	8.0×10^{-6}	8.3×10^{-6}	5767.85
		(S3)	(P1)	4.0×10^{-11}	2.1×10^{-15}	4.5×10^{-5}	4.5×10^{-5}	443.39
			(P2)	4.4×10^{-11}	1.4×10^{-15}	4.6×10^{-5}	4.6×10^{-5}	606.88
			(P3)	4.1×10^{-11}	1.8×10^{-15}	2.0×10^{-5}	2.0×10^{-5}	403.16
100	40	(S1)	—	2.8×10^{-7}	4.8×10^{-6}	1.0×10^{-3}	1.1×10^{-3}	444.26
		(S2)	—	—	—	—	—	> 20000.00
		(S3)	(P1)	1.7×10^{-10}	1.2×10^{-14}	1.9×10^{-5}	1.9×10^{-5}	611.50
			(P2)	1.8×10^{-10}	1.2×10^{-14}	2.1×10^{-3}	2.1×10^{-3}	496.43
			(P3)	2.2×10^{-10}	1.1×10^{-14}	4.6×10^{-5}	4.6×10^{-5}	447.75
100	60	(S1)	—	2.4×10^{-7}	3.7×10^{-6}	1.4×10^{-3}	1.5×10^{-3}	989.07
		(S2)	—	—	—	—	—	> 20000.00
		(S3)	(P1)	1.9×10^{-10}	9.9×10^{-13}	4.4×10^{-5}	4.4×10^{-5}	658.53
			(P2)	1.5×10^{-10}	9.9×10^{-13}	4.4×10^{-5}	4.4×10^{-5}	704.39
			(P3)	1.6×10^{-10}	1.0×10^{-12}	9.1×10^{-5}	9.1×10^{-5}	444.73
100	80	(S1)	—	8.6×10^{-8}	5.6×10^{-6}	1.7×10^{-3}	1.8×10^{-3}	1951.81
		(S2)	—	—	—	—	—	> 20000.00
		(S3)	(P1)	1.3×10^{-10}	1.0×10^{-15}	8.3×10^{-3}	8.3×10^{-3}	334.23
			(P2)	1.4×10^{-10}	6.5×10^{-16}	7.3×10^{-5}	7.3×10^{-5}	721.70
			(P3)	1.8×10^{-10}	7.5×10^{-16}	1.9×10^{-4}	1.9×10^{-4}	488.65

Tab. 2.8: Results for DNN relaxation problem of Problem (2.36)

n	m	sol	precond	err1	err3	err5	err6	time
150	30	(S1)	—	2.7×10^{-7}	3.5×10^{-06}	4.9×10^{-4}	5.2×10^{-4}	1912.14
		(S2)	—	—	—	—	—	> 20000.00
		(S3)	(P1)	5.0×10^{-11}	4.8×10^{-15}	2.8×10^{-5}	2.8×10^{-5}	2756.81
			(P2)	5.7×10^{-11}	7.4×10^{-15}	4.1×10^{-3}	4.1×10^{-3}	1593.80
			(P3)	5.9×10^{-11}	7.1×10^{-15}	1.7×10^{-5}	1.7×10^{-5}	2308.51
150	60	(S1)	—	7.7×10^{-8}	5.8×10^{-06}	6.2×10^{-4}	6.5×10^{-4}	5020.98
		(S2)	—	—	—	—	—	> 20000.00
		(S3)	(P1)	4.5×10^{-10}	1.0×10^{-13}	1.1×10^{-4}	1.1×10^{-4}	2580.70
			(P2)	3.8×10^{-10}	1.0×10^{-13}	7.5×10^{-4}	7.5×10^{-4}	1968.93
			(P3)	3.5×10^{-10}	1.0×10^{-13}	3.1×10^{-5}	3.1×10^{-5}	2348.41
150	90	(S1)	—	—	—	—	—	> 20000.00
		(S2)	—	—	—	—	—	> 20000.00
		(S3)	(P1)	9.3×10^{-10}	3.1×10^{-14}	2.6×10^{-5}	2.6×10^{-5}	3207.93
			(P2)	1.0×10^{-9}	3.0×10^{-14}	4.1×10^{-5}	4.1×10^{-5}	4119.35
			(P3)	1.2×10^{-9}	3.0×10^{-14}	6.3×10^{-5}	6.3×10^{-5}	2532.22
150	120	(S1)	—	—	—	—	—	> 20000.00
		(S2)	—	—	—	—	—	> 20000.00
		(S3)	(P1)	6.3×10^{-9}	4.9×10^{-15}	2.9×10^{-5}	2.9×10^{-5}	3931.59
			(P2)	5.0×10^{-9}	4.9×10^{-15}	4.7×10^{-5}	4.7×10^{-5}	5143.90
			(P3)	5.1×10^{-9}	5.0×10^{-15}	1.0×10^{-4}	1.0×10^{-4}	2285.77

Tab. 2.9: Results for DNN relaxation problem of Problem (2.37)

n	m	solver	precond	err1	err3	err5	err6	time
100	0	(S1)	—	4.2×10^{-12}	2.4×10^{-12}	8.1×10^{-7}	8.1×10^{-7}	78.07
		(S2)	—	1.1×10^{-10}	1.1×10^{-12}	3.9×10^{-7}	3.9×10^{-7}	112.85
		(S3)	(P1)	1.1×10^{-16}	2.8×10^{-15}	9.8×10^{-7}	9.8×10^{-7}	78.19
			(P2)	4.4×10^{-16}	2.8×10^{-15}	9.8×10^{-7}	9.8×10^{-7}	62.36
			(P3)	1.7×10^{-16}	2.8×10^{-15}	9.8×10^{-7}	9.8×10^{-7}	52.23
100	20	(S1)	—	3.9×10^{-10}	4.0×10^{-11}	5.4×10^{-6}	5.4×10^{-6}	334.29
		(S2)	—	9.2×10^{-11}	3.5×10^{-12}	7.5×10^{-7}	7.5×10^{-7}	6798.27
		(S3)	(P1)	3.0×10^{-16}	1.0×10^{-14}	8.1×10^{-7}	8.1×10^{-7}	101.85
			(P2)	3.0×10^{-16}	1.0×10^{-14}	8.0×10^{-7}	8.0×10^{-7}	101.66
			(P3)	3.6×10^{-16}	1.0×10^{-14}	8.1×10^{-7}	8.1×10^{-7}	68.86
100	40	(S1)	—	2.6×10^{-10}	1.0×10^{-10}	3.6×10^{-6}	3.6×10^{-6}	774.44
		(S2)	—	—	—	—	—	> 20000.00
		(S3)	(P1)	2.1×10^{-15}	1.6×10^{-14}	9.8×10^{-7}	9.8×10^{-7}	175.95
			(P2)	1.8×10^{-15}	1.6×10^{-14}	4.1×10^{-6}	4.1×10^{-6}	172.37
			(P3)	1.7×10^{-15}	1.6×10^{-14}	9.4×10^{-7}	9.4×10^{-7}	116.26
100	60	(S1)	—	1.1×10^{-7}	1.3×10^{-10}	9.6×10^{-4}	9.9×10^{-4}	1622.76
		(S2)	—	—	—	—	—	> 20000.00
		(S3)	(P1)	4.5×10^{-16}	1.4×10^{-14}	8.3×10^{-7}	8.3×10^{-7}	278.42
			(P2)	3.8×10^{-16}	1.4×10^{-14}	8.4×10^{-7}	8.4×10^{-7}	286.58
			(P3)	4.3×10^{-16}	1.4×10^{-14}	8.5×10^{-7}	8.5×10^{-7}	195.61
100	80	(S1)	—	3.7×10^{-9}	2.0×10^{-10}	8.7×10^{-4}	9.2×10^{-4}	2247.93
		(S2)	—	—	—	—	—	> 20000.00
		(S3)	(P1)	2.6×10^{-16}	1.0×10^{-14}	9.3×10^{-7}	9.3×10^{-7}	324.70
			(P2)	2.4×10^{-16}	1.0×10^{-14}	8.3×10^{-7}	8.3×10^{-7}	323.09
			(P3)	2.3×10^{-16}	1.0×10^{-14}	8.5×10^{-7}	8.5×10^{-7}	206.21

Tab. 2.10: Results for DNN relaxation problems of Problem (2.37)

n	m	solver	precond	err1	err3	err5	err6	time
150	0	(S1)	—	2.1×10^{-12}	4.0×10^{-12}	7.5×10^{-7}	7.5×10^{-7}	704.51
		(S2)	—	3.9×10^{-11}	2.0×10^{-12}	5.9×10^{-7}	5.9×10^{-7}	887.36
		(S3)	(P1)	2.0×10^{-16}	3.8×10^{-15}	6.6×10^{-7}	6.6×10^{-7}	325.28
			(P2)	3.4×10^{-16}	3.8×10^{-15}	6.6×10^{-7}	6.6×10^{-7}	311.83
			(P3)	4.2×10^{-17}	3.8×10^{-15}	6.6×10^{-7}	6.6×10^{-7}	158.51
150	30	(S1)	—	3.8×10^{-9}	6.9×10^{-11}	1.7×10^{-5}	1.7×10^{-5}	3436.14
		(S2)	—	—	—	—	—	> 20000.00
		(S3)	(P1)	5.5×10^{-16}	1.8×10^{-14}	4.6×10^{-7}	4.6×10^{-7}	557.99
			(P2)	7.5×10^{-16}	1.8×10^{-14}	4.6×10^{-7}	4.6×10^{-7}	547.75
			(P3)	8.7×10^{-16}	1.8×10^{-14}	4.6×10^{-7}	4.6×10^{-7}	341.62
150	60	(S1)	—	—	—	—	—	> 20000.00
		(S2)	—	—	—	—	—	> 20000.00
		(S3)	(P1)	2.6×10^{-16}	3.3×10^{-14}	5.9×10^{-7}	5.9×10^{-7}	960.21
			(P2)	2.8×10^{-16}	3.3×10^{-14}	1.2×10^{-6}	1.2×10^{-6}	978.65
			(P3)	2.6×10^{-16}	3.3×10^{-14}	5.4×10^{-7}	5.4×10^{-7}	593.90
150	90	(S1)	—	—	—	—	—	> 20000.00
		(S2)	—	—	—	—	—	> 20000.00
		(S3)	(P1)	2.6×10^{-16}	2.9×10^{-14}	8.5×10^{-7}	8.5×10^{-7}	1757.71
			(P2)	2.3×10^{-16}	2.9×10^{-14}	2.2×10^{-6}	2.2×10^{-6}	1680.13
			(P3)	2.8×10^{-16}	2.9×10^{-14}	1.0×10^{-6}	1.0×10^{-6}	1042.87
150	120	(S1)	—	—	—	—	—	> 20000.00
		(S2)	—	—	—	—	—	> 20000.00
		(S3)	(P1)	2.8×10^{-16}	2.9×10^{-14}	7.6×10^{-7}	7.6×10^{-7}	2436.41
			(P2)	2.6×10^{-16}	2.9×10^{-14}	8.5×10^{-7}	8.5×10^{-7}	2634.78
			(P3)	2.8×10^{-16}	2.9×10^{-14}	7.2×10^{-7}	7.2×10^{-7}	1528.55

2.5 Concluding remarks

In this chapter, we saw the approaches for DNN optimization proposed by Tanaka *et al.* [87, 88]. They overcame the following two difficulties in solving the DNN relaxation problem (2.4) for (2.2): Primal-dual interior-point methods cannot solve them accurately since Problem (2.4) has no interior feasible solutions. Also, the size SDP equivalent to Problem (2.1) is too large to solve by using commonly used software.

To overcome the first difficulty, Tanaka *et al.* [87] have reduced the size of Problem (2.4) by exploiting degeneracy. Tanaka *et al.* [88] have also proposed the numerical reduction method for Problem (2.1). These reduction methods are based on facial reduction algorithms and improved numerical stability of software.

To overcome the second difficulty, Tanaka *et al.* [87] proposed an inexact primal-dual path-following method for (2.22)–(2.23). This algorithm is based on that for QSDPs proposed by Toh [90]. In their algorithm, to compute a search direction, we solve a large linear system via the preconditioned symmetric quasi-minimal residual (PSQMR) method [36]. To accelerate convergence of the PSQMR method, extending preconditioners proposed in [90], they proposed three preconditioners. Numerical results showed that the preconditioners work well and that their algorithm could compute an optimal solution quickly with moderately high accuracy.

An application of facial reduction algorithm for Problem (2.3) has never been proposed. It may enable us to obtain tight bounds for NP-hard optimization problems. Geometrical properties of the completely positive cone [30] may help us to apply the algorithm.

Chapter 3

Well-conditioned matrix approximation

Let us discuss in this chapter well-conditioned matrix approximation problems considered by Tanaka and Nakata [85, 86]. Tanaka and Nakata [85] have considered a problem to find the well-conditioned positive definite matrix which is the nearest to a given matrix. This problem is called the basic problem in this chapter. Tanaka and Nakata [86] have dealt with problems with additional constraints. These problems are introduced in Section 3.1. They have proposed in [85] a reformulation of the basic problem. The basic problem can be converted to a univariate convex optimization problem by employing Ky Fan *p-k* norm. The reformulation technique is shown in Section 3.2. They have also proposed in the same paper solution methods for the resulting univariate problem. In particular, they an analytical solution for the use of the spectral norm and the nuclear norm. Otherwise, they have shown the problem is easily solved by a binary search. We review these ways in Section 3.3. They have proposed in [86] a successive projection method for the extended problem. We see in Section 3.4 their projection algorithm. Section 3.5 shows the numerical results of their algorithms. Section 3.6 provides some concluding remarks.

3.1 Introduction

A matrix $\boldsymbol{X}$ is called *well-conditioned* if its condition number, the ratio of the largest to the smallest singular value, is close to unity. Otherwise, it is called *ill-conditioned.* The condition number of matrices is a commonly used barometer for the numerical stability of them. Well-conditioned matrices are often required in many fields of science and engineering, including signal processing and finance, as the following basic examples indicate:

Example 3.1 (Error of perturbed linear equation). Let us see how numerical error affects the solution to a linear equation. Let $\boldsymbol{x}^*$ be the solution to a linear equation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ and $\boldsymbol{x}^* + \Delta\boldsymbol{x}^*$ be the solution to a *perturbed* linear equation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} + \Delta\boldsymbol{b}$. This is the case where only $\boldsymbol{b}$ includes numerical error. Since $\boldsymbol{x}^*$ is the solution to the original linear equation, $\Delta\boldsymbol{x}^*$ can be regarded as error in the solution caused by the perturbation. It is known that the relative error is known to be bounded in such situation with the condition number of the coefficient

matrix as:

$$\frac{\|\Delta \boldsymbol{x}^*\|_2}{\|\boldsymbol{x}^*\|_2} \leq \mathrm{cond}(\boldsymbol{A}) \frac{\|\Delta \boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2}.$$

See Horn and Johnson [46, Section 5.8] for analysis for a case where $\boldsymbol{A}$ was also perturbed.

Example 3.2 (Convergence rate of the steepest decent method). We consider to solve unconstrained optimization problem $\min f(\boldsymbol{x})$ with the steepest decent method and see how fast the algorithm converges to a local optimal solution $\boldsymbol{x}^*$. Let $\{\boldsymbol{x}^{(k)}\}$ be a sequence generated by the steepest decent method which converges to $\boldsymbol{x}^*$ as $k \longrightarrow \infty$. It is known that the algorithm converges linearly and its asymptotic convergence rate is

$$\left(\frac{1 - \mathrm{cond}(\nabla^2 f(\boldsymbol{x}^*))}{1 + \mathrm{cond}(\nabla^2 f(\boldsymbol{x}^*))} \right)^2.$$

See [64] for a proof.

Unfortunately, matrices that arises from science and engineering is sometimes ill-conditioned as the following examples imply.

Example 3.3 (Covariance estimation from high dimensional data with few samples). Let us consider estimating covariance matrix $\boldsymbol{X}$ of random variable $\boldsymbol{x} \in \mathbb{R}^n$ from observed data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$. The maximum likelihood estimator $\hat{\boldsymbol{X}}$ is

$$\hat{\boldsymbol{X}} = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^{\mathrm{T}}, \tag{3.1}$$

where $\bar{\boldsymbol{x}} = (1/N) \sum_{i=1}^{N} \boldsymbol{x}_i$. Let us consider a situation that the dimension n of the data is large and the number N of them is small. In such cases, the maximum likelihood estimator $\hat{\boldsymbol{X}}$ represented by Equation (3.1) is singular, *i.e.*, extremely ill-conditioned, since $\mathrm{rank}(\hat{\boldsymbol{X}}) \leq N \ll n$. Hence, we cannot compute its inverse, Cholesky decomposition, *etc.*

Example 3.4 (Adjustment of covariance matrices). Let us suppose the estimation of covariance matrix again. Let $\hat{\boldsymbol{X}}$ be an estimator of it. We sometimes make post-hoc modifications to some entries of $\hat{\boldsymbol{X}}$ in practice. For example, when $\hat{X}_{ij} < 0$ appears counter-intuitive to analyst's eye, we often modify the estimate to $\hat{X}_{ij} = \hat{X}_{ji} \geq 0$ as mentioned in [21]. The resulting estimate $\hat{\boldsymbol{X}}$ in this case is not always positive semidefinite. Thus, it can be inappropriate to a covariance matrix. In such situations, it is natural to approximate a given symmetric matrix $\hat{\boldsymbol{X}}$ by the nearest symmetric positive semidefinite matrix $\boldsymbol{X}$. We can formulate it as the following optimization problem:

$$\left| \begin{array}{ll} \text{minimize} & \|\boldsymbol{X} - \hat{\boldsymbol{X}}\| \\ \text{subject to} & \boldsymbol{X} \in \mathcal{S}_+^n. \end{array} \right. \tag{3.2}$$

When we use the Frobenius norm, we can easily prove that $\boldsymbol{X}^* = \sum_{i:\lambda_i > 0} \lambda_i \boldsymbol{p}_i \boldsymbol{p}_i^{\mathrm{T}}$ is the unique optimal solution, where $\hat{\boldsymbol{X}} = \sum_{i=1}^{n} \lambda_i \boldsymbol{p}_i \boldsymbol{p}_i^{\mathrm{T}}$ is the spectral decomposition. Hence the resulting $\boldsymbol{X}^*$ is singular unless $\hat{\boldsymbol{X}}$ is positive definite.

