論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Study on Cutting-plane Algorithms for Mixed-integer Semidefinite Optimization
著者(和文)	小林健
Author(English)	Ken Kobayashi
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11762号, 授与年月日:2022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,松井 知己,塩浦 昭義,梅室 博行
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第11762号, Conferred date:2022/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Study on Cutting-plane Algorithms for Mixed-integer Semidefinite Optimization

Ken Kobayashi
Department of Industrial Engineering and Economics
School of Engineering
Tokyo Institute of Technology

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Engineering

Supervisor: Professor Kazuhide Nakata

February 2022

ABSTRACT

This thesis focuses on mixed-integer semidefinite optimization problems, which minimize or maximize a linear objective function subject to constraints wherein a given matrix formed from the decision variables is positive semidefinite, and some of the variables are integer-valued. Since this problem includes nonlinearity and discreteness, various practical optimization problems can be formulated as a mixed-integer optimization problem. However, studies on algorithms for mixed-integer semidefinite optimization problems are relatively few, and solving large-sized mixed-integer semidefinite optimization problems has been difficult in practice. With this background in mind, this thesis reviews some cutting-plane algorithms for efficiently solving mixed-integer semidefinite optimization problems. Cutting-plane algorithms are optimization techniques to handle complex constraints or an objective function. Because of the extendability and ease of implementation, the cutting-plane algorithms have the potential to efficiently solve mixed-integer semidefinite optimization problems by exploiting the structure of the individual problems.

First, we consider the standard form of the mixed-integer semidefinite optimization problems and propose a general-purpose cutting-plane algorithm for solving it. We also devise a branch-and-cut algorithm to improve computational efficiency, which integrates the cutting-plane algorithm and a branch-and-bound algorithm.

Second, we focus on the best subset selection problem for eliminating multicollinearity from linear regression models. We give a mixed-integer semidefinite optimization formulation of the problem, in which a subset of explanatory variables is selected under an upper bound on the condition number of the correlation matrix of the selected variables. Then, we propose a specialized cutting-plane algorithm that exploits the structure of the condition number constraint.

Finally, we focus on portfolio selection problems formulated as a mixed-integer semidefinite optimization problem. Specifically, we consider a distributionally robust portfolio optimization problem with limiting the number of invested assets. By exploiting the problem structure, we develop a scalable cutting-plane algorithm for solving the model with the technique of positive semidefinite matrix completion. In addition, we extend the cutting-plane algorithm for the cardinality-constrained distributionally robust portfolio optimization so as to deal with another portfolio optimization problem that minimizes the conditional value-at-risk under a cardinality constraint. For this problem, we propose a cutting-plane method that incorporates into its subroutine another cutting-plane algorithm that efficiently minimizes the conditional value-at-risk.

Acknowledgements

First of all, I would like to express my deepest gratitude to Professor Kazuhide Nakata. He introduced me to the study of mathematical optimization and operations research and always gave me insightful comments. The stimulating discussions with him have been invaluable experiences to me. Without his support, I could not complete my research.

I would also like to thank Professor Yuichi Takano. His passionate and energetic attitude towards research always motivated me. Through the collaboration with him, I learned what a researcher should be.

I am grateful to my co-authors, Mr. Ryuta Tamura, Professor Ryuhei Miyashiro, and Professor Tomomi Matsui for their valuable comments and advices.

I would also like to thank my superiors, Dr. Hirokazu Anai, Dr. Hiromich Kobashi, Dr. Yuhei Umeda, and Dr. Hiroya Inakoshi for giving me a chance to pursue a doctoral degree. They carefully coordinated my work so that I could carry out my research comfortably.

Last but not least, I wish to thank my family for their continuous encouragement and support.

Ken Kobayashi, February 2022

Contents

Chapter 1

Introduction

Mixed-integer semidefinite optimization problems involve minimizing or maximizing a linear objective function subject to the constraints in which a given matrix formed from the decision variables is positive semidefinite and some of the variables are integer-valued. Due to the nonlinearity of the positive semidefinite constraint and discreteness of the integer constraints, this problem includes various practical optimization problems. This thesis concerns cutting-plane algorithms for solving mixed-integer semidefinite optimization problems. In this chapter, we give an introduction to this thesis.