Such instability in estimator of covariance matrices has been of interest in statistics and many alternative estimators have been proposed [3, 22, 29, 41, 49, 60, 99]. The estimator studied by Won and Kim [99] and Aubry *et al.* [3] is especially one that can maximize the likelihood under a semidefinite constraint and a condition number constraint. They have derived analytical solutions to the estimators under the multivariate Gaussian distribution.

We generally cannot always obtain such matrices easily. It is natural to approximate a given ill-conditioned matrix with the nearest well-conditioned one in such cases in view of optimization. Tanaka and Nakata [85] proposed ways of obtaining the nearest well-conditioned positive definite matrix by solving the following optimization problem:

$$\left|\begin{array}{ll} \text{minimize} & \|\boldsymbol{X} - \hat{\boldsymbol{X}}\| \\ \text{subject to} & \boldsymbol{X} \in \mathcal{S}_+^n, \\ & \mathrm{cond}(\boldsymbol{X}) \leq \kappa, \end{array}\right. \tag{3.3}$$

where $\mathrm{cond}(\boldsymbol{X})$ is the condition number of $\boldsymbol{X}$ and $\kappa \geq 1$ is a given upper bound for the condition number of $\boldsymbol{X}$. Here, we define $\mathrm{cond}(\boldsymbol{O}) = 1$ in this thesis. Since $\boldsymbol{X}$ is positive semidefinite, we can see $\mathrm{cond}(\boldsymbol{X}) = \lambda_{\max}(\boldsymbol{X})/\lambda_{\min}(\boldsymbol{X})$. Hence, we obtain the following problem equivalent to (3.3):

$$\left|\begin{array}{ll} \text{minimize} & \|\boldsymbol{X} - \hat{\boldsymbol{X}}\| \\ \text{subject to} & \boldsymbol{X} \in \mathcal{S}_+^n, \\ & \lambda_{\max}(\boldsymbol{X}) \leq \kappa\lambda_{\min}(\boldsymbol{X}). \end{array}\right. \tag{3.4}$$

In the remainder of this chapter, we refer this problem as the *basic problem*.

When we use an appropriate norm as a metric, we can convert this problem to a symmetric cone optimization problem. For example, if we use the Frobenius norm, Problem (3.4) is equivalent to the following problem:

$$\left|\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_{\mathrm{F}} \leq t, \\ & \boldsymbol{X} - \mu\boldsymbol{I} \in \mathcal{S}_+^n, \\ & \kappa\mu\boldsymbol{I} - \boldsymbol{X} \in \mathcal{S}_+^n. \end{array}\right. \tag{3.5}$$

Note that the first constraint can be expressed as a second-order cone constraint. Hence, we can theoretically solve such problem in polynomial time but we cannot solve large instances with this approach within a practical time.

Tanaka and Nakata [85] proved that (3.4) is solvable in $O(n^3)$ computational time by using some commonly used norms. Their analysis is as follows: First, they demonstrated that Problem (3.4) can be converted to a simpler problem when we use a *unitary similarity invariant norm*. They also proved a fundamental inequality on unitary similarity invariant norms to derive it. A matrix norm $\|\cdot\|$ on $\mathbb{C}^{n\times n}$ is called unitary similarity invariant if for any matrix $\boldsymbol{X} \in \mathbb{C}^{n\times n}$ and any unitary matrix $\boldsymbol{U} \in \mathbb{C}^{n\times n}$:

$$\|\boldsymbol{U}\boldsymbol{X}\boldsymbol{U}^{\mathrm{H}}\| = \|\boldsymbol{X}\|,$$

where $\boldsymbol{U}^{\mathrm{H}}$ denotes the conjugate transposition of $\boldsymbol{U}$. For more details about this norm, see Horn and Johnson [47]. They also demonstrated that Problem (3.4) can be converted to a univariate piecewise convex optimization problem when we use a *Ky Fan p-k norm* as a unitary similarity invariant norm. The Ky Fan *p-k* norm of $\boldsymbol{X} \in \mathbb{C}^{m \times n}$ is defined by

$$\|\boldsymbol{X}\|_{p,k} = \left(\sum_{i=1}^{k} \sigma_i(\boldsymbol{X})^p \right)^{1/p},$$

where $\sigma_i(\boldsymbol{X})$ denotes the i-th largest singular value of $\boldsymbol{X}$ and $p \geq 1$ and $k \in \{1, \ldots, \min\{m, n\}\}$ are parameters For more information about this norm, see also Horn and Johnson [47]. Note that this norm is unitary similarity invariant. This norm contains commonly used norms as a special cases. It is called the Ky Fan k-norm when $p = 1$ and is called the Schatten p-norm when $k = n$. It is especially called the spectral norm or induced 2-norm when $k = 1$, is called the nuclear norm or the trace norm when $p = 1$ and $k = n$, and is called the Frobenius norm when $p = 2$ and $k = n$. They presented an analytical solutions to Problem (3.4) if we use the spectral norm and the nuclear norm. Otherwise, they proved that we can solve the univariate problem with a binary search.

After short, Tanaka and Nakata [86] have considered the well-conditioned matrix approximation problem with additional constraints. Recall Example 3.4. The optimal solution to (3.4) does not become intuitive in general. Hence, here we should consider the following problem:

$$\left| \begin{array}{lll} \text{minimize} & \|\boldsymbol{X} - \hat{\boldsymbol{X}}\| & \\ \text{subject to} & \boldsymbol{X} \in \mathcal{S}_+^n, & \\ & \lambda_{\max}(\boldsymbol{X}) \leq \kappa \lambda_{\min}(\boldsymbol{X}), & \\ & X_{ij} \geq 0 & ((i,j) \in P), \\ & X_{ij} \leq 0 & ((i,j) \in N), \end{array} \right. \tag{3.6}$$

where $P, N \subset \{1, \ldots, n\}^2$ are given sets of indices corresponding to the sign constraints.

This problem has a relationship with sparse inverse covariance estimation [24, 27, 37]. Let us consider a multivariate Gaussian distribution with covariance matrix $\boldsymbol{\Sigma}$. It is known that $[\boldsymbol{\Sigma}^{-1}]_{ij} = 0$ if and only if variables i and j are conditionally independent. Thus, if we know set I_0 of pairs of variables that are conditionally independent *a priori*, we can approximate sample covariance matrix $\hat{\boldsymbol{\Sigma}}$ with positive definite matrix $\boldsymbol{\Sigma}$ that satisfies $[\boldsymbol{\Sigma}^{-1}]_{ij} = 0$ for all $(i, j) \in I_0$ and $\mathrm{cond}(\boldsymbol{\Sigma}) \leq c$ by solving (3.6) with $\hat{\boldsymbol{X}} = \hat{\boldsymbol{\Sigma}}^{-1}$, $\kappa = c$, and $P = N = I_0$ and taking $\boldsymbol{\Sigma} = (\boldsymbol{X}^*)^{-1}$, where $\boldsymbol{X}^*$ is the optimal solution.

In addition, we also consider the following problem to approximate a cor-

relation matrix:

$$\left|\begin{array}{lll} \text{minimize} & \|\boldsymbol{X} - \hat{\boldsymbol{X}}\| & \\ \text{subject to} & \boldsymbol{X} \in \mathcal{S}_+^n, & \\ & \lambda_{\max}(\boldsymbol{X}) \leq \kappa \lambda_{\min}(\boldsymbol{X}), & \\ & X_{ii} = 1 & (i = 1, \ldots, n), \\ & X_{ij} \geq 0 & ((i, j) \in P), \\ & X_{ij} \leq 0 & ((i, j) \in N). \end{array}\right. \tag{3.7}$$

In what follows, we can assume $(i, i) \notin N$ for all i without loss of generality. The reason for this is that when $(i, i) \in N$ for some i, (3.6) has no feasible solution other than $\boldsymbol{X} = \boldsymbol{O}$ and (3.7) becomes infeasible. We also assume $\kappa > 1$. Otherwise, we can prove that $\boldsymbol{X} = (\mathrm{tr}(\hat{\boldsymbol{X}})/n)\boldsymbol{I}$ is optimal for both problems.

When $P = N = \emptyset$, (3.7) corresponds to the nearest correlation matrix problem with the condition number constraint. Although the nearest correlation matrix problem has been extensively studied [44, 78], there are few researchers who have considered the condition number of the correlation matrix simultaneously.

Since (3.6) and (3.7) can also be formulated as symmetric cone optimization problems like (3.5), they are also solvable in polynomial time with an interior-point method However, it is still difficult to solve large-scale instances at reasonable computational cost.

Tanaka and Nakata [86] proposed an efficient algorithm for (3.6) and (3.7) employing the *successive projection method* [15, 32, 42], which is a classical algorithm for finding the projection to the intersection of multiple convex sets. In this algorithm, we solve (3.2) by utilizing the binary search proposed by Tanaka and Nakata [85] as a subroutine in this algorithm.

3.2 Reformulation of basic problem

This section explains how we can reformulate Problem (3.4) to a simpler one when we use a unitary similarity invariant norm and how we can reformulate it to a univariate piecewise convex optimization problem when we use a Ky Fan *p-k* norm.

3.2.1 Use of unitary similarity invariant norm

When we use a unitary similarity invariant norm in Problem (3.4), it can be converted to one whose variables are the eigenvalues of $\boldsymbol{X}$ as the following theorem implies.

Theorem 3.5. *Let the norm in Problem* (3.4) *be a unitary similarity invariant norm and* $\hat{\boldsymbol{X}} = \boldsymbol{P}\,\mathrm{Diag}(\hat{\boldsymbol{\lambda}})\boldsymbol{P}^{\mathrm{T}}$ *be the spectral decomposition. Then, for any optimal solution* $\boldsymbol{\lambda}^*$ *to the following problem*

$$\left|\begin{array}{ll} \text{minimize} & \|\,\mathrm{Diag}(\boldsymbol{\lambda} - \hat{\boldsymbol{\lambda}})\| \\ \text{subject to} & \min_i \lambda_i \geq 0, \\ & \max_i \lambda_i \leq \kappa \min_i \lambda_i, \end{array}\right. \tag{3.8}$$

$\boldsymbol{X}^* = \boldsymbol{P}\operatorname{Diag}(\boldsymbol{\lambda}^*)\boldsymbol{P}^{\mathrm{T}}$ *is an optimal solution to Problem* (3.4)*.*

To prove this theorem, we introduce the following inequality on unitary similarity invariant norms.

Lemma 3.6. *For any n-dimensional Hermite matrix* $\boldsymbol{X}$ *and for any unitary similarity invariant norm* $\|\cdot\|$ *on the space of n-dimensional Hermite matrices it hold that*

$$\|\operatorname{Diag}(\operatorname{diag}(\boldsymbol{X}))\| \leq \|\boldsymbol{X}\|.$$

Proof. Let $\boldsymbol{X} = \boldsymbol{P}\operatorname{Diag}(\boldsymbol{\lambda})\boldsymbol{P}^{\mathrm{H}}$ be the spectral decomposition and $\boldsymbol{S} = \boldsymbol{P} \circ \bar{\boldsymbol{P}}$, where $\bar{\boldsymbol{P}}$ is denotes the matrix with complex conjugate entries of $\boldsymbol{P}$. Since $\boldsymbol{P}$ is a unitary matrix, $\boldsymbol{S}$ is a doubly stochastic matrix, *i.e.*, it hold that $\boldsymbol{S} \geq \boldsymbol{O}$, $\boldsymbol{S}\boldsymbol{e} = \boldsymbol{e}$, $\boldsymbol{e}^{\mathrm{T}}\boldsymbol{S} = \boldsymbol{e}^{\mathrm{T}}$. By using Birkhoff's theorem [46, Theorem 8.7.2], there exist $N \in \mathbb{N}$, permutation matrices $\boldsymbol{\Pi}_1, \ldots, \boldsymbol{\Pi}_N$, and positive scalars $\alpha_1, \ldots, \alpha_N$ such that $\sum_{i=1}^{n} \alpha_i = 1$ and $\boldsymbol{S} = \sum_{i=1}^{N} \alpha_i \boldsymbol{\Pi}_i$. Since $\operatorname{diag}(\boldsymbol{X}) = \boldsymbol{S}\boldsymbol{\lambda}$,

$$\begin{aligned}
\|\operatorname{Diag}(\operatorname{diag}(\boldsymbol{X}))\| = \|\operatorname{Diag}(\boldsymbol{S}\boldsymbol{\lambda})\| &= \left\|\operatorname{Diag}\left(\sum_{i=1}^{N} \alpha_i \boldsymbol{\Pi}_i \boldsymbol{\lambda}\right)\right\| \\
&= \left\|\sum_{i=1}^{N} \alpha_i \boldsymbol{\Pi}_i \operatorname{Diag}(\boldsymbol{\lambda})\boldsymbol{\Pi}_i^{\mathrm{T}}\right\| \leq \sum_{i=1}^{N} \alpha_i \|\boldsymbol{\Pi}_i \operatorname{Diag}(\boldsymbol{\lambda})\boldsymbol{\Pi}_i^{\mathrm{T}}\| \\
&= \sum_{i=1}^{N} \alpha_i \|\operatorname{Diag}(\boldsymbol{\lambda})\| = \|\operatorname{Diag}(\boldsymbol{\lambda})\| \\
&= \|\boldsymbol{P}\operatorname{Diag}(\boldsymbol{\lambda})\boldsymbol{P}^{\mathrm{H}}\| = \|\boldsymbol{X}\|.
\end{aligned}$$

□

We prove Theorem 3.5 by using this lemma.

Proof of Theorem 3.5. We prove this theorem by contradiction. For any optimal solution $\boldsymbol{\lambda}^*$ to Problem (3.8), let $\boldsymbol{X}^* = \boldsymbol{P}\operatorname{Diag}(\boldsymbol{\lambda}^*)\boldsymbol{P}^{\mathrm{T}}$. It is clear that $\boldsymbol{X}^*$ is a feasible solution to Problem (3.4). Hence, we only have to prove its optimality. Let us assume that $\boldsymbol{X}^*$ is not an optimal solution to Problem (3.4), Then there exist $\boldsymbol{X}^\dagger$ such that $\|\boldsymbol{X}^\dagger - \hat{\boldsymbol{X}}\| < \|\boldsymbol{X}^* - \hat{\boldsymbol{X}}\|$, $\boldsymbol{X}^\dagger \in \mathcal{S}_+^n$, and $\lambda_{\max}(\boldsymbol{X}^\dagger) \leq \kappa\lambda_{\min}(\boldsymbol{X}^\dagger)$. Let $\boldsymbol{\lambda}^\dagger = \operatorname{diag}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{X}^\dagger\boldsymbol{P})$, and we obtain

$$\begin{aligned}
\max_i \lambda_i^\dagger \leq \lambda_{\max}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{X}^\dagger\boldsymbol{P}) &= \lambda_{\max}(\boldsymbol{X}^\dagger) \\
&\leq \kappa\lambda_{\min}(\boldsymbol{X}^\dagger) = \kappa\lambda_{\min}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{X}^\dagger\boldsymbol{P}) \leq \kappa \min_i \lambda_i^\dagger.
\end{aligned}$$

This implies that $\boldsymbol{\lambda}^\dagger$ is a feasible solution to Problem (3.8). However, using Lemma 3.6, we obtain

$$\begin{aligned}
\|\operatorname{Diag}(\boldsymbol{\lambda}^\dagger - \hat{\boldsymbol{\lambda}})\| &= \|\operatorname{Diag}(\operatorname{diag}(\boldsymbol{P}^{\mathrm{T}}(\boldsymbol{X}^\dagger - \hat{\boldsymbol{X}})\boldsymbol{P}))\| \\
&\leq \|\boldsymbol{P}^{\mathrm{T}}(\boldsymbol{X}^\dagger - \hat{\boldsymbol{X}})\boldsymbol{P}\| = \|\boldsymbol{X}^\dagger - \hat{\boldsymbol{X}}\| \\
&< \|\boldsymbol{X}^* - \hat{\boldsymbol{X}}\| = \|\boldsymbol{P}^{\mathrm{T}}(\boldsymbol{X}^* - \hat{\boldsymbol{X}})\boldsymbol{P}\| \\
&= \|\operatorname{Diag}(\boldsymbol{\lambda}^* - \hat{\boldsymbol{\lambda}})\|.
\end{aligned}$$

It contradicts the optimality of $\boldsymbol{\lambda}^*$. Therefore, $\boldsymbol{X}^*$ is an optimal solution to Problem (3.4). □

3.2.2 Use of Ky Fan *p*-*k* norm

Next, we consider using a Ky Fan *p*-*k* norm in Problem (3.4) as a unitary similarity invariant norm. Since the singular values in this case coincide with the absolute values of the eigenvalues, Problem (3.8) can be converted to the following problem:

$$\left|\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=1}^{k} |\boldsymbol{\lambda} - \hat{\boldsymbol{\lambda}}|_{(i)}^{p} \\ \text{subject to} & \min_i \lambda_i \geq 0, \\ & \max_i \lambda_i \leq \kappa \min_i \lambda_i, \end{array}\right. \tag{3.9}$$

where $|\boldsymbol{x}|$ for $\boldsymbol{x} \in \mathbb{R}^n$ denotes the vector whose elements are absolute values of each element of $\boldsymbol{x}$, and $\boldsymbol{x}_{(i)}$ denotes the i-th largest element of $\boldsymbol{x}$. This problem can be converted to a univariate one as the following theorem implies.

Theorem 3.7. *We assume $\hat{\boldsymbol{X}}$ is an infeasible solution to Problem* (3.4)*. Let $\hat{\boldsymbol{X}} = \boldsymbol{P}\,\mathrm{Diag}(\hat{\boldsymbol{\lambda}})\boldsymbol{P}^{\mathrm{T}}$ be the spectral decomposition such that $\hat{\lambda}_1 \leq \cdots \leq \hat{\lambda}_n$. Let $f_{p,k,\kappa,\hat{\boldsymbol{\lambda}}}(\mu)$ be the univariate function on $\mu \geq 0$ that returns the sum of the largest k items of*

$$(\mu - \hat{\lambda}_1)^p, \ldots, (\mu - \hat{\lambda}_l)^p, (\hat{\lambda}_u - \kappa\mu)^p, \ldots, (\hat{\lambda}_n - \kappa\mu)^p$$

when $\hat{\lambda}_l \leq \mu \leq \hat{\lambda}_{l+1}$ and $\hat{\lambda}_{u-1} \leq \kappa\mu \leq \hat{\lambda}_u$. Also, let μ^* be an optimal solution to the following optimization problem:

$$\left|\begin{array}{ll} \text{minimize} & f_{p,k,\kappa,\hat{\boldsymbol{\lambda}}}(\mu) \\ \text{subject to} & \mu \geq 0. \end{array}\right. \tag{3.10}$$

Then, $\boldsymbol{X}^* = \boldsymbol{P}\,\mathrm{Diag}(\boldsymbol{\lambda}^*)\boldsymbol{P}^{\mathrm{T}}$ is an optimal solution to Problem (3.4)*, where $\boldsymbol{\lambda}^*$ is defined as*

$$\lambda_1^* = \cdots = \lambda_l^* = \mu^*, \lambda_{l+1}^* = \hat{\lambda}_{l+1}, \ldots, \lambda_{u-1}^* = \hat{\lambda}_{u-1}, \lambda_u^* = \cdots = \lambda_n^* = \kappa\mu^*. \tag{3.11}$$

Proof. By using Theorem 3.5, we only have to show that $\boldsymbol{\lambda}^*$ defined by Equation (3.11) is an optimal solution to Problem (3.9). We convert Problem (3.9) by employing auxiliary variable μ:

$$\left|\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=1}^{k} |\boldsymbol{\lambda} - \hat{\boldsymbol{\lambda}}|_{(i)}^{p} \\ \text{subject to} & \mu \geq 0, \\ & \mu \leq \lambda_i \leq \kappa\mu \quad (i = 1, \ldots, n). \end{array}\right. \tag{3.12}$$

Fixing μ, we obtain the following subproblem:

$$\left|\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=1}^{k} |\boldsymbol{\lambda} - \hat{\boldsymbol{\lambda}}|_{(i)}^{p} \\ \text{subject to} & \mu \le \lambda_i \le \kappa\mu \quad (i = 1, \ldots, n). \end{array}\right. \tag{3.13}$$

We can see that when $\hat{\lambda}_l \le \mu \le \hat{\lambda}_{l+1}$ and $\hat{\lambda}_{u-1} \le \kappa\mu \le \hat{\lambda}_u$, $\boldsymbol{\lambda}(\mu)$ defined by

$$\begin{gathered} \lambda_1(\mu) = \cdots = \lambda_l(\mu) = \mu, \\ \lambda_{l+1}(\mu) = \hat{\lambda}_{l+1}, \ldots, \lambda_{u-1}(\mu) = \hat{\lambda}_{u-1}, \\ \lambda_u(\mu) = \cdots = \lambda_n(\mu) = \kappa\mu \end{gathered}$$

is an optimal solution to Subproblem (3.13). At this point, the optimal value $f_{p,k,\kappa,\hat{\boldsymbol{\lambda}}}(\mu)$ of Subproblem (3.13) is the sum of the largest k items of

$$(\mu - \hat{\lambda}_1)^p, \ldots, (\mu - \hat{\lambda}_l)^p, (\hat{\lambda}_u - \kappa\mu)^p, \ldots, (\hat{\lambda}_n - \kappa\mu)^p.$$

Then, unfixing and activating μ, we obtain the following univariate optimization problem:

$$\left|\begin{array}{ll} \text{minimize} & f_{p,k,\kappa,\hat{\boldsymbol{\lambda}}}(\mu) \\ \text{subject to} & \mu \ge 0. \end{array}\right.$$

A pair of an optimal solution μ^* to this problem and $\boldsymbol{\lambda}(\mu^*) = \boldsymbol{\lambda}^*$ is an optimal solution to Problem (3.12). Therefore, $\boldsymbol{\lambda}^*$ is an optimal solution to (3.9). □

3.3 Solution methods for basic problem

The previous section explained how we could convert the basic problem (3.4) to an univariate piecewise convex optimization problem (3.10) by introducing a Ky Fan *p-k* norm. This section presents an analytical solution to Problem (3.10) whose metric is the spectral norm and the nuclear norm. It also show Problem (3.10) can be solved by using a binary search.

3.3.1 Analytical solutions for particular cases

First, let us consider employing the spectral norm, *i.e.*, a parameter in the Ky Fan *p-k* norm is set at $k = 1$. The following theorem shows an analytical solution to (3.9).

Theorem 3.8. *We assume that $\hat{\boldsymbol{\lambda}}$ with $\hat{\lambda}_1 \le \cdots \le \hat{\lambda}_n$ is infeasible to Problem* (3.8)*. Then, the following μ^* is an optimal solution to Problem* (3.10) *with $k = 1$:*

$$\mu^* = \frac{1}{\kappa + 1} \max\{\hat{\lambda}_1 + \hat{\lambda}_n, 0\}. \tag{3.14}$$

Proof. From the definition of $f_{p,k,\kappa,\hat{\boldsymbol{\lambda}}}(\mu)$, it is clear that

$$f_{p,1,\kappa,\hat{\boldsymbol{\lambda}}}(\mu) = \max\{\mu - \hat{\lambda}_1, \hat{\lambda}_n - \kappa\mu\}.$$

If $\hat{\lambda}_1 + \hat{\lambda}_n < 0$, $\mu^* = 0$ is an optimal solution to Problem (3.10) since $f_{p,1,\kappa,\hat{\boldsymbol{\lambda}}}(\mu) = \mu - \hat{\lambda}_1$. Otherwise,

$$f_{p,1,\kappa,\hat{\boldsymbol{\lambda}}}(\mu) = \begin{cases} \hat{\lambda}_n - \kappa\mu & \text{if } \mu \le (\hat{\lambda}_1 + \hat{\lambda}_n)/(\kappa+1), \\ \mu - \hat{\lambda}_1 & \text{otherwise.} \end{cases}$$

Thus,

$$\mu^* = \frac{1}{\kappa+1}(\hat{\lambda}_1 + \hat{\lambda}_n)$$

is a optimal solution to Problem (3.10). Combining the two results, we can see the optimality of (3.14). □

Next, we presents an analytical solution to Problem (3.9) with the nuclear norm, *i.e.*, the parameters are set at $p = 1$, $k = n$.

Theorem 3.9. *We assume that $\hat{\boldsymbol{\lambda}}$ with $\hat{\lambda}_1 \le \cdots \le \hat{\lambda}_n$ is infeasible to Problem* (3.8) *and $n \le \kappa$. Then, the following μ^* is an optimal solution to Problem* (3.10) *with $p = 1$, $k = n$:*

$$\mu^* = \frac{1}{\kappa}\max\{\hat{\lambda}_n, 0\}. \tag{3.15}$$

Proof. From the definition of $f_{p,k,\kappa,\hat{\boldsymbol{\lambda}}}(\mu)$, it is clear that, when $\hat{\lambda}_l \le \mu \le \hat{\lambda}_{l+1}$, $\hat{\lambda}_{u-1} \le \kappa\mu \le \hat{\lambda}_u$,

$$\begin{aligned} f_{1,n,\kappa,\hat{\boldsymbol{\lambda}}}(\mu) &= \sum_{i=1}^{l}(\mu - \hat{\lambda}_i) + \sum_{i=u}^{n}(\hat{\lambda}_i - \kappa\mu) \\ &= [l - \kappa(n-u+1)]\mu + \text{const.} \end{aligned}$$

If $\hat{\lambda}_n < 0$, since $f_{1,n,\kappa,\hat{\boldsymbol{\lambda}}}(\mu) = n\mu + \text{const.}$, $\mu^* = 0$ is an optimal solution to Problem (3.10). Otherwise,

$$f_{1,n,\kappa,\hat{\boldsymbol{\lambda}}}(\mu) = \begin{cases} (l-\kappa)\mu + \text{const.} & \text{if } \hat{\lambda}_l \le \mu \le \hat{\lambda}_{l+1} \text{ and } \hat{\lambda}_{n-1} \le \kappa\mu \le \hat{\lambda}_n, \\ l\mu + \text{const.} & \text{if } \hat{\lambda}_l \le \mu \le \hat{\lambda}_{l+1} \text{ and } \hat{\lambda}_n \le \kappa\mu. \end{cases}$$

From the assumption,

$$\mu^* = \frac{1}{\kappa}\hat{\lambda}_n$$

is an optimal solution to Problem (3.10). Combining the two results, we obtain the optimality of (3.15). □

Remark 3.10. We have to assume $n \le \kappa$ in this analysis but it is not important in practice since it is usual to use large κ, say 10^6, for double-precision floating point arithmetic.

3.3.2 Binary search

As we will mention below, we can solve Problem (3.10) with a binary search in $O(n \log n)$ of computational time if the spectral decomposition of $\hat{\boldsymbol{X}}$ has already been computed. Hence, we can solve the basic problem (3.4) within $O(n^3)$ of computational time.

Since Problem (3.10) is convex, an optimal solution is point μ^* at which the subgradient of $f_{p,k,\kappa,\hat{\boldsymbol{\lambda}}}$ becomes zero. Let $0 = \mu_0 < \mu_1 < \cdots < \mu_{I-1} < \mu_I$ be indifferencial points of $f_{p,k,\kappa,\hat{\boldsymbol{\lambda}}}$ and $\mu_{I+1} = +\infty$ for the sake of simplicity. If the right-hand derivative $f'_{p,k,\kappa,\hat{\boldsymbol{\lambda}}}$ at μ_i is less than zero, we can see $\mu^* > \mu_i$. Otherwise, we obtain $\mu^* \leq \mu_i$. Employing this, we can solve Problem (3.10) with the binary search shown in Algorithm 3.1. Algorithm 3.1 terminates $O(\log n)$ of iterations since I is at most polynomial of n. Also, the evaluation of $f'_{p,k,\kappa,\hat{\boldsymbol{\lambda}}}$ takes $O(n)$ of computational time. Moreover, the computation of the root $f'_{p,k,\kappa,\hat{\boldsymbol{\lambda}}}(\mu)$ within $\mu_{r-1} \leq \mu \leq \mu_r$ only takes $O(1)$ of computational time. Thus, the algorithm will terminate within $O(n \log n)$ of computational time.

Algorithm 3.1 Binary search for Problem (3.10)

```
1: l := 0, r := I + 1
2: while r − l > 1 do
3:     c := ⌊(l + r)/2⌋
4:     if f'_{p,k,κ,λ̂}(μ_c) < 0 then
5:         l := c
6:     else
7:         r := c
8:     end if
9: end while
10: Let μ* be the minimizer of f_{p,k,κ,λ̂}(μ) = 0 within μ_{r−1} ≤ μ ≤ μ_r.
```

3.4 Successive projection method for extended problem

This section shows an efficient algorithm for Problems (3.6) and (3.7) proposed by Tanaka and Nakata [86]. We can regard the problems as the projection to the intersection of a closed convex cone corresponding to the condition number constraint and a convex polytope corresponding to the remaining linear constraints.

Let us consider the more general problem below:

$$\left| \begin{array}{ll} \text{minimize} & \|\boldsymbol{x} - \hat{\boldsymbol{x}}\| \\ \text{subject to} & \boldsymbol{x} \in C_i \quad (i = 1, \ldots, m), \end{array} \right. \tag{3.16}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is a decision variable, $\|\cdot\|$ is a norm induced by an inner product on $\mathbb{R}^n$, and $C_1, \ldots, C_m \subset \mathbb{R}^n$ are given closed convex sets. An optimal solution to (3.16) can be considered as a projection of $\hat{\boldsymbol{x}}$ to $\bigcap_{i=1}^m C_i$.

We can solve (3.16) with a successive projection method when the computation of each projection to C_i is easy, *i.e.*, for each i we can obtain an optimal solution to the following problem at modest computational cost:

$$\left| \begin{array}{ll} \text{minimize} & \|\boldsymbol{x} - \hat{\boldsymbol{x}}\| \\ \text{subject to} & \boldsymbol{x} \in C_i. \end{array} \right. \tag{3.17}$$

In what follows, $P_i(\hat{\boldsymbol{x}})$ denotes an optimal solution to (3.17). The successive projection method is a classical algorithm to solve (3.16) by successively projecting a point to each set. The pseudo-code of this algorithm is quite simple as seen in Algorithm 3.2. This algorithm was first proposed by Dykstra [32] for a family of closed convex cones. Boyle and Dykstra [15] then extended this algorithm to general closed convex sets in Hilbert space. Thus, this algorithm is also called *Dykstra's algorithm.* Han [42] has given a precise description of the extended algorithm in Euclidean space. The following theorem implies the global convergence of this algorithm.

Algorithm 3.2 Successive projection method for (3.16)

1: $\boldsymbol{x}_m^{(0)} := \hat{\boldsymbol{x}}$, $\boldsymbol{y}_1^{(0)}, \ldots, \boldsymbol{y}_m^{(0)} := \boldsymbol{0}$, and $k = 0$
2: **while** $\boldsymbol{x}_m^{(k)}$ is not optimal **do**
3: $\quad \boldsymbol{x}_0^{(k+1)} := \boldsymbol{x}_m^{(k)}$
4: $\quad$ **for** $i = 1, \ldots, m$ **do**
5: $\quad\quad \boldsymbol{x}_i^{(k+1)} := P_i(\boldsymbol{x}_{i-1}^{(k+1)} + \boldsymbol{y}_i^{(k)})$
6: $\quad\quad \boldsymbol{y}_i^{(k+1)} := \boldsymbol{x}_{i-1}^{(k+1)} + \boldsymbol{y}_i^{(k)} - \boldsymbol{x}_i^{(k+1)}$
7: $\quad$ **end for**
8: $\quad k := k + 1$
9: **end while**

Theorem 3.11 (Han [42, Theorem 4.8])**.** *Let $C_1, \ldots, C_p$ be convex polyhedra and $C_{p+1}, \ldots, C_m$ be closed convex sets such that $(\bigcap_{i=1}^p C_i) \cap (\bigcap_{i=p+1}^m \operatorname{int} C_i) \neq \emptyset$, then sequence $\{\boldsymbol{x}_m^{(k)}\}$ generated by Algorithm 3.2 converges to an optimal solution to* (3.16) *as $k \longrightarrow \infty$.*

Next, we will consider applying this algorithm to our problems. Let us define the following closed convex sets for (3.6):

$$\begin{aligned} C_{\text{poly}} &= \{\boldsymbol{X} \in \mathcal{S}^n : X_{ij} \geq 0 \ ((i,j) \in P), X_{ij} \leq 0 \ ((i,j) \in N)\}, \\ C_{\text{cond}} &= \{\boldsymbol{X} \in \mathcal{S}_+^n : \operatorname{cond}(\boldsymbol{X}) \leq \kappa\} \end{aligned}$$

and for (3.7) we replace C_{poly} with

$$C_{\text{poly}} = \{\boldsymbol{X} \in \mathcal{S}^n : X_{ii} = 1 \ (i = 1, \ldots, n), X_{ij} \geq 0 \ ((i,j) \in P), X_{ij} \leq 0 \ ((i,j) \in N)\}.$$

For both problems, it is easy to see C_{poly} is a closed convex set. The following lemma guarantees that C_{cond} is also closed convex set.

Lemma 3.12. *C_{cond} is a closed convex set.*

Proof. First, we will prove closedness. Since we defined $\text{cond}(\boldsymbol{O}) = 1$ in this paper, we can prove that:

$$C_{\text{cond}} = \{\boldsymbol{X} \in \mathcal{S}_+^n : \lambda_{\max}(\boldsymbol{X}) \leq \kappa\lambda_{\min}(\boldsymbol{X})\}.$$

By using the continuity of $\lambda_{\min}(\cdot)$ and $\lambda_{\max}(\cdot)$ [46, Theorem 2.4.9.2], we can find closedness.

Next, we will prove convexity. We arbitrarily take $\boldsymbol{A}, \boldsymbol{B} \in C_{\text{cond}}$ and $\alpha \in [0, 1]$. In addition, let $\boldsymbol{v}$ be an eigenvector of $\boldsymbol{C} = (1-\alpha)\boldsymbol{A} + \alpha\boldsymbol{B}$ with $\|\boldsymbol{v}\| = 1$ corresponding to $\lambda_{\min}(\boldsymbol{C})$. Then, we obtain the following inequality:

$$\lambda_{\min}(\boldsymbol{C}) = \boldsymbol{v}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{v} = (1-\alpha)\boldsymbol{v}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v} + \alpha\boldsymbol{v}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{v} \geq (1-\alpha)\lambda_{\min}(\boldsymbol{A}) + \alpha\lambda_{\min}(\boldsymbol{B}).$$

Similarly, we can prove the inequality below:

$$\lambda_{\max}(\boldsymbol{C}) \leq (1-\alpha)\lambda_{\max}(\boldsymbol{A}) + \alpha\lambda_{\max}(\boldsymbol{B}).$$

Since $\lambda_{\max}(\boldsymbol{A}) \leq \kappa\lambda_{\min}(\boldsymbol{A})$ and $\lambda_{\max}(\boldsymbol{B}) \leq \kappa\lambda_{\min}(\boldsymbol{B})$, we find the following inequality:

$$\text{cond}(\boldsymbol{C}) = \frac{\lambda_{\max}(\boldsymbol{C})}{\lambda_{\min}(\boldsymbol{C})} \leq \frac{(1-\alpha)\lambda_{\max}(\boldsymbol{A}) + \alpha\lambda_{\max}(\boldsymbol{B})}{(1-\alpha)\lambda_{\min}(\boldsymbol{A}) + \alpha\lambda_{\min}(\boldsymbol{B})} \leq \kappa,$$

which implies convexity. □

We also define $P_{\text{poly}}(\cdot)$ and $P_{\text{cond}}(\cdot)$ as projectors to the corresponding cone. Since the corresponding optimization problem is separable, the computation of $P_{\text{poly}}(\cdot)$ becomes quite easy as:

$$[P_{\text{poly}}(\boldsymbol{X})]_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise if } (i,j) \in P \text{ with } X_{ij} < 0 \text{ or } (i,j) \in N \text{ with } X_{ij} > 0, \\ X_{ij} & \text{otherwise.} \end{cases}$$

In addition, the computation of $P_{\text{cond}}(\cdot)$ is computed in $O(n^3)$ of computational time with a binary search [85]. Thus, we can efficiently apply the successive projection method to our problem. Tanaka and Nakata [86] have presented a successive projection method for (3.6) and (3.7). Their algorithm are shown in Algorithm 3.3.