We describe the background and motivation of this thesis in Section 1.1. In Section 1.2, we present the objective of this thesis and summarize our results. In Section 1.3, we define the mixed-integer semidefinite optimization problem and review its applications. In Section 1.4, we introduce the general framework of the cutting-plane algorithm and give a brief review of its recent development, including our results. In Section 1.5, we give an overview of this thesis, and in Section 1.6, we define the notation used in this thesis.

1.1 Background and motivation

In recent years, we have faced the problems of a shrinking workforce and exhaustion of fossil fuels. Therefore, there is a growing need to efficiently use limited management resources to realize a sustainable society. In addition, with the development of high-speed information and communication technologies, we often have to make decisions quickly in complex and uncertain situations.

A key technology to assist such complex decisions making is mathematical optimization. Mathematical optimization refers to methods of finding a solution that minimizes or maximizes a given objective function from a set of feasible solutions. This is one of the most important fields in applied mathematics and has been used in various fields of science and engineering, such as operations research, statistics, finance engineering, control theory, and chemistry.

When we formulate a certain problem as an optimization problem, we often encounter nonlinear objective functions, constraints, and discrete variables. For example, in statistics, we estimate the parameters of a model by minimizing a loss function. The loss function is often nonlinear, and such loss minimization problem is formulated as a nonlinear optimization problem. Also, if the data contains a lot of irrelevant features, it is necessary to exclude these features from the model when we estimate the model parameters. We can achieve such exclusion by introducing discrete variables that indicate whether we use the features for the estimation or not.

This thesis focuses on solving mixed-integer semidefinite optimization (MISDO)

problems. MISDO problems involve minimizing a linear objective function subject to constraints in which a given matrix formed from the decision variables is positive semidefinite (PSD) and some of the variables are integer-valued. Various nonlinear constraints can be expressed as PSD constraints [155, 161], and some important combinatorial optimization problems can be reformulated as (or approximated by) semidefinite optimization (SDO) problems [134, 147]. Also, integrality constraints in the MISDO problem can naturally describe the discrete nature of practical decision-making (e.g., a yes-or-no decision) [163]. Because of these advantages, MISDO problems have found uses in diverse fields, such as architecture [40, 171], control systems [89, 111, 153, 158], graph theory [10, 11, 135], signal processing [61, 130], surgery planning [172], machine learning [152], and statistical data analysis [8, 127, 151]. Therefore, solving MISDO problems efficiently is of great value in the sense of providing one of the tools for solving various kinds of problems in these fields.

In recent years, there has been remarkable progress in developing commercial and non-commercial solvers for mixed-integer optimization (MIO) problems and semidefinite optimization (SDO) problems. For MIO problems, we can use powerful commercial solvers such as Gurobi and CPLEX. These solvers are based on a branch-and-bound algorithm, which explores an enumeration tree while solving a continuous relaxation problem at each node. With these powerful solvers, large-scale mixed-integer linear optimization (MILO) problems and mixed-integer convex quadratic optimization (MIQO) problems can be solved efficiently. For SDO problems, on the other hand, interior-point methods are powerful algorithms to solve in practice. In fact, most of the available SDO solvers, such as MOSEK [120], SDPA [169], SDPT3 [156], and SeDuMi [148] are based on the interior-point method, and improving their scalability has been an important and active area of research.

While algorithms and solvers for MIO and SDO problems have been well studied, there are relatively few treatments for MISDO problems. The current MIO solvers (i.e., Gurobi and CPLEX) cannot handle semidefinite constraints by themselves, and methods of solving MISDO problems are currently in an immature stage. A natural way of solving an MISDO problem is to combine a branch-and-bound algorithm for MIO problems and an interior-point algorithm for SDO problems. Recently, Gally *et al.* [62] developed a general-purpose MISDO solver named SCIP-SDP; it is regarded as the most promising solver at present. SCIP-SDP combines the branch-and-bound framework of SCIP [1] with SDO solvers that use interior-point methods. Also, most of the specialized algorithms for specific MISDO applications are based on the branch-and-bound algorithm [8, 10, 11, 40, 130, 135, 139, 171]. However, there is a serious drawback to this approach, wherein if we just combine a branch-and-bound algorithm and an interior-point method, we cannot implement a warm-starting strategy for efficiently solving a series of SDO problems because the interior-point methods do not receive a given initial solution. Indeed, Gleixner *et al.* [63] implemented several warm-starting techniques [39, 68, 81, 143, 146] in SCIP-SDP, but their effects are marginal or sometimes negative in some problem instances. Therefore, the size of the MISDO problems that we can solve still has been limited.