The following theorem implies global convergence of the algorithm for (3.6) and (3.7).

Theorem 3.13. *A sequence $\{\boldsymbol{X}_{\text{poly}}^{(k)}\}$ generated by Algorithm 3.3 converges to an optimal solution to* (3.6) *and* (3.7) *as $k \longrightarrow \infty$.*

Algorithm 3.3 Successive projection method for (3.6) and (3.7)

1: $\boldsymbol{Y}^{(0)}_{\mathrm{poly}}, \boldsymbol{Y}^{(0)}_{\mathrm{cond}} := \boldsymbol{O}$, $\boldsymbol{X}^{(1)}_{\mathrm{poly}} := P_{\mathrm{poly}}(\hat{\boldsymbol{X}})$, and $k = 1$
2: **while** $\boldsymbol{X}^{(k)}_{\mathrm{poly}}$ is not optimal **do**
3: $\quad \boldsymbol{X}^{(k+1)}_{\mathrm{cond}} := P_{\mathrm{cond}}(\boldsymbol{X}^{(k)}_{\mathrm{poly}} + \boldsymbol{Y}^{(k)}_{\mathrm{cond}})$
4: $\quad \boldsymbol{Y}^{(k+1)}_{\mathrm{cond}} := \boldsymbol{X}^{(k)}_{\mathrm{poly}} + \boldsymbol{Y}^{(k)}_{\mathrm{cond}} - \boldsymbol{X}^{(k+1)}_{\mathrm{cond}}$
5: $\quad \boldsymbol{X}^{(k+1)}_{\mathrm{poly}} := P_{\mathrm{poly}}(\boldsymbol{X}^{(k+1)}_{\mathrm{cond}} + \boldsymbol{Y}^{(k)}_{\mathrm{poly}})$
6: $\quad \boldsymbol{Y}^{(k+1)}_{\mathrm{poly}} := \boldsymbol{X}^{(k+1)}_{\mathrm{cond}} + \boldsymbol{Y}^{(k)}_{\mathrm{poly}} - \boldsymbol{X}^{(k+1)}_{\mathrm{poly}}$
7: $\quad k := k + 1$
8: **end while**

Proof. The C_{poly} is virtually convex polyhedral for each problem. From Lemma 3.12, we can also see the convexity of C_{cond}. Moreover, $\boldsymbol{I} \in C_{\mathrm{poly}} \cap \operatorname{int} C_{\mathrm{cond}}$ holds. Thus, by using Theorem 3.11, we can see that sequence $\{\boldsymbol{X}^{(k)}_{\mathrm{poly}}\}$ converges to an optimal solution to each problem. □

Remark 3.14. Algorithm 3.3 can be applied to the following well-conditioned matrix approximation problem with linear equality constraints:

$$\left|\begin{array}{ll} \text{minimize} & \|\boldsymbol{X} - \hat{\boldsymbol{X}}\| \\ \text{subject to} & \boldsymbol{X} \in \mathcal{S}^n_+, \\ & \operatorname{cond}(\boldsymbol{X}) \leq \kappa, \\ & \mathcal{A}\boldsymbol{X} = \boldsymbol{b}, \end{array}\right.$$

where $\mathcal{A} : \mathcal{S}^n \longrightarrow \mathbb{R}^m$ is a linear mapping and $\boldsymbol{b} \in \mathbb{R}^m$, by defining

$$C_{\mathrm{poly}} = \{\boldsymbol{X} \in \mathcal{S}^n : \mathcal{A}\boldsymbol{X} = \boldsymbol{b}\}.$$

Letting $\mathcal{A}^{\mathrm{T}} : \mathbb{R}^m \longrightarrow \mathcal{S}^n$ be the adjoint of $\mathcal{A}$, we can prove that

$$P_{\mathrm{poly}}(\hat{\boldsymbol{X}}) = \hat{\boldsymbol{X}} + \mathcal{A}^{\mathrm{T}}(\mathcal{A}\mathcal{A}^{\mathrm{T}})^{-1}(\boldsymbol{b} - \mathcal{A}\hat{\boldsymbol{X}}).$$

Note that $\mathcal{A}$ does not change at each iteration. Thus, Algorithm 3.3 works well when $\mathcal{A}\mathcal{A}^{\mathrm{T}}$ has the following nice structures, *e.g.*, when the number m of equality constraints is small, the closed form of $(\mathcal{A}\mathcal{A}^{\mathrm{T}})^{-1}$ is available, and the Cholesky decomposition of $\mathcal{A}\mathcal{A}^{\mathrm{T}}$ becomes sparse.

3.5 Numerical results

Let us see in this section the numerical results to verify the binary search and the successive projection method were effective.

First, let us see the numerical results for Problem (3.4). We solved multiple instances of (3.4) with an interior-point method and the binary search and compared the results. We implemented the binary search on MATLAB 7.14.

Tab. 3.1: Results for small-scale instances of Problem (3.4)

n	time (IPM)	time (BS)
20	2.8×10^{-1}	3.8×10^{-4}
30	9.8×10^{-1}	6.7×10^{-4}
40	3.3×10^{0}	9.8×10^{-4}
50	9.5×10^{0}	1.2×10^{-3}
60	$2.9 \times 10^{+1}$	1.6×10^{-3}
70	$1.0 \times 10^{+2}$	1.9×10^{-3}
80	$3.4 \times 10^{+2}$	2.6×10^{-3}
90	$8.1 \times 10^{+2}$	3.3×10^{-3}
100	$1.6 \times 10^{+3}$	3.8×10^{-3}

Tab. 3.2: Results for large-scale instances of Problem (3.4)

n	time (BS)
128	6.7×10^{-3}
256	2.5×10^{-2}
512	1.3×10^{-1}
1024	7.1×10^{-1}
2048	5.2×10^{0}
4096	$4.3 \times 10^{+1}$
8192	$3.3 \times 10^{+2}$
16384	$2.1 \times 10^{+3}$

We modeled (3.4) with YALMIP 3 [62] and solved the resulting symmetric cone optimization problem with SeDuMi 1.3 [83].

We generated instances of Problem (3.4) as follows: First, we sampled every entry of matrix $\boldsymbol{U} \in \mathbb{R}^{n \times n}$ from the uniform distribution on $[-1, +1)$. Next, we used $\boldsymbol{U} + \boldsymbol{U}^{\mathrm{T}}$ as $\hat{\boldsymbol{X}}$ in (3.4) and set $\kappa = 10^6$.

The results for small-scale instances of (3.4) are shown in Table 3.1. The sizes of instances are shown in "n" columns and the elapsed time in seconds for the interior-point method and the binary search are in the "time (IPM)" columns for the former and "time (BS)" columns for the latter. We can see the following from these tables: As the size of instances became large, the elapsed time for the interior-point method rapidly increased. However, the elapsed time for the binary search gradually increased.

We also solved large-scale instances with the binary search. The results are depicted in Tables 3.2. We can see that the algorithm could quickly solve large-scale instances that the interior-point method could not during the modest computational time.

Next, let us see the numerical results for Problems (3.6) and (3.7) presented in Tanaka and Nakata [86]. They solved multiple instances of Problems (3.6) and (3.7) with their successive projection method and an interior-point method and compared the results. They implemented the algorithm on

Tab. 3.3: Results for small-scale instances of Problem (3.6)

n	time (IPM)	time (SPM)	iter (SPM)
20	3.5×10^{-1}	5.6×10^{-2}	98
30	1.1×10^{0}	9.7×10^{-2}	133
40	3.1×10^{0}	9.7×10^{-2}	94
50	9.1×10^{0}	1.1×10^{-1}	75
60	$2.6 \times 10^{+1}$	1.3×10^{-1}	79
70	$1.0 \times 10^{+2}$	1.8×10^{-1}	78
80	$3.1 \times 10^{+2}$	1.7×10^{-1}	70
90	$7.4 \times 10^{+2}$	2.3×10^{-1}	69
100	$1.6 \times 10^{+3}$	3.0×10^{-1}	71

Tab. 3.4: Results for small-scale instances of Problem (3.7)

n	time (IPM)	time (SPM)	iter (SPM)
20	3.7×10^{-1}	5.5×10^{-2}	97
30	9.7×10^{-1}	9.9×10^{-2}	130
40	3.3×10^{0}	1.7×10^{-1}	154
50	8.8×10^{0}	2.3×10^{-1}	176
60	$2.7 \times 10^{+1}$	2.9×10^{-1}	186
70	$1.0 \times 10^{+2}$	3.7×10^{-1}	195
80	$3.1 \times 10^{+2}$	4.8×10^{-1}	211
90	$7.9 \times 10^{+2}$	7.1×10^{-1}	219
100	$1.4 \times 10^{+3}$	9.8×10^{-1}	238

MATLAB 7.14. In their implementation, the algorithm stopped when $\boldsymbol{X}_{\mathrm{poly}}^{(k)} \in \mathcal{S}_{+}^{n}$ and $\mathrm{cond}(\boldsymbol{X}_{\mathrm{poly}}^{(k)}) \leq (1+10^{-6})\kappa$ hold simultaneously. They also modeled (3.6) and (3.7) with YALMIP 3 [62] and solved the resulting symmetric cone optimization problem with SeDuMi 1.3 [83].

They generated instances of Problems (3.6) and (3.7) as follows. They generated $\hat{\boldsymbol{X}}$ with the same procedure in the previous experiments and set $\kappa = 10^{6}$. They set P and N to the set of indices corresponding to the smallest and the largest off-diagonal $2n$ entries of $\hat{\boldsymbol{X}}$.

The results for small-scale instances of (3.6) and (3.7) are listed in Tables 3.3 and 3.4. The sizes of instances are shown in "n" columns, the elapsed time in seconds for the interior-point method and the successive projection method are in the "time (IPM) [sec.]" columns for the former and "time (SPM) [sec.]" columns for the latter, and the numbers of iterations for the successive projection method are in the "iter (SPM)" columns. We can see the following from these tables: As the size of instances became large, the elapsed time for the interior-point method rapidly increased. However, the elapsed time for their successive projection method gradually increased.

We also solved large-scale instances with the successive projection method. The results are summarized in Tables 3.5 and 3.6. We can see that the algorithm

Tab. 3.5: Results for large-scale instances of Problem (3.6)

n	time (SPM)	iter (SPM)
128	4.3×10^{-1}	60
256	1.4×10^{0}	49
512	5.1×10^{0}	45
1024	$2.4 \times 10^{+1}$	42
2048	$1.5 \times 10^{+2}$	38
4096	$1.1 \times 10^{+3}$	36
8192	$8.4 \times 10^{+3}$	35

Tab. 3.6: Results for large-scale instances of Problem (3.7)

n	time (SPM)	iter (SPM)
128	1.1×10^{0}	237
256	6.0×10^{0}	308
512	$4.5 \times 10^{+1}$	402
1024	$2.7 \times 10^{+2}$	481
2048	$2.5 \times 10^{+3}$	654
4096	$2.5 \times 10^{+4}$	826
8192	$2.7 \times 10^{+5}$	1115

could quickly solve large-scale instances. In addition, we can see that the size of the matrix for (3.6) contributed less to the number of iterations, although it certainly did for (3.7).

Let us loot at how the sequence, $\{\boldsymbol{X}_{\text{poly}}^{(k)}\}$, generated by the successive projection method converged to the approximate optimal solution, $\boldsymbol{X}^*$, obtained from the algorithm for the instance of $n = 100$. The behavior of the error norms, $\|\boldsymbol{X}_{\text{poly}}^{(k)} - \boldsymbol{X}^*\|$, are plotted in Figure 3.1. This figure implies that $\{\boldsymbol{X}_{\text{poly}}^{(k)}\}$ converged to $\boldsymbol{X}^*$ *linearly*. The sequence generated by the algorithm behaved similarly for the other instances. Similar results (Figure 3.2) were also obtained for (3.7).

3.6 Concluding remarks

In this chapter, we saw the analysis of well-conditioned matrix approximation problems studied by Tanaka and Nakata [86, 85]. They derived efficient solution methods for the basic problem by restricting the norm to the Ky Fan *p*-*k* norm in [85]. They also demonstrated that the extended problem with the Frobenius norm can be solved efficiently by employing the successive projection method in [86].

They focused on the simplicity of the resulting problem. However, its appropriateness as a mathematical model is as important. Thus, the (in)appropriateness of the norms should be considered.

Linear convergence of their successive projection method has never proved.

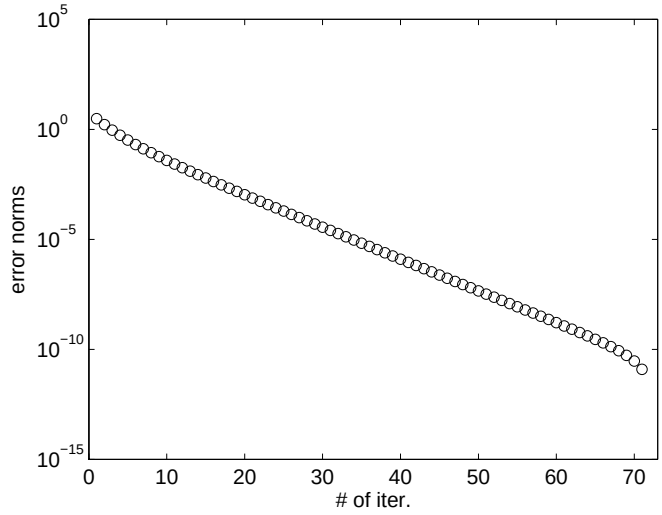

Fig. 3.1: Behavior of error norms $\|\boldsymbol{X}_{\text{poly}}^{(k)} - \boldsymbol{X}^*\|_{\text{F}}$ for Problem (3.6).

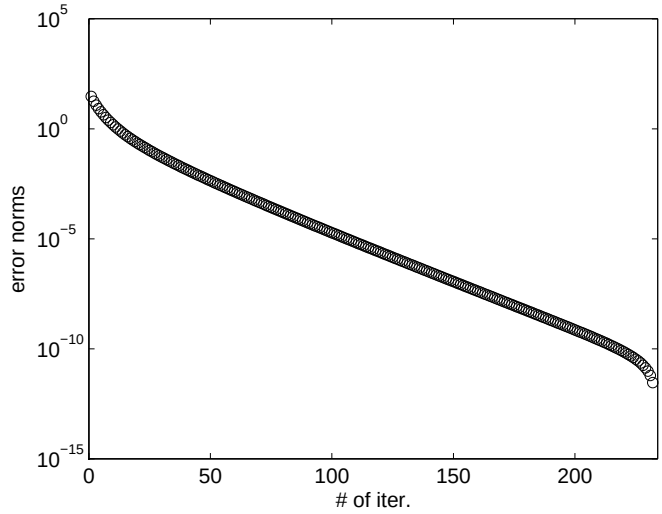

Fig. 3.2: Behavior of error norms $\|\boldsymbol{X}_{\text{poly}}^{(k)} - \boldsymbol{X}^*\|_{\text{F}}$ for Problem (3.7).

Higham [44] also mentioned that the successive projection method (Algorithm 3.2) linearly converges to an optimal solution *at best* for the nearest correlation matrix problem, which is a simpler one than our problems. Deutsch and Hundal [28] proved linear convergence when the convex sets were all subspaces. However, a necessary and sufficient condition to linear convergence by the algorithm has never been known. A *worst-case* analysis of the algorithm should be studied.

Efficient algorithm for Problems (3.6) and (3.7) with different norms other than the Frobenius norm should also be studied. For instance, the confidence of entries in an estimator may not be uniform to approximate a covariance matrix. In such cases weighted norms like $\|\boldsymbol{X}\|_{\boldsymbol{H}} = \|\boldsymbol{H} \circ \boldsymbol{X}\|_{\mathrm{F}} = \sqrt{\sum_{i,j} H_{ij}^2 X_{ij}^2}$ should be used instead of the (unweighted) Frobenius norm $\|\boldsymbol{X}\|_{\mathrm{F}} = \sqrt{\sum_{i,j} X_{ij}^2}$. However, weighted norms are generally not unitary similarity invariant. Hence, $P_{\mathrm{cond}}(\cdot)$ may not be able to be computed easily to use of them since we cannot simply use the reformulation and the binary search proposed in [85].

Chapter 4

Ship navigation problem

In this chapter we deal with the ship navigation problem discussed by Tanaka and Kobayashi [84] and Kobayashi and Tanaka [54]. This problem is to find a shipping route from the origin port to the destination port and shipping speed on each leg in the route that minimize the total fuel consumption and let the ship reach at the destination by the designated time. Tanaka and Kobayashi [84] formulated this problem as an MISOCP. We introduce their MISOCP formulation in Section 4.2. MISOCP can be solved by branch-and-bound-like algorithms. In such algorithms tight bounds for the optimal value is of importance. In Section 4.3, we see the reformulation proposed by Kobayashi and Tanaka [54] that provides tighter bound than the original formulation. Tanaka and Kobayashi [84] have proposed an algorithm named a route generation algorithm. The basic idea of the algorithm is to generate a shipping route and to optimize log speed on each leg in the route iteratively. In Section 4.4, we introduce the route generation algorithm. In Section 4.5, we summarize their numerical experiments and verify the effectiveness of their approaches. Section 4.6 is devoted to the concluding remark.

4.1 Introduction

In this chapter, we consider a problem to find a shipping route from the origin port to the destination port and shipping speed on each leg in the route that minimize the total fuel consumption and let the ship reach at the destination by the designated time. We call this problem *ship navigation problem*. This problem has studied by Tanaka and Kobayashi [84] and Kobayashi and Tanaka [54]. Let us see their results in this chapter. They incorporated the effect of the weather conditions in a deterministic way. More specifically, they assumed that the available information correctly depicts the actual weather conditions. If the difference between the information and the actual condition is small, this would provide a reasonable solution.

Ship routing has been extensively studied from James [48]. Recently, the reduction of the fuel emission has grown in importance since the price of fuel has increased. See the latest survey by Christiansen *et al.* [20]. The fuel emissions depend on a choice of shipping route and shipping speed. Thus, it

is important to optimize them to reduce the fuel emissions in the navigation of ships.

Perakis and Papadakis [76, 77] have considered problems to minimize the cost of operating a fleet of ships. They introduced a model to select shipping speeds of ships in a fleet which carry cargoes between two ports to minimize the annual fleet operating cost.

Problems to optimize shipping speed of a single ship in visiting multiple ports have been considered. Fagerholt *et al.* [33] introduced a model to minimize fuel consumption of a single ship which visits multiple ports in a voyage. The decision variable is shipping speed between each pair of ports in the voyage. In this model, the sequence of ports to visit and the shipping route between two ports are fixed. Ronen [81] considered a problem of a single ship to maximize the revenue, in which the decision variable is shipping speed on a leg. They considered a trade-off between fuel savings through the reduction of the shipping speed and the loss of revenues due to the resulting voyage extension. In the model, the shipping route is fixed.

Problems to optimize the shipping speed between two ports also have been investigated. Lo and McCord [61] considered the effect of the current and treated the problem as an adaptive optimization problem under uncertainty. In their model, the decision variables are the heading of the ship and its log speed (the speed through the water) and the objective is to minimize the expected value of the fuel consumption. Moreover, the ship is required to depart from a origin and arrive at a destination at a prescribed time. Azaron and Kianfar [4] treated the problem as a dynamic shortest path problem in stochastic dynamic networks. In the model, the decision variable is shipping route. They assume that we know the environmental states upon arriving at each node.

In this chapter, a problem to optimize both of a shipping route and log speed on each leg in a route to move from a port to another port is considered. A paper by Tanaka and Kobayashi [84] is the first studies on such a problem. They modeled the ship navigation problem as an optimization problem on a network. In their model, we identified a set of geographical locations on the ocean on which the ship may pass through with the set N of nodes of the network. Especially, nodes $s \in N$ and $t \in N$ represent the origin and the destination, respectively. When the ship could move directly from node i to node j, we define arc (i, j). Let A be the set of such arcs. Thus, a shipping route is represented as an s–t path in network (N, A).

Each arc $(i, j) \in A$ is associated with three values, the distance d_{ij} from node i to node j, a log speed v_{ij} on arc (i, j), and a speed reduction r_{ij} on arc (i, j), the details are described later. A log speed must lie between the minimum value $v_{\min}$ and the maximum value $v_{\max}$.

They assumed that the fuel consumption per unit distance is given as the cubic function $\alpha v^3 + \beta v^2 + \gamma v$ of log speed v for the interval $[v_{\min}, v_{\max}]$. This model has been given by Fagerholt *et al.* [33]. They gave the estimated values of $\alpha = 0.0036$, $\beta = -0.1015$, and $\gamma = 0.8848$ with $v_{\min} = 14$, and $v_{\max} = 20$.

In their model, we take into account the effect of the weather condition. More specifically, the transition time from each node to an adjacent node is

assumed to be dependent on the weather conditions of the locations of the nodes. They assumed that the weather condition on arc (i, j) is represented as a speed reduction of the ship. It is denoted by a constant r_{ij}. In other word, when the ship passes through arc (i, j) with log speed v_{ij}, the ground speed, the speed over the ground, is given by $v_{ij} - r_{ij}$. They also assumed that $r_{ij} < v_{\min}$. Under this assumption, the transition time on arc (i, j) is given by $d_{ij}/(v_{ij} - r_{ij})$. Thus, the fuel consumption on arc (i, j) is given by $(\alpha v_{ij}^3 + \beta v_{ij}^2 + \gamma v_{ij}) d_{ij}/(v_{ij} - r_{ij})$. Then, the ship navigation problem can be formulated as the following mixed-integer nonlinear optimization problem (MINLP):

$$
\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{(i,j)\in A} \frac{\alpha v_{ij}^3 + \beta v_{ij}^2 + \gamma v_{ij}}{v_{ij} - r_{ij}} d_{ij} & \\
\text{subject to} & \displaystyle\sum_{j:(s,j)\in A} x_{sj} = 1, & \\
& \displaystyle\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = 0 & (i \in N \setminus \{s, t\}), \\
& \displaystyle\sum_{i:(i,t)\in A} x_{it} = 1, & \\
& x_{ij} \in \{0, 1\} & ((i, j) \in A), \\
& \displaystyle\sum_{(i,j)\in A} \frac{d_{ij} x_{ij}}{v_{ij} - r_{ij} x_{ij}} \le T, & \\
& v_{\min} x_{ij} \le v_{ij} \le v_{\max} x_{ij} & ((i, j) \in A),
\end{array}
\tag{4.1}
$$

where $x_{ij} = 1$ if the ship passes through arc (i, j) and $x_{ij} = 0$ otherwise, and v_{ij} represents the log speed if arc (i, j) is used and $v_{ij} = 0$ otherwise. Note that if $x_{ij} = 0$ then the corresponding summands in the objective function in (4.1) and those in the fifth constraint in (4.1) vanish. Thus, the objective function and the left-hand side of the fifth constraint represent the total fuel consumption and the total transition time in the route, respectively.

Tanaka and Kobayashi [84] have reformulated Problem (4.1) into an MISOCP by approximating the objective function in Problem (4.1) with a convex quadratic function. They have also proposed a route generation algorithm to solve the MISOCP quickly. Kobayashi and Tanaka [54] have reformulated the MISOCP into another MISOCP that is called a perspective reformulation. They have compared the numerical results for two MISOCP formulations.

4.2 MISOCP reformulation

The ship navigation problem can be described as MINLP (4.1). By approximating the objective function by a quadratic function, we can reformulate Problem (4.1) as an MISOCP.

Each term in objective function can be approximated by a quadratic function

as the followings: It is represented as

$$\frac{\alpha v_{ij}^3 + \beta v_{ij}^2 + \gamma v_{ij}}{v_{ij} - r_{ij}} = \alpha v_{ij}^2 + \beta'_{ij} v_{ij} + \gamma'_{ij} + \frac{\delta_{ij}}{v_{ij} - r_{ij}}, \tag{4.2}$$

where

$$\beta'_{ij} = \alpha r_{ij} + \beta, \qquad \gamma'_{ij} = \alpha r_{ij}^2 + \beta r_{ij} + \gamma, \qquad \delta_{ij} = \alpha r_{ij}^3 + \beta r_{ij}^2 + \gamma r_{ij}.$$

We approximate the right-hand side of (4.2) by a quadratic function using the Taylor approximation. The Taylor approximation at $\bar{v} = (v_{\min} + v_{\max})/2$ of the forth term of the right-hand side of (4.2) is given by

$$\frac{\delta_{ij}}{v_{ij} - r_{ij}} \simeq \frac{\delta_{ij}}{\bar{v} - r_{ij}} - \frac{\delta_{ij}}{(\bar{v} - r_{ij})^2}(v_{ij} - \bar{v}) + \frac{\delta_{ij}}{(\bar{v} - r_{ij})^3}(v_{ij} - \bar{v})^2. \tag{4.3}$$

Then, each term for (i, j) in the objective function in (4.1) is approximated by a quadratic function of log speed v_{ij} below:

$$\tilde{\alpha}_{ij} v_{ij}^2 + \tilde{\beta}_{ij} v_{ij} + \tilde{\gamma}_{ij}, \tag{4.4}$$

where

$$\begin{aligned}
\tilde{\alpha}_{ij} &= \alpha + \frac{\delta_{ij}}{(\bar{v} - r_{ij})^3}, \\
\tilde{\beta}_{ij} &= \beta'_{ij} - \frac{\delta_{ij}}{(\bar{v} - r_{ij})^2} - 2\frac{\delta_{ij}}{(\bar{v} - r_{ij})^3}\bar{v}, \\
\tilde{\gamma}_{ij} &= \gamma'_{ij} + \frac{\delta_{ij}}{\bar{v} - r_{ij}} + \frac{\delta_{ij}}{(\bar{v} - r_{ij})^2}\bar{v} + \frac{\delta_{ij}}{(\bar{v} - r)^3}\bar{v}^2.
\end{aligned}$$

Under a mild assumption, $\tilde{\alpha}_{ij} > 0$ holds. Therefore, Quadratic function (4.4) is convex. In addition, the approximation error in (4.3) is small for small r_{ij}. More specifically, when $v_{ij} \geq \bar{v}$, the approximation error, *i.e.*, the remainder term of the Taylor approximation in (4.3), is $-\delta_{ij}(v - \bar{v})^3/(v^\dagger - r_{ij})^4$ for some $v^\dagger$ with $\bar{v} \leq v^\dagger \leq v_{ij}$. For example, the error for $v_{\min} = 14$, $v_{\max} = 20$, and $r_{ij} = 1$ is evaluated as the followings:

$$\left| -\frac{\delta_{ij}}{(v^\dagger - r_{ij})^4}(v - \bar{v})^3 \right| \leq \frac{\delta_{ij}}{(\bar{v} - r_{ij})^4}(v_{\max} - \bar{v})^3 < 3.3 \times 10^{-4}.$$

Each term of (4.4) should vanish when arc (i, j) is not used, *i.e.*, $x_{ij} = 0$. Thus, we replace the constant term $\tilde{\gamma}_{ij}$ by the linear term $\tilde{\gamma}_{ij} x_{ij}$. As a result of this, we

obtain the following optimization problem:

$$
\left|\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{(i,j)\in A} (\tilde{\alpha}_{ij} v_{ij}^2 + \tilde{\beta}_{ij} v_{ij} + \tilde{\gamma}_{ij} x_{ij}) d_{ij} & \\
\text{subject to} & \displaystyle\sum_{j:(s,j)\in A} x_{sj} = 1, & \\
& \displaystyle\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = 0 & (i \in N \setminus \{s,t\}), \\
& \displaystyle\sum_{i:(i,t)\in A} x_{it} = 1, & \\
& x_{ij} \in \{0,1\} & ((i,j) \in A), \\
& \displaystyle\sum_{(i,j)\in A} \frac{d_{ij} x_{ij}}{v_{ij} - r_{ij} x_{ij}} \le T, & \\
& v_{\min} x_{ij} \le v_{ij} \le v_{\max} x_{ij} & ((i,j) \in A).
\end{array}\right. \tag{4.5}
$$

As we describe below, Problem (4.5) is reformulated as an MISOCP. For each quadratic term of the objective function in (4.5), we introduce an auxiliary variable u_{ij} for $(i,j) \in A$. Then, Problem (4.5) can be described as

$$
\left|\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{(i,j)\in A} (\tilde{\alpha}_{ij} u_{ij} + \tilde{\beta}_{ij} v_{ij} + \tilde{\gamma}_{ij} x_{ij}) d_{ij} & \\
\text{subject to} & \displaystyle\sum_{j:(s,j)\in A} x_{sj} = 1, & \\
& \displaystyle\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = 0 & (i \in N \setminus \{s,t\}), \\
& \displaystyle\sum_{i:(i,t)\in A} x_{it} = 1, & \\
& x_{ij} \in \{0,1\} & ((i,j) \in A), \\
& \displaystyle\sum_{(i,j)\in A} \frac{d_{ij} x_{ij}}{v_{ij} - r_{ij} x_{ij}} \le T, & \\
& v_{\min} x_{ij} \le v_{ij} \le v_{\max} x_{ij} & ((i,j) \in A) \\
& v_{ij}^2 \le u_{ij} & ((i,j) \in A).
\end{array}\right.
$$

Since the last constraint can be described as a restricted hyperbolic constraint, it can be represented as a second-order cone constraint below:

$$
\begin{pmatrix} u_{ij} + 1 \\ u_{ij} - 1 \\ 2v_{ij} \end{pmatrix} \in Q^3 \ ((i,j) \in A).
$$

Also, the fifth constraint can be described as a combination of a linear constraint and second-order cone constraints. For each $(i,j) \in A$, we introduce auxiliary

variable y_{ij}. Then, it can be represented as

$$\sum_{(i,j)\in A} d_{ij} y_{ij} \leq T,$$
$$\frac{x_{ij}}{v_{ij} - r_{ij} x_{ij}} \leq y_{ij} \ ((i,j) \in A). \tag{4.6}$$

Since $v_{ij} \geq v_{\min} > r_{ij}$ and $x_{ij} \in \{0,1\}$, the relations $v_{ij} - r_{ij}x_{ij} \geq 0$ and $x_{ij}^2 = x_{ij}$ hold. From these relations, we see that Inequality (4.6) is represented as

$$x_{ij}^2 \leq y_{ij}(v_{ij} - r_{ij}x_{ij}) \ ((i,j) \in A).$$

Since this inequality can be described as the restricted hyperbolic constraint, this inequality can be represented as a second-order cone constraint below:

$$\begin{pmatrix} y_{ij} + v_{ij} - r_{ij}x_{ij} \\ y_{ij} - v_{ij} + r_{ij}x_{ij} \\ 2x_{ij} \end{pmatrix} \in Q^3 \ ((i,j) \in A).$$

Consequently, Problem (4.5) can be described as the following MISOCP:

$$\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{(i,j)\in A} (\tilde{\alpha}_{ij} u_{ij} + \tilde{\beta}_{ij} v_{ij} + \tilde{\gamma}_{ij} x_{ij}) d_{ij} & \\
\text{subject to} & \displaystyle\sum_{j:(s,j)\in A} x_{sj} = 1, & \\
& \displaystyle\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = 0 & (i \in N \setminus \{s,t\}), \\
& \displaystyle\sum_{i:(i,t)\in A} x_{it} = 1, & \\
& x_{ij} \in \{0,1\} & ((i,j) \in A), \\
& v_{\min} x_{ij} \leq v_{ij} \leq v_{\max} x_{ij} & ((i,j) \in A), \\
& \begin{pmatrix} u_{ij} + 1 \\ u_{ij} - 1 \\ 2v_{ij} \end{pmatrix} \in Q^3 & ((i,j) \in A), \\
& \displaystyle\sum_{(i,j)\in A} d_{ij} y_{ij} \leq T, & \\
& \begin{pmatrix} y_{ij} + v_{ij} - r_{ij}x_{ij} \\ y_{ij} - v_{ij} + r_{ij}x_{ij} \\ 2x_{ij} \end{pmatrix} \in Q^3 & ((i,j) \in A).
\end{array}$$

4.3 Perspective reformulation

We introduce reformulation of Problem (4.5) for which continuous relaxation provides tight lower bound. Replacing v_{ij}^2 in the objective function with v_{ij}^2/x_{ij},

we obtain the following problem:

$$
\left|\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{(i,j)\in A}\left(\tilde{\alpha}_{ij}\frac{v_{ij}^2}{x_{ij}}+\tilde{\beta}_{ij}v_{ij}+\tilde{\gamma}_{ij}x_{ij}\right)d_{ij} & \\
\text{subject to} & \displaystyle\sum_{j:(s,j)\in A} x_{sj}=1, & \\
& \displaystyle\sum_{j:(i,j)\in A} x_{ij}-\sum_{j:(j,i)\in A} x_{ji}=0 & (i\in N\setminus\{s,t\}), \\
& \displaystyle\sum_{i:(i,t)\in A} x_{it}=1, & \\
& x_{ij}\in\{0,1\} & ((i,j)\in A), \\
& \displaystyle\sum_{(i,j)\in A}\frac{d_{ij}x_{ij}}{v_{ij}-r_{ij}x_{ij}}\le T, & \\
& v_{\min}x_{ij}\le v_{ij}\le v_{\max}x_{ij} & ((i,j)\in A),
\end{array}\right. \tag{4.7}
$$

where we define $v_{ij}^2/x_{ij}=0$ for $x_{ij}=v_{ij}=0$. Problem (4.7) is equivalent to (4.5) since $v_{ij}^2/x_{ij}=v_{ij}^2$ for $x_{ij}\in\{0,1\}$. When we solve (4.5) and (4.7) by a branch-and-bound-like procedure, constraint $x_{ij}\in\{0,1\}$ is relaxed by $0\le x_{ij}\le 1$. In such situation, lower bounds for (4.7) is tighter than those for (4.5) since $v_{ij}^2/x_{ij}>v_{ij}^2$ for $0<x_{ij}<1$. We call this formulation *perspective reformulation* of (4.5). It should be noted that Frangioni and Gentle [35, Section 2] discussed more general situation. In their paper, the objective function obtained by such replacement is called the *perspective function*.