Against this background, this thesis focuses on cutting-plane algorithms as alternatives to the branch-and-cut algorithm and reviews the cutting-plane algorithms for solving MISDO problems. Cutting-plane algorithms are optimization techniques to handle complex constraints or an objective function. A cutting-plane algorithm first relaxes the constraints or the objective function and searches for an optimal solution by solving the resultant relaxed problems while adding

linear constraints called *cutting planes* or *valid inequalities*. The cutting-plane algorithm was devised by Gomory [67] to solve MILO problems, and it was later extended by Kelly [86] to handle convex optimization problems. The cutting-plane algorithm has the advantages of extensibility and ease of implementation. Because of its tractability, the cutting-plane algorithm has been actively studied in nonsmooth convex optimization (see, for example, [64, 73, 100, 110]). In addition, if the relaxation problem is easy to solve, we can improve the performance of the cutting-plane algorithm by using this property. Thus, if an MISDO problem can be relaxed into a tractable one, a cutting-plane algorithm will be a promising way of solving it efficiently.

1.2 Objective and results

From this perspective, we aim to design efficient cutting-plane algorithms to solve MISDO problems. Specifically, we firstly propose a general-purpose cutting-plane algorithm for solving the standard form of MISDO problems. This method allows us to handle the general MISDO problem with existing state-of-the-art MIO solvers. Secondly, we devise specialized cutting-plane algorithms for some important MISDO problems. In statistics and financial engineering, we sometimes encounter large-sized MISDO problem instances due to the increasing size of the data sets in these fields, and such problems are difficult to solve even with general-purpose algorithms. To handle large-sized problem instances, we design specialized cutting-plane algorithms that exploit the individual structures of these problems. In particular, this thesis focuses on a variable selection and portfolio selection problem formulated as MISDO problems and proposes specialized cutting-plane algorithms for solving them efficiently.

The results of this thesis are summarized as follows:

Branch-and-cut algorithm for mixed-integer semidefinite optimization (Chapter 2)

First, we study general-purpose cutting-plane algorithms for solving the standard form of MISDO problems. We develop a cutting-plane algorithm for solving MISDO problems by extending the existing algorithm for SDO problems proposed by Konno *et al.* [100]. In the proposed algorithm, we relax the PSD constraint and solve the relaxed MILO problem repeatedly while adding a cutting plane to the constraints at each iteration so as to satisfy the PSD constraint. We prove the convergence properties of the algorithm. In addition, to speed up the computation, we devise a branch-and-cut algorithm, where cutting planes are dynamically added during a branch-and-bound procedure. Our experimental results demonstrate that, for many problem instances, our branch-and-cut algorithm delivered superior performance compared with other general-purpose MISDO solvers in terms of computational efficiency and stability.

Cutting-plane algorithm for best subset selection for eliminating multicollinearity (Chapter 3)

Second, we consider a subset selection of explanatory variables for eliminating multicollinearity from linear regression models. Specifically, we select the best subset of the variables subject to an upper bound on the condition number of the correlation matrix of selected variables. We first formulate the best subset selection under the condition number constraint as an MISDO problem. Since the

current general-purpose MISDO solver only handled very small-sized instances, which we will show in the corresponding chapter, we also developed a specialized cutting-plane algorithm. To approximate the condition number constraint, this cutting-plane algorithm iteratively appends cutting planes to the relaxed MIQO problem. Here, we also propose to generate strong cutting planes by effectively using heuristic search to improve the computational efficiency of our algorithm. Computational results demonstrate that subset selection with our MISDO formulation succeeds when the number of candidate explanatory variables is small. Additionally, our specialized cutting-plane algorithm frequently provides solutions of better quality than those obtained by local search algorithms for subset selection.

Cutting-plane algorithm for cardinality-constrained distributionally robust portfolio optimization (Chapter 4)

Third, we focus on portfolio selection problems. In particular, we study a distributionally robust portfolio optimization model with a cardinality constraint, which limits the number of invested assets. This problem is formulated as a MISDO problem, and thus, solving it exactly is computationally challenging when the number of investable assets is large. To overcome this issue, we propose a specialized cutting-plane algorithm to solve the cardinality-constrained distributionally robust optimization problem. We first reformulate the problem as a bilevel optimization problem and design a cutting-plane algorithm for solving the upper-level problem. To generate cutting planes efficiently, we apply the technique of positive semidefinite matrix completion to the lower-level problem and show that we can calculate subgradients efficiently regardless of the number of investable assets. The numerical experiments demonstrate that our cutting-plane algorithm was very effective compared with the existing MISDO solver, especially when the number of investable assets is large. In addition, the out-of-sample investment performances given by the cardinality-constrained distributionally robust model were better than those of the conventional mean-variance model.

Bilevel cutting-plane algorithm for cardinality-constrained mean-CVaR optimization (Chapter 5)

Finally, we focus on cardinality-constrained conditional value-at-risk (CVaR) minimization problems and extend the cutting-plane algorithm for the distributionally-robust portfolio optimization problem. While the cardinality-constrained CVaR minimization problem is formulated as an MILO problem, which is a special case of MISDO problem and can be handled by state-of-the-art MIO solvers, its problem size depends not only on the number of investable assets but also on the number of asset return scenarios. Thus, the computational efficiency decreases when the number of scenarios is large. To overcome this challenge, we propose a specialized cutting-plane algorithm named the *bilevel cutting-plane algorithm* for exactly solving the cardinality-constrained mean-CVaR portfolio optimization problem. In our proposal, we extend the cutting-plane algorithm discussed in Chapter 4 so that it incorporates another cutting-plane algorithm that efficiently minimizes CVaR. Numerical experiments demonstrate that, compared with other MIO approaches, our algorithm can provide optimal solutions to large problem instances faster.

1.3 Mixed-integer semidefinite optimization problems

In this section, we define MISDO problems as an extension of SDO problems and give a brief overview of their applications.

We begin by introducing linear optimization (LO) problems. An LO problem is the most basic optimization problem. It aims to minimize or maximize a linear function subject to finite linear constraints. The standard form of LO problems is formulated as:

$$\begin{array}{llll} \text{(LO-P)} & \underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}} & \boldsymbol{c}^\top\boldsymbol{x} & \\ & \text{subject to} & \boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}, & \\ & & \boldsymbol{x}\geq\boldsymbol{0}, & \end{array} \qquad \begin{array}{llll} \text{(LO-D)} & \underset{\boldsymbol{y}\in\mathbb{R}^M}{\text{maximize}} & \boldsymbol{b}^\top\boldsymbol{y} & \\ & \text{subject to} & \boldsymbol{A}^\top\boldsymbol{y}\leq\boldsymbol{c}, & \end{array} \tag{1.1}$$

where (LO-P) and (LO-D) denote the primal and dual LO problem, respectively; $\boldsymbol{A}\in\mathbb{R}^{M\times N}$, $\boldsymbol{b}\in\mathbb{R}^M$, and $\boldsymbol{c}\in\mathbb{R}^N$ are given constants; $(\boldsymbol{x},\boldsymbol{y})$ is a decision variable and $\boldsymbol{x}\geq\boldsymbol{0}$ represents that all the entries of $\boldsymbol{x}$ are nonnegative.