Problem (4.7) can also be implemented as an MISOCP. In Problem (4.7), we introduce an auxiliary variable u_{ij} for each $(i,j)\in A$ in the objective function. Then, we obtain the following equivalent formulation:

$$
\left|\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{(i,j)\in A}(\tilde{\alpha}_{ij}u_{ij}+\tilde{\beta}_{ij}v_{ij}+\tilde{\gamma}_{ij}x_{ij})d_{ij} & \\
\text{subject to} & \displaystyle\sum_{j:(s,j)\in A} x_{sj}=1, & \\
& \displaystyle\sum_{j:(i,j)\in A} x_{ij}-\sum_{j:(j,i)\in A} x_{ji}=0 & (i\in N\setminus\{s,t\}), \\
& \displaystyle\sum_{i:(i,t)\in A} x_{it}=1, & \\
& x_{ij}\in\{0,1\} & ((i,j)\in A), \\
& \displaystyle\sum_{(i,j)\in A}\frac{d_{ij}x_{ij}}{v_{ij}-r_{ij}x_{ij}}\le T, & \\
& v_{\min}x_{ij}\le v_{ij}\le v_{\max}x_{ij} & ((i,j)\in A), \\
& \dfrac{v_{ij}^2}{x_{ij}}\le u_{ij} & ((i,j)\in A).
\end{array}\right.
$$

The last constraint can be represented as the second-order cone constraint of

the form

$$\begin{pmatrix} u_{ij} + x_{ij} \\ u_{ij} - x_{ij} \\ 2v_{ij} \end{pmatrix} \in Q^3 \ ((i,j) \in A).$$

Consequently, an MISOCP formulation of (4.7) is described as follows:

$$\begin{array}{ll}
\text{minimize} & \displaystyle\sum_{(i,j)\in A} (\tilde{\alpha}_{ij} u_{ij} + \tilde{\beta}_{ij} v_{ij} + \tilde{\gamma}_{ij} x_{ij}) d_{ij} \\
\text{subject to} & \displaystyle\sum_{j:(s,j)\in A} x_{sj} = 1, \\
& \displaystyle\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = 0 \qquad (i \in N \setminus \{s,t\}), \\
& \displaystyle\sum_{i:(i,t)\in A} x_{it} = 1, \\
& x_{ij} \in \{0,1\} \qquad ((i,j) \in A), \\
& v_{\min} x_{ij} \le v_{ij} \le v_{\max} x_{ij} \qquad ((i,j) \in A), \\
& \begin{pmatrix} u_{ij} + x_{ij} \\ u_{ij} - x_{ij} \\ 2v_{ij} \end{pmatrix} \in Q^3 \qquad ((i,j) \in A), \\
& \displaystyle\sum_{(i,j)\in A} d_{ij} s_{ij} \le T, \\
& \begin{pmatrix} s_{ij} + v_{ij} - r_{ij} x_{ij} \\ s_{ij} - v_{ij} + r_{ij} x_{ij} \\ 2x_{ij} \end{pmatrix} \in Q^3 \qquad ((i,j) \in A).
\end{array}$$

4.4 Route generation algorithm

We have already given the MISOCP formulations of Problem (4.1) in Section 4.2. When we solve them with general-purpose solvers, it may take long computation time. In order to obtain an optimal solution quickly, Tanaka and Kobayashi [84] have proposed the *route generation algorithm*. In their algorithm, we do not optimize a shipping route and log speed simultaneously. Instead, we firstly generate candidates of an optimal shipping route and subsequently optimize the shipping speed on each leg in each candidate route.

Clearly, we can find an optimal solution to Problem (4.5) by enumerating all s–t paths and optimizing shipping speed for each s–t path. For a fixed s–t path, we can optimize the shipping speed on each arc in the path by solving a small SOCP as we see below. Let $P \subset (N, A)$ be a fixed s–t path and $A(P)$ denote the arc set of P. Then, we obtain the optimal shipping speed on each arc $(i, j) \in A(P)$

by solving the following SOCP:

$$\left|\begin{array}{lll} \text{minimize} & \displaystyle\sum_{(i,j)\in A(P)} (\tilde{\alpha}_{ij}u_{ij} + \tilde{\beta}_{ij}v_{ij} + \tilde{\gamma}_{ij})d_{ij} & \\ \text{subject to} & \displaystyle\sum_{(i,j)\in A(P)} d_{ij}y_{ij} \leq T, & \\ & v_{\min} \leq v_{ij} \leq v_{\max} & ((i,j) \in A(P)), \\ & \begin{pmatrix} u_{ij}+1 \\ u_{ij}-1 \\ 2v_{ij} \end{pmatrix} \in \boldsymbol{Q}^3 & ((i,j) \in A(P)), \\ & \begin{pmatrix} y_{ij}+v_{ij}-r_{ij} \\ y_{ij}-v_{ij}+r_{ij} \\ 2 \end{pmatrix} \in \boldsymbol{Q}^3 & ((i,j) \in A(P)). \end{array}\right. \tag{4.8}$$

Note that we can obtain the problem by fixing variable x_{ij} as $x_{ij} = 1$ for $(i,j) \in A(P)$ and $x_{ij} = 0$ for $(i,j) \notin A(P)$ in Problem (4.5). We can solve Problem (4.8) quickly with interior-point methods since the size of (4.8) is much smaller than the continuous relaxation problem of (4.5). Suppose that we have an optimal s–t path P^* but do not know an optimal speed v^*_{ij} on each arc $(i,j) \in A(P^*)$. In such a case, v^*_{ij} can easily be obtained by solving (4.8) corresponding to P^*.

Note that an optimal s–t path P^* is expected to be a short s–t path. Based on this observation, we generate short s–t paths and optimize speeds on arcs in each generated s–t path. In order to generate short s–t paths, we introduce the following constrained shortest path problem:

$$\left|\begin{array}{lll} \text{minimize} & \displaystyle\sum_{(i,j)\in A} (\tilde{\alpha}_{ij}\tilde{v}^2_{ij} + \tilde{\beta}_{ij}\tilde{v}_{ij} + \tilde{\gamma}_{ij})d_{ij}x_{ij} & \\ \text{subject to} & \displaystyle\sum_{j:(s,j)\in A} x_{sj} = 1, & \\ & \displaystyle\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = 0 & (i \in N \setminus \{s,t\}), \\ & \displaystyle\sum_{i:(i,t)\in A} x_{it} = 1, & \\ & x_{ij} \in \{0,1\} & ((i,j) \in A), \\ & \displaystyle\sum_{(i,j)\in A} \frac{d_{ij}}{\tilde{v}_{ij} - r_{ij}} x_{ij} \leq T, & \end{array}\right. \tag{4.9}$$

where $\tilde{v}_{ij}$ is a fixed shipping speed. Note that we can obtain this problem by fixing variable v_{ij} as $\tilde{v}_{ij}$ in Problem (4.5). By solving this problem, we can find a short s–t path P. Subsequently, we solve (4.8) corresponding to P and obtain an approximate optimal solution of Problem (4.5). After solving (4.8), we search another short s–t path by solving Problem (4.9). One of the simplest way to find another s–t path is to add the constraint below to (4.9):

$$\sum_{(i,j)\in A(P)} x_{ij} \leq |A(P)| - 1. \tag{4.10}$$

Although we can optimize shipping speed v_{ij} by solving (4.8) in polynomial time, it is difficult to solve (4.8) for all s–t paths since there are an enormous number of s–t paths. To overcome this difficulty, we make it possible to stop the enumeration of s–t paths. For this purpose, we introduce the following optimization problem:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in A} (\tilde{\alpha}_{ij}(v_{ij}^*)^2 + \tilde{\beta}_{ij} v_{ij}^* + \tilde{\gamma}_{ij}) d_{ij} x_{ij} \\
\text{subject to} \quad & \sum_{j:(s,j)\in A} x_{sj} = 1, \\
& \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = 0 \qquad (i \in N \setminus \{s,t\}), \\
& \sum_{i:(i,t)\in A} x_{it} = 1, \\
& x_{ij} \in \{0,1\} \qquad ((i,j) \in A), \\
& \sum_{(i,j)\in A} \frac{d_{ij}}{v_{\max} - r_{ij}} x_{ij} \le T.
\end{aligned}
\tag{4.11}
$$

where v_{ij}^* is the minimizer of $\tilde{\alpha}_{ij} v_{ij}^2 + \tilde{\beta}_{ij} v_{ij} + \tilde{\gamma}_{ij}$ in $v_{\min} \le v_{ij} \le v_{\max}$. It is obtained by replacing $\tilde{v}_{ij}$ in the objective function in (4.9) with v_{ij}^* and $\tilde{v}_{ij}$ in the last constraint of (4.9) with $v_{\max}$. Instead of (4.9), we use (4.11) to generate short s–t paths. By solving Problem (4.11), we obtain a lower bound for the optimal value θ^* to Problem (4.5) as well as a short s–t path. The reason is as follows: For any feasible solution (x_{ij}, v_{ij}) for $(i,j) \in A$ to Problem (4.5), x_{ij} for $(i,j) \in A$ is a feasible solution to (4.11) since the following relations hold:

$$
\sum_{(i,j)\in A} \frac{d_{ij}}{v_{\max} - r_{ij}} x_{ij} \le \sum_{(i,j)\in A} \frac{d_{ij}}{v_{ij} - r_{ij}} x_{ij} = \sum_{(i,j)\in A} \frac{d_{ij} x_{ij}}{v_{ij} - r_{ij} x_{ij}} \le T.
$$

In addition, the corresponding objective value in (4.11) is not larger than that in Problem (4.5) since v_{ij}^* is the minimizer of $\tilde{\alpha}_{ij} v_{ij}^2 + \tilde{\beta}_{ij} v_{ij} + \tilde{\gamma}_{ij}$.

Employing (4.8) and (4.11), we can construct an efficient algorithm for solving Problem (4.5) as described in Algorithm 4.1. There exist several ways to implement "delete P from the network." In the implementation in Tanaka and Kobayashi [84], we add Inequality (4.10) into the constraints.

An advantage of the route generation algorithm is that we can stop the enumeration of s–t paths in this algorithm. The reason is as follows: In each iteration, the optimal value θ_{R} of (4.11), θ_{S} of (4.8), θ^* of problem (4.5), and the best known objective value $\theta^\dagger$ satisfies $\theta_{\mathrm{R}} \le \theta_{\mathrm{S}}$ and $\theta^* \le \theta^\dagger$. Note that θ_{R} is monotonically increasing since (4.11) finds the shortest s–t path which is different from the ones found in the previous iterations. Suppose that $\theta^\dagger \le \theta_{\mathrm{R}}$ holds in a certain iteration. In such a case, $\theta^* \le \theta^\dagger \le \theta_{\mathrm{R}} \le \theta_{\mathrm{S}}$ holds. Since θ_{R} is monotonically increasing, this relation always holds in the following iterations. This means that any s–t path which will be found in the following iterations cannot be better than the best known solution. This guarantees the optimality

Algorithm 4.1 Route generation algorithm

$\theta^\dagger := +\infty$
while Problem (4.11) is feasible **do**
  Solve Problem (4.11) and let P and θ_R be the optimal s–t path corresponding to an optimal solution and the optimal value, respectively
  if $\theta^\dagger \leq \theta_R$ **then**
    break
  end if
  Solve (4.8) corresponding to P and let θ_S denote its optimal value
  if $\theta_S < \theta^\dagger$ **then**
    $\theta^\dagger := \theta_S$
  end if
  Delete P from the network
end while

of $\theta^\dagger$ so that we can stop the enumeration of s–t paths. Another advantage of this algorithm is that it generates a feasible solution at each iteration. Owing to these advantages, we can easily obtain an approximate optimal solution for the feasible problem and the certificate of the infeasibility for the infeasible one.

4.5 Numerical results

Tanaka and Kobayashi [84] have demonstrated the effectiveness of the route generation algorithm. Kobayashi and Tanaka [54] have also tested those of the perspective reformulation. In this section, summarize these numerical results.

Test instances for these experiments are generated as the follows: They considered a grid network illustrated in Figure 4.1. The network has m nodes in the vertical direction and n nodes in the horizontal direction. They generated instances for $m = 5, 10$ and $n = 50, 100, 150, 200$. The value of speed reduction r_{ij} is a randomly chosen integer from the set $\{1, 2, 3, 4\}$ for each arc (i, j). They changed T to make the instance feasible except the last one with each pair of (m, n). They used the same values of $v_{\min}$, $v_{\max}$, α, β, and γ in problem (4.5) as those in Fagerholt *et al.* [33].

Tanaka and Kobayashi [84] solved the instances of Problem (4.5) with the route generation algorithm and Gurobi Optimizer 5.1.0. Kobayashi and Tanaka [54] solved the corresponding instances of Problem (4.7) by Gurobi Optimizer 5.1.0. The route generation algorithm is implemented in Python 2.7.3, and (4.8) and (4.11) are solved with Gurobi Optimizer 5.1.0. In their experiments, they stopped the algorithms when the elapsed time exceeds ten minutes. The numerical experiments were executed on a computer with Intel Xeon 2.80 GHz 6-Core CPUs and 12 GB of RAM with Scientific Linux release 5.9.

In tables in this section, we use the following symbols and notation. Figures in "m" and "n" columns denote the number of nodes in the vertical and horizontal direction, respectively. A figure in "T" column denotes the value in the

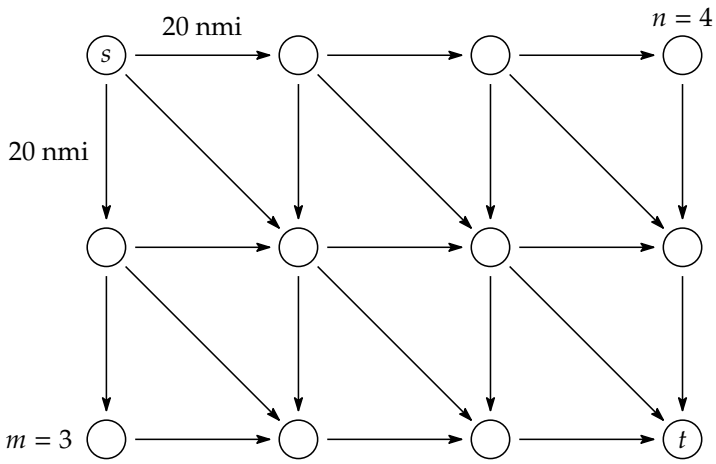

Fig. 4.1: Network structure (the symbol nmi means the nautical mile)

right-hand side of total transition time constraint. A figure in "time" column denotes the total computational time in seconds. A figure in "obj" column denotes the best known objective value. For the route generation algorithm, a figure in "mip gap" column denotes the relative difference between the lower bound θ_{R} and the best known objective value $\theta^{\dagger}$ at the last iteration defined by $\max\{\theta^{\dagger} - \theta_{\mathrm{R}}, 0\}/\theta^{\dagger}$. For Gurobi Optimizer, we used the value defined similarly. We use the notation "infeasible" to denote that the algorithm detected the infeasibility of the instance, the notation "optimal" to denote that it guaranteed the optimality of the best known solution, and the notation "no info" to denote that it could neither detect the infeasibility nor find any feasible solution. A figure in "node" column denotes the number of nodes in the search tree for Gurobi Optimizer.

Table 4.1–4.3 shows the numerical results for the instances of $m = 5$. The route generation algorithm and Gurobi Optimizer for Problem (4.7) found a feasible solution or detected its infeasibility for all instances. In contrast to this, Gurobi Optimizer for Problem (4.5) could neither find a feasible solution nor detect the infeasibility of four instances with $n = 150$ and four instances with $n = 200$. We note that the "obj" values by Gurobi Optimizer for (4.7) and the route generation algorithm for all instances. Gurobi Optimizer for (4.7) and the route generation algorithm obtained approximately-same "obj" values. Since Gurobi Optimizer for (4.7) obtained optimality except two instances, the route generation algorithm also found approximately optimal solutions. However, "obj" and "mip gap" values by Gurobi Optimizer for (4.5) is larger than those by the others. The difference of the results for (4.5) and (4.7) implies that Gurobi Optimizer generated tight lower bound for (4.7) but it could not generate it for (4.5). In fact, the number of explored nodes for (4.7) is smaller than that for (4.5).

The reason is that Gurobi Optimizer generated only loose lower bound for (4.5) formulation so that the bounding procedure did not work well. Concerning the computational time, the route generation algorithm found an optimal solution in short computational time for each instance with small or large T values. The reason is as follows: For large T value, *i.e.*, the case that the time constraint is loose, v_{ij} with small values can be feasible. Hence, the optimal value of Problem (4.11) is close to that of Problem (4.5). In other words, we can obtain a tight bound in such cases. For small T value, *i.e.*, the case that the time constraint is tight, there are a few feasible s–t paths. Thus, the route generation algorithm enumerate all of them and stopped.

Table 4.4–4.6 shows the results for the instances of $m = 10$. Even for (4.7) Gurobi Optimizer could neither find a feasible solution nor detect the infeasibility for some instances. For all instances the route generation algorithm obtained the smallest "obj" values. We can see that the route generation algorithm is effective for large instances.