SDO is an extension of LO problems to an optimization problem over the space of $N\times N$ real symmetric matrices. Here, we denote the set of $N\times N$ real symmetric matrices by $\mathcal{S}^N$, and define the standard inner product between two matrices $\boldsymbol{X}=(X_{nm}),\boldsymbol{Y}=(Y_{nm})\in\mathcal{S}^N$ as

$$\boldsymbol{X}\bullet\boldsymbol{Y}=\mathrm{Tr}(\boldsymbol{X}\boldsymbol{Y})=\sum_{n=1}^{N}\sum_{m=1}^{N}X_{nm}Y_{nm}.$$

Let $\boldsymbol{A}_1,\boldsymbol{A}_2,\ldots,\boldsymbol{A}_M\in\mathcal{S}^N$, $\boldsymbol{b}\in\mathbb{R}^M$, and $\boldsymbol{C}\in\mathcal{S}^N$ be given constants. Then, the standard primal and dual forms of SDO problems are given by

$$\begin{array}{llll} \text{(SDO-P)} & \underset{\boldsymbol{X}\in\mathcal{S}^N}{\text{minimize}} & \boldsymbol{C}\bullet\boldsymbol{X} & \\ & \text{subject to} & \mathcal{A}(\boldsymbol{X})=\boldsymbol{b}, & \\ & & \boldsymbol{X}\succeq\boldsymbol{O}, & \end{array} \qquad \begin{array}{llll} \text{(SDO-D)} & \underset{\boldsymbol{y}\in\mathbb{R}^M}{\text{maximize}} & \boldsymbol{b}^\top\boldsymbol{y} & \\ & \text{subject to} & \mathcal{A}^\top(\boldsymbol{y})\preceq\boldsymbol{C}, & \end{array} \tag{1.2}$$

where (SDO-P) and (SDO-D) denote the primal and dual SDO problem, respectively; $(\boldsymbol{X},\boldsymbol{y})$ is a decision variable, and $\boldsymbol{X}\succeq\boldsymbol{O}$ denotes that $\boldsymbol{X}$ is positive semidefinite i.e., $\boldsymbol{v}^\top\boldsymbol{X}\boldsymbol{v}\geq 0$ for all $\boldsymbol{v}\in\mathbb{R}^N$; $\mathcal{A}:\mathcal{S}^N\to\mathbb{R}^M$ is a linear mapping defined by $\mathcal{A}(\boldsymbol{X}):=(\boldsymbol{A}_1\bullet\boldsymbol{X},\boldsymbol{A}_2\bullet\boldsymbol{X},\ldots,\boldsymbol{A}_M\bullet\boldsymbol{X})^\top$ and $\mathcal{A}^\top:\mathbb{R}^M\to\mathcal{S}^N$ is its adjoint defined by $\mathcal{A}^\top(\boldsymbol{y}):=\sum_{m=1}^{M}y_m\boldsymbol{A}_m$.

It is easy to see that the SDO problem (1.2) includes the LO problem (1.1) as a special case. For a vector $\boldsymbol{x}\in\mathbb{R}^N$, let $\mathrm{Diag}(\boldsymbol{x})\in\mathcal{S}^N$ be the diagonal matrix whose diagonal elements are $\boldsymbol{x}$. Then, the nonnegativity of $\boldsymbol{x}$ is equivalent to the positive semidefiniteness of $\mathrm{Diag}(\boldsymbol{x})$ as follows:

$$\boldsymbol{x}\geq\boldsymbol{O}\Longleftrightarrow\mathrm{Diag}(\boldsymbol{x})\succeq\boldsymbol{O}.$$

In addition, the inner product of two vectors $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^N$ is expressed as that of $\mathrm{Diag}(\boldsymbol{x})$ and $\mathrm{Diag}(\boldsymbol{y})$:

$$\boldsymbol{x}^\top\boldsymbol{y}=\mathrm{Diag}(\boldsymbol{x})\bullet\mathrm{Diag}(\boldsymbol{y}).$$

Thus, we can reformulate the LO problem (1.1) as an SDP problem (1.2).

The MISDO problem is defined as an SDO problem with integer constraints and its dual form is given by:

$$\begin{array}{llll} \text{(MISDO-D)} & \underset{\boldsymbol{y}\in\mathbb{R}^M}{\text{maximize}} & \boldsymbol{b}^\top\boldsymbol{y} & \\ & \text{subject to} & \mathcal{A}^\top(\boldsymbol{y})\preceq\boldsymbol{C}, & \\ & & \boldsymbol{y}\in\{0,1\}^{M_\mathrm{b}}\times\mathbb{R}^{M_\mathrm{c}}. & \end{array} \tag{1.3}$$

where $\boldsymbol{y} := (y_m) \in \mathbb{R}^M$; M_b and M_c are, respectively, the numbers of binary and continuous variables satisfying $M = M_\mathrm{b} + M_\mathrm{c}$. Alternatively, we may also consider MISDO problems in the primal form: .

$$\begin{aligned} \text{(MISDO-P)} \quad & \underset{\boldsymbol{X} \in \mathcal{S}^N}{\text{minimize}} && \boldsymbol{C} \bullet \boldsymbol{X} \\ & \text{subject to} && \mathcal{A}(\boldsymbol{X}) = \boldsymbol{b}, \\ & && \boldsymbol{X} \succeq \boldsymbol{O}, \\ & && X_{nm} \in \{0,1\}, \quad (\forall (n,m) \in B), \end{aligned}$$

where $\boldsymbol{X} := (X_{nm}) \in \mathcal{S}^N$, and the index set $B \subseteq \{1,2,\ldots,N\} \times \{1,2,\ldots,N\}$ imposes binary-valued entries in $\boldsymbol{X}$.

Since the MISDO problem involves nonlinearity and discreteness because of the semidefinite and binary constraints, it includes various practical optimization problems. One of the main fields where MISDO problems appear is combinatorial optimization. For example, graph partitioning problems are formulated as an MISDO problem [10, 11]. Also, Rendl and Rinaldi [135] proposed an algorithm for exactly solving max-cut problems with an MISDO formulation. In structural design, Yonekura and Kannno [171] considered a robust truss topology optimization and formulated this problem as an MISDO problem. In control theory, Joshi and Boyd [89] focused on a sensor selection problem that minimizes the error in estimated parameters and showed that this problem can be expressed as an MISDO problem. In data mining, MISDO problems are used to obtain an optimal clustering in the sense of a given metric [8, 127]. In statistics, Tamura *et al.* [151] examined the best subset selection for eliminating multicollinearity from a linear regression model and proposed a MISDO formulation with a condition number constraint. Moreover, MISDO problems have recently been used in robust optimization. In particular, Zhang *et al.* [172] focused on a robust optimization for allocating surgery blocks and formulated the problem as an MISDO problem.

1.4 Cutting-plane algorithms

In this section, we explain the general framework of cutting-plane algorithms for solving convex optimization problems and give a brief overview of recent developments on cutting-plane algorithms, including our results.

The cutting-plane algorithm was first introduced by Gomory [67] for solving MILO problems, and it was extended by Kelly [86] to general convex optimization problems. Here, we explain the general framework proposed by Kelly [86]. Let us consider the following convex optimization problem:

$$\underset{\boldsymbol{x} \in \mathbb{R}^N}{\text{minimize}} \quad f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{x} \in \mathcal{F}, \tag{1.5}$$

where $\mathcal{F} \subseteq \mathbb{R}^N$ is a convex set, $f : \mathbb{R}^N \to \mathbb{R}$ is a convex function, and we assume that the above problem has an optimal solution.

The main ingredient of cutting-plane algorithms is the *hyperplane separation theorem* [36, 82, 136]. For a convex set $\mathcal{F} \subseteq \mathbb{R}^N$ and a point $\hat{\boldsymbol{x}} \notin \mathcal{F}$, the hyperplane separation theorem ensures the existence of $\boldsymbol{a} \neq \boldsymbol{0}$ and b such that $\boldsymbol{a}^\top \hat{\boldsymbol{x}} > b$ and $\boldsymbol{a}^\top \boldsymbol{x} \leq b$ for all $\boldsymbol{x} \in \mathcal{F}$. We call this inequality separating $\mathcal{F}$ and $\hat{\boldsymbol{x}}$ a *cutting plane*. Also, we call the inequality satisfying $\boldsymbol{a}^\top \boldsymbol{x} \leq b$ for all $\boldsymbol{x} \in \mathcal{F}$ *valid* or a *valid inequality* for $\mathcal{F}$.