Tab. 4.1: Numerical results for the instances of $m = 5$ and $n = 50, 100$

			Route generation algorithm			Gurobi for (4.5)				Gurobi for (4.7)			
m	n	T	time	obj	mip gap	time	obj	mip gap	node	time	obj	mip gap	node
5	50	90.0	0.3	201.3	optimal	600.0	201.7	82.4%	579927	3.2	201.3	optimal	0
		80.0	13.2	202.3	optimal	600.0	206.2	137.2%	82945	19.1	202.3	optimal	398
		70.0	600.8	223.1	8.9%	600.0	225.9	109.9%	89349	24.0	223.1	optimal	480
		60.0	601.5	292.4	30.5%	600.0	298.2	75.7%	77288	25.1	292.3	optimal	679
		57.1	601.3	328.5	38.1%	181.0	328.5	optimal	7758	16.9	328.5	optimal	528
		57.0	221.3	330.0	optimal	151.0	330.0	optimal	4908	18.9	330.0	optimal	508
		56.9	35.3	331.5	optimal	20.9	331.5	optimal	2586	13.8	331.5	optimal	599
		56.8	3.9	333.0	optimal	67.7	333.0	optimal	2012	15.6	333.0	optimal	396
		56.7	0.3	334.5	optimal	72.0	334.5	optimal	1890	15.2	334.5	optimal	457
		56.6	0.2	—	infeasible	14.3	—	infeasible	1643	13.3	—	infeasible	356
5	100	170.0	0.6	405.7	optimal	600.0	406.3	170.0%	42554	40.2	405.7	optimal	179
		160.0	602.2	408.4	0.2%	600.0	411.0	193.0%	26228	103.8	408.4	optimal	875
		150.0	601.9	421.9	3.4%	600.0	429.7	202.5%	20670	106.3	421.9	optimal	821
		140.0	602.1	451.5	9.7%	600.1	458.3	185.1%	22997	103.1	451.4	optimal	1066
		130.0	601.5	504.2	19.2%	600.1	524.2	28.0%	5196	102.2	504.2	optimal	1613
		120.0	601.4	591.1	31.1%	600.2	601.4	137.5%	16910	91.2	591.1	optimal	1833
		114.1	602.5	665.5	38.8%	449.2	665.5	optimal	15560	64.6	665.5	optimal	1067
		114.0	259.1	667.0	optimal	399.8	667.0	optimal	8107	58.8	667.0	optimal	1199
		113.9	34.2	668.5	optimal	544.5	668.5	optimal	5724	52.0	668.5	optimal	1044
		113.8	3.1	670.1	optimal	377.8	670.1	optimal	3769	58.3	670.1	optimal	1046
		113.7	0.7	671.7	optimal	315.1	671.7	optimal	3087	67.1	671.7	optimal	719
		113.6	0.2	—	infeasible	344.4	—	infeasible	2496	56.6	—	infeasible	772

Tab. 4.2: Numerical results for the instances of $m = 5$ and $n = 150$

			Route generation algorithm			Gurobi for (4.5)				Gurobi for (4.7)			
m	n	T	time	obj	mip gap	time	obj	mip gap	node	time	obj	mip gap	node
5	150	250.0	1.1	609.1	optimal	600.0	616.9	274.3%	15735	272.2	609.1	optimal	1685
		240.0	602.5	613.1	0.4%	600.0	621.4	278.1%	14459	357.0	613.0	optimal	2268
		230.0	602.2	624.3	2.2%	600.1	633.3	248.4%	14114	234.0	624.3	optimal	1813
		220.0	600.4	644.9	5.3%	600.0	654.5	225.9%	13848	292.1	644.9	optimal	1785
		210.0	601.0	677.5	9.9%	600.0	690.3	223.0%	13879	411.4	677.4	optimal	2027
		200.0	602.8	725.3	15.8%	600.1	747.8	224.8%	12668	216.8	725.2	optimal	1600
		190.0	601.6	792.8	23.0%	600.0	810.9	207.0%	12168	200.0	792.8	optimal	2237
		180.0	601.1	886.0	31.1%	600.0	908.1	153.4%	11364	392.8	886.0	optimal	2365
		170.8	604.2	1002.5	39.1%	600.0	—	no info	5621	145.5	1002.5	optimal	1803
		170.7	87.6	1004.1	optimal	600.0	—	no info	6690	154.2	1004.0	optimal	2303
		170.6	6.5	1005.6	optimal	600.0	—	no info	3331	159.1	1005.6	optimal	1933
		170.5	1.1	1007.2	optimal	600.0	—	no info	2602	133.0	1007.2	optimal	1546
		170.4	0.6	—	infeasible	554.5	—	infeasible	9039	134.1	—	infeasible	1875

Tab. 4.3: Numerical results for the instances of $m = 5$ and $n = 200$

			Route generation algorithm			Gurobi for (4.5)				Gurobi for (4.7)			
m	n	T	time	obj	mip gap	time	obj	mip gap	node	time	obj	mip gap	node
5	200	340.0	1.2	810.1	optimal	600.0	821.6	344.2%	19236	35.5	810.1	optimal	0
		330.0	4.6	810.6	optimal	600.0	824.3	394.7%	11454	444.3	810.6	optimal	1717
		320.0	602.2	815.1	0.4%	600.1	837.1	173.1%	5605	600.3	816.6	0.3%	2368
		310.0	600.9	825.0	1.6%	600.1	851.5	211.0%	3490	368.8	825.0	optimal	1881
		300.0	602.4	841.4	3.6%	600.1	854.7	166.2%	4476	315.3	841.4	optimal	1417
		290.0	600.9	865.7	6.3%	600.1	880.1	186.1%	3800	403.7	865.7	optimal	1319
		280.0	600.1	899.4	9.8%	600.1	921.1	237.0%	3472	356.5	899.4	optimal	2647
		270.0	602.3	944.5	14.1%	600.0	976.6	248.4%	3764	253.7	944.5	optimal	1488
		260.0	601.8	1003.4	19.1%	600.0	1022.0	396.7%	8598	294.7	1003.3	optimal	1850
		250.0	602.1	1078.9	24.8%	600.1	1115.6	160.4%	3483	416.5	1078.9	optimal	3051
		240.0	600.5	1174.9	30.9%	600.1	1203.6	144.5%	6011	293.5	1174.9	optimal	2292
		230.0	603.0	1296.0	37.4%	600.0	1329.1	252.9%	7548	600.4	1296.0	0.3%	3502
		227.2	600.8	1336.5	39.3%	600.7	—	no info	1864	383.4	1336.5	optimal	2044
		227.1	97.2	1338.1	optimal	603.4	—	no info	1869	376.1	1338.1	optimal	2033
		227.0	7.1	1339.7	optimal	600.1	—	no info	2090	238.8	1339.7	optimal	1693
		226.9	0.9	—	infeasible	600.0	—	no info	1999	212.7	—	infeasible	1755

Tab. 4.4: Numerical results for the instances of $m = 10$ and $n = 50, 100$

			Route generation algorithm			Gurobi for (4.5)				Gurobi for (4.7)			
m	n	T	time	obj	mip gap	time	obj	mip gap	node	time	obj	mip gap	node
10	50	90.0	0.6	206.2	optimal	600.0	218.3	574.0%	21894	31.8	206.2	optimal	521
		80.0	600.8	209.2	0.9%	600.0	232.6	524.1%	16931	76.7	209.2	optimal	1323
		70.0	601.7	237.3	12.6%	600.0	264.0	496.1%	10942	67.4	237.3	optimal	861
		60.0	600.7	321.0	35.4%	600.0	345.6	383.8%	3053	58.7	321.0	optimal	450
		58.6	601.8	340.4	39.1%	600.0	346.1	176.3%	4118	53.4	340.4	optimal	595
		58.5	175.9	342.0	optimal	549.9	342.0	optimal	8614	53.2	342.0	optimal	683
		58.4	3.8	343.5	optimal	389.1	343.5	optimal	2575	60.8	343.5	optimal	1191
		58.3	0.3	—	infeasible	443.0	—	infeasible	2145	60.4	—	infeasible	695
10	100	170.0	1.2	407.0	optimal	600.0	429.4	609.4%	9236	449.4	407.0	optimal	2240
		160.0	601.4	410.4	0.6%	600.1	440.0	612.0%	4569	404.1	410.4	optimal	1980
		150.0	601.8	425.5	4.1%	600.1	465.3	589.2%	5830	384.3	425.5	optimal	1574
		140.0	600.3	457.5	10.8%	600.0	515.5	541.6%	3253	600.1	460.4	2.0%	2030
		130.0	601.9	514.0	20.6%	600.0	581.1	490.9%	3426	385.2	514.1	optimal	1896
		120.0	600.3	606.6	32.7%	600.1	695.4	428.2%	2101	600.1	627.3	18.8%	2150
		114.9	602.1	673.8	39.5%	600.0	—	no info	2069	600.0	—	no info	2435
		114.8	90.2	675.3	optimal	600.0	—	no info	1926	600.7	—	no info	2262
		114.7	4.0	676.9	optimal	600.0	—	no info	2012	600.0	—	no info	2111
		114.6	1.1	—	infeasible	600.0	—	no info	1885	600.1	—	no info	2467

Tab. 4.5: Numerical results for the instances of $m = 10$ and $n = 150$

			Route generation algorithm			Gurobi for (4.5)				Gurobi for (4.7)			
m	n	T	time	obj	mip gap	time	obj	mip gap	node	time	obj	mip gap	node
10	150	250.0	4.6	604.7	optimal	600.1	635.8	642.6%	1994	600.1	626.0	21.2%	950
		240.0	600.4	608.0	0.4%	600.1	642.4	635.8%	1998	600.1	632.5	22.2%	1077
		230.0	600.6	618.9	2.2%	600.1	668.0	620.9%	2208	600.1	653.5	23.1%	1907
		220.0	601.4	639.2	5.3%	600.0	682.6	602.4%	2361	600.2	662.3	38.0%	1995
		210.0	601.8	671.7	9.9%	600.1	729.9	570.7%	1937	600.1	705.5	32.4%	1578
		200.0	601.6	719.9	15.9%	600.1	784.0	536.0%	1980	600.1	752.9	24.3%	2451
		190.0	600.7	788.1	23.2%	600.1	893.4	478.0%	1953	600.1	855.2	38.8%	1370
		180.0	601.3	882.7	31.4%	600.1	982.2	456.7%	1598	600.1	942.2	33.9%	2335
		170.6	600.5	1003.8	39.7%	600.0	—	no info	1874	601.4	—	no info	1917
		170.5	139.8	1005.3	optimal	600.0	—	no info	1966	600.1	—	no info	2352
		170.4	1.7	—	infeasible	600.0	—	no info	1925	600.7	—	no info	2270

Tab. 4.6: Numerical results for the instances of $m = 10$ and $n = 200$

			Route generation algorithm			Gurobi for (4.5)				Gurobi for (4.7)			
m	n	T	time	obj	mip gap	time	obj	mip gap	node	time	obj	mip gap	node
10	200	340.0	2.7	811.2	optimal	600.1	860.0	629.2%	2680	600.0	830.9	6.9%	281
		330.0	351.4	811.9	optimal	600.1	860.2	609.8%	3519	600.1	856.0	27.1%	250
		320.0	601.0	816.9	0.6%	600.1	871.2	608.4%	2173	600.0	847.9	33.0%	749
		310.0	600.2	827.4	1.9%	600.1	888.3	571.4%	2589	600.1	852.6	33.1%	888
		300.0	600.2	844.6	3.9%	600.1	909.7	581.1%	2056	600.1	874.3	41.9%	1181
		290.0	600.2	870.0	6.7%	600.1	947.9	571.7%	2064	600.8	901.7	29.5%	1046
		280.0	601.2	905.0	10.3%	600.1	986.8	551.9%	2029	600.1	930.8	45.3%	1321
		270.0	601.4	951.7	14.7%	600.1	1033.0	525.0%	2273	600.1	993.0	42.0%	1024
		260.0	600.0	1012.5	19.8%	600.1	1136.3	489.6%	2130	600.1	1067.1	46.6%	1209
		250.0	600.6	1090.6	25.5%	600.2	1189.9	490.1%	1749	600.3	1153.6	47.3%	791
		240.0	601.4	1189.6	31.7%	600.1	1285.8	449.1%	2067	600.1	1250.1	45.2%	1175
		230.0	602.0	1314.6	38.2%	600.1	—	no info	1776	600.1	—	no info	1434
		228.0	601.4	1344.4	39.6%	600.1	—	no info	1528	600.0	—	no info	2041
		227.9	10.3	1346.0	optimal	600.0	—	no info	1525	600.2	—	no info	1802
		227.8	2.3	—	infeasible	600.0	—	no info	1503	600.1	—	no info	1482

4.6 Concluding remarks

In this chapter we have reviewed the MISOCP formulation for the ship navigation problem and the two approaches to solve it.

One is the perspective reformulation proposed by Kobayashi and Tanaka [54]. This formulation provides tight bound for the MISOCP. The numerical results have demonstrated that this approach is effective for small instances.

The other is the route generation algorithm proposed by Tanaka and Kobayashi [84]. This algorithm generates short shipping routes and optimizes the shipping speed on each leg for each shipping route. By employing the lower bound for the optimal value, they have made it possible to stop the enumeration of shipping routes and guarantee the optimality of the obtained solution or detect the infeasibility of the problem. The numerical results have shown the following features of the algorithm: If the total transition time constraint is either loose or tight, the algorithm obtain an optimal solution in short computational time. Otherwise, the algorithm returns a good feasible solution although it might not guarantee its optimality.

There is future work described below: In their papers, we approximate the objective function by the quadratic function. Therefore, the solution to the MISOCP model is an approximate solution to the exact MINLP formulation. Solution methods for an exact optimal solution to the MINLP formulation may be needed. It also should be compared with an optimal solution to the approximate model. Moreover, the route generation algorithm sometimes cannot guarantee the optimality of the obtained solution. It is needed to develop ways to obtain tighter lower bounds.

Chapter 5

Summary and prospects

In this thesis we have seen several modeling techniques and algorithm on conic optimization problem. We know symmetric cone optimization problems can be solved in polynomial time with interior-point algorithms. Hence, if we can formulate a problem as a symmetric cone optimization problem with polynomial size within polynomial time, we can solve it within polynomial time. However, we should ask ourselves the following questions at the time: Could we really solve the problem? Was optimization is really of benefit of the real-world? May we finish the work at the time? —The answers are No, No, and No!

There are numerous optimization problems in our real-world. Some of them can be modeled as (mixed-integer) conic optimization problems. However, we usually cannot solve them with naive approaches even if they are theoretically solvable within polynomial time. The reason is that a real-world problem is often too large for even polynomial time algorithms to solve it within resonable computational time and is often degenerate. In such situations, the exploitation of structures of the problem is a powerful approach. By exploiting them, we may be able to reformulate, reduce, or decompose it. The resulting problems may be solvable with simple calculations, existing approaches, or specialized algorithms.

In Chapter 2 we have discussed approaches to doubly nonnegative optimization proposed by Tanaka *et al.* [87, 88]. Although this problem can be reformulated as a semidefinite optimization problem, even great semidefinite optimization software cannot solve large instances of doubly nonnegative optimization problem. The reason is that the reformulated SDP is often degenerate and large. Tanaka *et al.* [87, 88] have overcame these difficulties by exploiting structure of the problem. They proposed the following ways to remove degeneracy: They found degeneracy in a doubly nonnegative optimization problem arising from a mixed-binary quadratic optimization problem and proposed a reduction method by employing the degeneracy in [87]. They also showed a numerical way to find degeneracy in a general doubly nonnegative optimization problem in [88]. Since the reduced problem is less degenerate, an interior-point method stably works. Tanaka *et al.* [87] also proposed an inexact primal-dual path-following algorithm. In each iteration of the algorithm, we solve a linear system with a Krylov subspace method to compute a search di-

rection. They also proposed preconditioners for the linear system to accelerate the convergence of the iterative algorithm by exploiting the structure of the coefficient matrix. The inexact interior-point method solves large instances of doubly nonnegative optimization problem.

In Chapter 3 we have considered well-conditioned matrix approximation problems. Although each of these problems can be also formulated as a symmetric cone optimization problem, the size of the resulting problem often beat existing solvers again. To reduce the problem, Tanaka and Nakata [85] introduced the Ky Fan *p-k* norm. By utilizing the property of the norm, they reformulated a basic problem to a univariate convex optimization problem. The resulting univariate problem is easily solved with a binary search. Tanaka and Nakata [86] proposed the successive projection method for problems with additional constraints. In their algorithm, we employ the binary search for the basic problem as a subroutine. Their algorithm is much faster than a general-purpose solver.

In Chapter 4 we have dealt with an optimization on maritime affair named a ship navigation problem. Tanaka and Kobayashi [84] modeled this problem as a mixed-integer second-order cone optimization problem. Although the problem can be handled with a general-purpose solver, it is still formidable. The reason is that an algorithm implemented in the solver is based on the branch-and-bound procedure and it is still developing. Kobayashi and Tanaka [54] proposed a perspective reformulation of the problem. Since it provide tighter bound for the optimal value, the reformulation accelerate the branch-and-bound-like algorithm. However, the algorithm is not a viable choice for large instances. Tanaka and Kobayashi [84] proposed a route generation algorithm. This algorithm successively optimize subproblems. Since the problem contains a network structure, the subproblems are smaller than the original problem. The route generation algorithm works efficiently even for large instances although it sometimes could not guarantee the optimality within reasonable computational time.

The results of these researches have the potential to promote algorithms for other optimization problems. We have the following future directions.

The results on doubly nonnegative optimization might provide us an efficient algorithm for NP-hard optimization problem. As we have seen in Chapter 2, doubly nonnegative relaxation provides a tight lower bound for mixed-binary quadratic optimization problem that includes many important NP-hard optimization problems arising in science and engineering. When we try to solve such problems, we often employ the branch-and-bound algorithm and the cutting plane method. If we can effectively employ doubly nonnegative relaxation in such algorithms, we might be able to solve such difficult problems within reasonable computational time. For this purpose, we must develop an algorithm that can more acculately solve larger instances within shorter computational time.

The algorithm for the well-conditioned matrix approximation problems can be extended to another well-conditioned matrix approximation problem. The results for the basic problem may seem limited. As we have seen in Chapter 3,

however, we could construct an efficient algorithm for the problem with additional constraints by employing the basic problem as a subproblem. By using similar way, we can construct an algorithm for another well-conditioned matrix approximation problem. As we saw, such problem arises from finance, signal processing, and statistics because of importance of well-conditioned matrices. Since constraints that we must take account can vary, it is necessary to derive an different algorithm for each problem.

The route generation algorithm for the ship navigation problem is easy to extend. In real ship navigation, we might take account other constraints that we have not considered in the ship navigation problem in Chapter 4. When such constraints are linear with respect to binary variables, we can employ the algorithm by adding the constraint to the constraint shortest path problem that is employed as a subproblem in the algorithm. Similarly, when such constraints can be written as second-order cone constraints with respect to continuous variables, we can do it by adding the constraints to the second-order cone optimization problem in the algorithm. More generally, we might be able to construct an algorithm for an optimization problem with special structure such as nonlinear optimization problem on a network.

As we have seen above, ways to exploit structures of problems differ depending on each problem. Here is our slogan: There is no royal road to *optimization.* However, it may be a time to assemble numerous ways. If an algorithm detected and utilized useful structures in a given problem *automatically*, we should be able to solve it efficiently. In other words, if we could employ software equipped with such an algorithm, we should be able to solve numerous real-world problem in various fields. To implement such software, we need to enumerate useful structures and ways to exploit them and develop methods to detect and utilize the structures. Such techniques have already been implemented in some of commercial solvers on integer optimization. Can we do it on conic optimization? The author hopes that the results in this thesis will facilitate the development of optimization and the betterment of society.

Bibliography

[1] F. Alizadeh: Interior point methods in semidefinite programming with applications to combinatorial optimization, *SIAM Journal on Optimization* **5** (1995) 13–51.

[2] F. Alizadeh and D. Goldfarb: Second-order cone programming, *Mathematical Programming* **95** (2003) 3–51.

[3] A. Aubry, A. De Maio, L. Pallotta, and A. Farina: Maximum likelihood estimation of a structured covariance matrix with a condition number constraint, *IEEE Transactions on Signal Processing* **60** (2012) 3004–3021.

[4] A. Azaron and F. Kianfar: Dynamic shortest path in stochastic dynamic networks: Ship routing problem, *European Journal of Operational Research* **144** (2003) 138–156.

[5] A. Ben-Tal and A. Nemirovski: *Lectures on Modern Convex Optimization,* Society for Industrial and Applied Mathematics, Philadelphia, 2001.

[6] A. Berman and N. Shaked-Monderer: *Completely Positive Matrices,* World Scientific, Singapore, 2003.

[7] I. M. Bomze: On standard quadratic optimization problems, *Journal of Global Optimization* **13** (1998) 369–387.

[8] I. M. Bomze and F. Jarre: A note on Burer's copositive representation of mixed-binary QPs, *Optimization Letters* **4** (2010) 465–472.

[9] I. M. Bomze, W. Schachinger, and G. Uchida: Think co(mpletely)positive! Matrix properties, examples and a clustered bibliography on copositive optimization, *Journal of Global Optimization* **52** (2012) 423–445.

[10] B. Borchers and J. G. Young: Implementation of a primal-dual method for SDP on a shared memory parallel architecture, *Computational Optimization and Applications* **37** (2007) 355–369.

[11] J. M. Borwein and H. Wolkowicz: Facial reduction for a cone-convex programming problem, *Journal of the Australian Mathematical Society* **30** (1981) 369–380.

[12] J. M. Borwein and H. Wolkowicz: Regularizing the abstract convex program, *Journal of Mathematical Analysis and Applications* **83** (1981) 495–530.

[13] S. Boyd and L. Vandenberghe: *Convex Optimization,* Cambridge University Press, Cambridge, 2004.

[14] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi: A tutorial on geometric programming, *Optimization and Engineering* **8** (2007) 67–127.

[15] J. P. Boyle and R. L. Dykstra: A method for finding projections onto the intersection of convex sets in Hilbert spaces, in *Advances in Order Restricted Statistical Inference,* R. Dykstra, T. Robertson, and F. T. Wright (eds.), Springer, New York, 1985, pp. 28–47.

[16] S. Burer: On the copositive representation of binary and continuous nonconvex quadratic programs, *Mathematical Programming* **120** (2009) 479–495.

[17] S. Burer: Optimizing a polyhedral-semidefinite relaxation of completely positive programs, *Mathematical Programming Computation* **2** (2010) 1–19.

[18] S. Burer and R. D. C. Monteiro: A nonlinear programming algorithm for solving semidefinite program via low-rank factorization, *Mathematical Programming* **95** (2003) 329–357.

[19] R. E. Burkard, E. Çela, S. E. Karisch, and F. Rendl: QAPLIB—A quadratic assignment problem library, *Journal of Global Optimization* **10** (1997) 391–403.

[20] M. Chistiansen, K. Fagerholt, B. Nygreen, and D. Ronen: Ship routing and schedulling in the millennium, *European Journal of Operational Research* **228** (2013) 467–483.

[21] G. Cornuejols and R. Tütüncü: *Optimization Methods in Finance,* Cambridge University Press, Cambridge, 2007.

[22] M. J. Daniels and R. E. Kass: Shrinkage estimators for covariance matrices, *Biometrics* **57** (2001) 1173–1184.

[23] G. B. Dantzig: *Linear Programming and Extensions,* Princeton University Press, Princeton, 1963.

[24] A. d'Aspremont, O. Banerjee, and L. El Ghaoui: First-order methods for sparse covariance selection, *SIAM Journal on Matrix Analysis and Applications* **30** (2008) 56–66.

[25] T. Davi and F. Jarre: On the stable solution of large scale problems over the doubly nonnegative cone, *Mathematical Programming,* 2013, Online First (DOI: `10.1007/s10107-013-0687-3`).

[26] E. de Klerk: *Aspects of Semidefinite Programming,* Kluwer Academic Publishers, Dordrecht, 2002.

[27] A. P. Dempster: Covariance selection, *Biometrics* **28** (1972) 157–175.

[28] F. Deutsch and H. Hundal: The rate of convergence for the method of alternating projections II, *Journal of Mathematical Analysis and Applications* **205** (1997) 381–405.

[29] D. K. Dey and C. Srinivasan: Estimation of a covariance matrix under Stein's loss, *Annals of Statistics* **13** (1985) 1581–1591.

[30] P. J. C. Dickinson: Geometry of the copositive and completely positive cones, *Journal of Mathematical Analysis and Applications* **380** (2011) 377–395.

[31] P. J. C. Dickinson and L. Gijben: On the computational complexity of membership problems for the completely positive cone and its dual, *Computational Optimization and Applications,* 2013, Online First (DOI: `10.1007/s10589-013-9594-z`).

[32] R. L. Dykstra: An algorithm for restricted least squares regressions, *Journal of the American Statistical Association* **78** (1983) 837–842.

[33] K. Fagerholt, G. Laporte, and I. Norstad: Reducing fuel emissions by optimizing speed on shipping routes, *Journal of the Operational Research Society* **61** (2010) 523–529.

[34] J. Faraut and A. Korányi: *Analysis on Symmetric Cones,* Oxford Science Publications, Oxford, 1994.

[35] A. Frangioni and C. Gentile: Perspective cuts for a class of convex 0-1 mixed integer programs, *Mathematical Programming* **106** (2006) 225–236.

[36] R. W. Freund and N. M. Nachtigal: A new Krylov-subspace method for symmetric indefinite linear systems, in *Proceedings of the 14th IMACS World Congress on Computational and Applied Mathematics,* 1994, pp. 1253–1256.

[37] J. Friedman, T. Hastie, and R. Tibshirani: Sparse inverse covariance estimation with the graphical lasso, *Biostatistics* **9** (2008) 432–441.

[38] D. Ge and Y. Ye: On doubly positive semidefinite programming relaxations, *Technical Report, Department of Management Science and Engineering, Stanford University,* 2010.

[39] M. X. Goemans and D. P. Williamson: Improved approximation algorithms for maximum cut and satisfybility problems using semidefinite programming, *Journal of the ACM* **42** (1995) 1115–1145.

[40] O. Günlük and J. Linderoth: Perspective reformulations of mixed integer nonlinear programs with indicator variables, *Mathematical Programming* **124** (2010) 183–205.

[41] L. Haff: The variational form of certain Bayes estimators, *Annals of Statistics* **19** (1991) 1163–1190.

[42] S.-P. Han: A successive projection method, *Mathematical Programming* **40** (1988) 1–14.

[43] C. Helmberg, F. Rendl, R. J. Vanderbei, and H. Wolkowicz: An interior-point method for semidefinite programming, *SIAM Journal on Optimization* **6** (1996) 342–361.

[44] N. J. Higham: Computing the nearest correlation matrix—a problem from finance, *IMA Journal of Numerical Analysis* **22** (2002) 392–343.

[45] S. Homer and M. Peinado: Design and performance of parallel and distributed approximation algorithms for maxcut, *Journal of Parallel and Distributed Computing* **46** (1997) 48–61.

[46] R. A. Horn and C. R. Johnson: *Matrix Analysis (Second Edition)*, Cambridge University Press, Cambridge, 2012.

[47] R. H. Horn and C. R. Johnson: *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, 1994.

[48] R. W. James: *Application of Wave Forecasts to Marine Navigation*, U.S. Naval Oceanographic Office, Washington, D.C., 1957.

[49] W. James and C. Stein: Estimation with quadratic loss, in *Proceedings of Fourth Berkeley Symposium on Mathematical Statistics and Probability*, 1961, pp. 361–379.

[50] N. Karmarkar: A new polynomial-time algorithm for linear programming, *Combinatorica* **4** (1984) 373–395.

[51] L. G. Khachiyan: A polynomial algorithm in linear programming, *Doklady Akademiia Nauk SSSR* **244** (1979) 1093–1096.

[52] V. Klee and G. J. Minty: How good is the simplex algorithm?, In *Inequalities III*, O. Shisha (ed.), Academic Press, New York, 1972, pp. 159–175.

[53] K. Kobayashi, K. Nakata, and M. Kojima: A conversion of an SDP having free variables into the standard form SDP, *Computational Optimization and Applications* **36** (2007) 289–307.

[54] K. Kobayashi and M. Tanaka: Perspective reformulation for ship navigation problem, *Technical Report, Department of Industrial Engineering and Management, Tokyo Institute of Technology*, 2013.

[55] M. Kŏcvara and M. Stingl: PENNON—a code for convex nonlinear and semidefinite programming, *Optimization Methods and Software* **8** (2003) 317–333.

[56] M. Kojima, S. Mizuno, and A. Yoshise: A primal-dual interior-point algorithm for linear programming, in *Progress in Mathematical Programming, Interior-Point and Related Methods,* N. Megiddo (eds.), Springer-Verlag, New York, 1989, 29–47.

[57] M. Kojima, M. Shida, and S. Sindoh: Search directions in the SDP and monotone SDLCP: generalization and inexact computation, *Mathematical Programming* **85** (1999) 51–80.

[58] M. Kojima and L. Tunçel: Cones of matrices and successive convex relaxations of nonconvex sets, *SIAM Journal on Optimization* **10** (2000) 750–778.

[59] A. N. Langville and W. J. Stewart: A Kronecker product approximate preconditioner for SANs, *Numerical Linear Algebra with Applications* **11** (2004) 723–752.

[60] O. Leodit and M. Wolf: A well-conditioned estimator for large-dimensional covariance matrices, *Journal of Multivariate Analysis* **88** (2004) 365–411.

[61] H. Lo and M. McCord: Adaptive ship routing through stochastic ocean current: general formulations and empirical results, *Transportation Research Part A* **32** (2003) 138–156.

[62] J. Löfberg: YALMIP: A toolbox for modeling and optimization in MATLAB, in *Proceedings of the CACSD Conference,* 2004, pp. 284–289.

[63] L. Lovaśz and A. Schrijver: Cones of matrices and set-functions and 0–1 optimization, *SIAM Journal on Optimization* **1** 166–190.

[64] D. G. Luenberger and Y. Ye: *Linear and Nonlinear Programming (Third Edition),* Springer, New York, 2010.

[65] H. D. Mittelmann: An independent benchmarking of SDP and SOCP solvers, *Mathematical Programming* **95** (2003) 407–430.

[66] R. D. C. Monteiro: Primal-dual path-following algorithms for semidefinite programming, *SIAM Journal on Optimization* **7** (1997) 663-678.

[67] R. D. C. Monteiro and Y. Zhang: A unified analysis for a class of path-following primal-dual interior-point algorithms for semidefinite programming, *Mathematical Programming* **81** (1998) 281–299.

[68] A. Nemirovskii and K. Scheinberg: Extension of Karmarkar's algorithm onto convex quadratically constrained quadratic problems, *Mathematical Programming* **72** (1996) 273–289.

[69] Y. E. Nesterov: Towards nonsymmetric conic optimization, *Optimization methods and Software* **27** (2012) 893–917.

[70] Y. E. Nesterov and A. Nemirovskii: *Interior Point Polynomial Methods for Convex Programming: Theory and Applications,* Society for Industrial and Applied Mathematics, Philadelphia, 1994.

[71] G. Pataki: A simple derivation of a facial reduction algorithm and extended dual systems, *Technical Report, Department of Statistics and Operations Research, University of North Carolina at Chapel Hill,* 2000.

[72] C. C. Paige and M. A. Saunders: Solution of sparse indefinite systems of linear equations, *SIAM Journal on Numerical Analysis* **12** (1975) 617–629.

[73] S. Poljak, F. Rendl, and H. Wolkowicz: A recipe for semidefinite relaxation for $(0,1)$-quadratic programming, *Journal of Global Optimization* **7** (1995) 51–73.

[74] J. Povh and F. Rendl: Copositive and semidefinite relaxation of the quadratic assignment problem, *Discrete Optimization* **6** (2009) 231–241.

[75] J. Peng and J. Wei: Approximating k-means-type clustering via semidefinite programming, *SIAM Journal on Optimization* **18** (2007) 186–205.

[76] A. N. Perakis and N. A. Papadakis: Fleet deployment optimization models. Part 1, *Maritime Policy & Management* **14** (1987) 127–144.

[77] A. N. Perakis and N. A. Papadakis: Fleet deployment optimization models. Part 2, *Maritime Policy & Management* **14** (1987) 145–155.

[78] H. Qi and D. Sun: A quadratically convergent newton method for computing the nearest correlation matrix, *SIAM Journal on Matrix Analysis and Applications* **28** (2006) 360–385.

[79] M. V. Ramana: An exact duality theory for semidefinite programming and its complexity implications, *Mathematical Programming* **77** (1997) 129–162.

[80] M. V. Ramana, L. Tunçel, and H. Wolkowicz: Strong duality for semidefinite programming, *SIAM Journal on Optimization* **7** (1997) 641–662.

[81] D. Ronen: The effect of oil price on the optimal speed of ships, *Journal of the Operational Research Society* **33** (1982) 1035–1040.

[82] A. Skajaa and Y. Ye: A homogeneous interior-point algorithm for nonsymmetric convex conic optimization, *Technical Report, Department of Management Science and Engineering, Stanford University,* 2012.

[83] J. F. Sturm: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones, *Optimization Methods and Software* **11 & 12** (1999) 625–653.

[84] M. Tanaka and K. Kobayashi: MISOCP formulation and route generation algorithm for ship navigation problem, *Technical Report, Department of Industrial Engineering and Management, Tokyo Institute of Technology,* 2013.

[85] M. Tanaka and K. Nakata: Positive definite matrix approximation with condition number constraint, *Optimization Letters,* 2013, Online First (DOI: 10.1007/s11590-013-0632-7).

[86] M. Tanaka and K. Nakata: Successive projection method for well-conditioned matrix approximation problems, *IEEE Signal Processing Letters* **21** (2014) 699–724.

[87] M. Tanaka, K. Nakata, and H. Waki: Application of a facial reduction algorithm and an inexact primal-dual path-following method for doubly nonnegative relaxation for mixed binary nonconvex quadratic optimization problems, *Pacific Journal of Optimization* **8** (2012) 418–422.

[88] M. Tanaka, K. Nakata, and H. Waki: Numerical reduction method for doubly nonnegative optimization problems, *Journal of Math-for-Industry* **5A** (2013) 41–50.

[89] M. J. Todd, K. C. Toh, and R. H. Tütüntü: On the Nesterov-Todd direction in semidefinite programming, *SIAM Journal on Optimization* **8** (1998) 769–796.

[90] K. C. Toh: An inexact primal-dual path following algorithm for convex quadratic SDP, *Mathematical Programming* **112** (2008) 221–254.

[91] K. C. Toh, M. J. Todd, and R. H. Tütüncü: On the implementation and usage of SDPT3—a MATLAB software package for semidefinite-quadratic-linear programming, version 4.0, in *Handbook of Semidefinite, Conic and Polynomial Optimization,* M. Anjos and J. B. Lasserre (eds.), Springer, New York, 2012, pp. 715–754.

[92] K. C. Toh, R. H. Tütüncü, and M. J. Todd: Inexact primal-dual path-following algorithms for a special class of convex quadratic SDP and related problems, *Pacific Journal of Optimization* **3** (2007) 135–164.

[93] T. Tsuchiya: A polynomial primal-dual path-following algorithm for second-order cone programming, *Research Memorandum, The Institute of Statistical Mathematics,* 1997.

[94] T. Tsuchiya: A convergence analysis of the scaling-invariant primal-dual path-following algorithms for second-order cone programming, *Optimization Methods and Software* **11** (1999) 141–182.

[95] L. Tunçel: On the Slater condition for the SDP relaxations of nonconvex sets, *Operations Research Letters* **29** (2001) 181–186.

[96] L. Tunçel and H. Wolkowicz: Strong duality and minimal representations for cone optimization, *Computational Optimization and Applications* **53** (2012) 619–648.

[97] H. Waki and M. Muramatsu: Facial reduction algorithm for conic optimization problems, *Journal of Optimization Theory and Applications* **158** (2013) 188–215.

[98] H. Waki, M. Nakata, and M. Muramatsu: Strange behaviors of interior-point methods for solving semidefinite programming problems in polynomial optimization, *Computational Optimization and Applications* **53** (2012) 823–844.

[99] J. H. Won and S.-J. Kim: Maximum likelihood covariance estimation with a condition number constraint, in *Proceedings of Fortieth Asilomar Conference on Signals, Systems, and Computers,* 2006, pp. 1445–1449.

[100] S. J. Wright: *Primal-Dual Interior-Point Methods,* Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[101] M. Yamashita, K. Fujisawa, M. Fukuda, K. Kobayashi, K. Nakata, and M. Nakata: Latest developments in the SDPA Family for solving large-scale SDPs, in *Handbook of Semidefinite, Conic and Polynomial Optimization,* M. Anjos and J. B. Lasserre (eds.), Springer, New York, 2012, pp. 687–714.

[102] M. Yamashita, K. Fujisawa, K. Nakata, M. Nakata, M. Fukuda, K. Kobayashi, and K. Goto: A high-performance software package for semidefinite programs: SDPA 7, *Research Report, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology,* 2010.

[103] A. Yoshise and Y. Matsukawa: On optimization over the doubly nonnegative cone, in *Proceedings of 2010 IEEE Multi-conference on Systems and Control,* 2010, pp. 13–19.

[104] Q. Zhao, S. E. Karisch, F. Rendl, and H. Wolkowicz: Semidefinite programming relaxations for the quadratic assignment problem, *Journal of Combinatorial Optimization* **2** (1998) 71–109.