The cutting-plane algorithm solves Problem (1.5) by refining convex polyhedrons that contain $\mathcal{F}$ while cutting off infeasible solutions with the hyperplane

separation theorem. Here, let us define $\mathcal{F}_1$ to be the initial feasible region that contains $\mathcal{F}$. At the tth iteration ($t \geq 1$), the cutting-plane algorithm solves the following optimization problem:

$$\underset{\boldsymbol{x} \in \mathbb{R}^N}{\text{minimize}} \quad f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{x} \in \mathcal{F}_t, \tag{1.6}$$

where $\mathcal{F}_t$ is a relaxed feasible region at the tth iteration satisfying $\mathcal{F}_t \subseteq \mathcal{F}_1$. Let $\boldsymbol{x}_t$ be an optimal solution to Problem (1.6). Then, we check the feasibility of $\boldsymbol{x}_t$. If $\boldsymbol{x}_t \in \mathcal{F}$, $\boldsymbol{x}_t$ is optimal for the original problem (1.5), and we terminate the algorithm and output $\boldsymbol{x}_t$ as an optimal solution. If $\boldsymbol{x}_t \notin \mathcal{F}$, we find a cutting plane $\boldsymbol{a}^\top \boldsymbol{x} \leq b$ separating $\boldsymbol{x}_t$ from $\mathcal{F}_t$ and update the relaxed feasible region by adding the cutting plane to the constraints as follows:

$$\mathcal{F}_{t+1} \leftarrow \mathcal{F}_t \cap \{\boldsymbol{x} \in \mathbb{R}^N \mid \boldsymbol{a}^\top \boldsymbol{x} \leq b\}, \tag{1.7}$$

After updating the feasible region, we set $t \leftarrow t + 1$ and solve Problem (1.6) again. We repeat this procedure until $\boldsymbol{x}_t \in \mathcal{F}$ (possibly up to some tolerance). We summarize the general scheme of the cutting-plane algorithm for Problem (1.5) by Algorithm 1.1.

Algorithm 1.1 General scheme of cutting-plane algorithms for solving Problem (1.5)

Step 0 (Initialization) Set $t \leftarrow 1$ and the initial feasible region $\mathcal{F}_1 \supseteq \mathcal{F}$.

Step 1 (Relaxed Problem) Solve Problem (1.6). Let $\boldsymbol{x}_t$ be an optimal solution.

Step 2 (Cut Generation) If $\boldsymbol{x}_t \in \mathcal{F}$, terminate the algorithm. Otherwise, find a cutting plane $\boldsymbol{a}^\top \boldsymbol{x} \leq b$ separating $\boldsymbol{x}_t$ from $\mathcal{F}$.

Step 3 (Update Feasible Region) Update the feasible region as in Eq. (1.7).

Step 4 Set $t \leftarrow t + 1$ and return to Step 1.

Cutting-plane algorithms have the following advantages:

- They do not require the objective and constraint functions to be differentiable. As long as we can determine whether a solution belongs to $\mathcal{F}$ and generate cutting planes for $\mathcal{F}$, we can apply the cutting-plane algorithm to the problem to be solved.
- If the relaxed problem has a nice solvable structure, we can use it to solve the relaxed problem and accelerate the entire computation. For example, if state-of-the-art solvers can handle the relaxed problem, we can use them in the cutting-plane algorithm.
- Cutting-plane algorithms have good extendability. If we can generate a cutting plane for $\mathcal{F}$, Algorithm 1.1 can also be applied to the following mixed-integer optimization problem:

$$\underset{\boldsymbol{x} \in \mathbb{R}^N}{\text{minimize}} \quad f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{x} \in \mathcal{F} \cap (\{0,1\}^p \times \mathbb{R}^{N-p}), \tag{1.8}$$

  where p denotes the number of binary variables, and $\{0,1\}^p$ is the set of p dimensional vectors whose all entries are 0 or 1.

Because of these advantages, cutting-plane algorithms have come to be applied to various optimization problems. Regrettably, we can not describe all of the studies on the cutting-plan algorithms, but we will mention the results that are strongly relevant to this thesis. Konno *et al.* [100] proposed a cutting-plane algorithm for solving SDO problems. A similar algorithm was proposed by Krishnan *et al.* [102]. Recently, an MISDO solver named CUTSDP, which is implemented in YALMIP [106], was made available; it employs the outer approximation algorithm, which is an extension of the cutting-plane algorithm. Coey *et al.* [45] developed a general-purpose solver named Pajarito for mixed-integer conic optimization problems. Kobayashi and Takano [94] extended the cutting-plane algorithm for SDO problems and proposed a branch-and-cut algorithm for solving MISDO problems.

For specific MIO problems, Bertsimas and King [32] considered subset selection in linear regression, which is a classical problem in statistics, and employed a cutting-plane strategy. In financial engineering, Bertsimas and Cory-Wright proposed a cutting-plane algorithm for cardinality-constrained mean-variance portfolio optimization [24]. This cutting-plane algorithm was extended to cover sparse principal component analysis [22] and covariance selection problems [33]. Kobayashi *et al.* [96] considered a cardinality-constrained distributionally robust portfolio optimization, which is formulated as an MISDO problem, and devised an efficient cutting-plane algorithm for solving it with the technique of positive semidefinite matrix completion [60, 121]. Künzi-Bay and Mayer [103] proposed a cutting-plane algorithm for minimizing CVaR, which is a risk measure in finance, and variants of this algorithm were developed [4, 80, 149]. Kobayashi *et al.* [95] proposed an algorithm named the *bilevel cutting-plane algorithm* for solving cardinality-constrained CVaR minimization problems.

Finally, we mention three points that should be kept in mind when we design an efficient cutting-plane algorithm. First, the relaxed problem (1.6), which is solved at each iteration, should be tractable. Second, cutting planes should be generated with a low computational cost. The last point is the *strength* of cutting planes. For two cutting planes $\boldsymbol{a}_1^\top \boldsymbol{x} \leq b_1$ and $\boldsymbol{a}_2^\top \boldsymbol{x} \leq b_2$ for $\mathcal{F}$, we call the former is *stronger* than the latter if the following condition holds:

$$\{\boldsymbol{x} \in \mathbb{R}^N \mid \boldsymbol{a}_1^\top \boldsymbol{x} \leq b_1\} \subseteq \{\boldsymbol{x} \in \mathbb{R}^N \mid \boldsymbol{a}_2^\top \boldsymbol{x} \leq b_2\}.$$

In this case, the stronger cutting plane cuts off more infeasible solutions; so it is preferred when we add a cutting plane to the constraints. Thus, when we generate a cutting plane, it should be strong.

1.5 Thesis overview

The remainder of this thesis is structured as follows. In Chapter 2, we study the standard form of MISDO problems and devise general-purpose cutting-plane algorithms for solving the problem. In Chapter 3, we consider the best subset selection problem for eliminating multicollinearity from linear regression models. We formulate this problem as an MISDO problem and propose a specialized cutting-plane algorithm that exploits the structure of the condition number constraint. In Chapters 4 and 5, we focus on cardinality-constrained portfolio optimization problems. We first focus on distributionally robust portfolio optimization in Chapter 4. We devise a scalable cutting-plane algorithm where we generate cutting planes efficiently by using the technique of positive semidefinite matrix completion. In Chapter 5, we deal with the cardinality-constrained CVaR

minimization problem and extend the cutting-plane algorithm discussed in Chapter 4 so that it can handle a large number of asset return scenarios. Chapter 6 is devoted to concluding remarks. Note that the details of the existing research on each problem are given in the corresponding chapter.

1.6 Notation

We define the set of real numbers, integers, and positive integers as $\mathbb{R}, \mathbb{Z}$, and $\mathbb{N}$, respectively. For $N \in \mathbb{N}$, $[N]$ represents the set $\{1, 2, \ldots, N\}$. We denote by $\boldsymbol{x} := (x_n) \in \mathbb{R}^N$ an N-dimensional column vector of real numbers. The zero and all-one vectors of appropriate size are written as $\mathbf{0}$ and $\mathbf{1}$, respectively. If all the entries of $\boldsymbol{x} \in \mathbb{R}^N$ are nonnegative, we write $\boldsymbol{x} \geq \mathbf{0}$. For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$, we use $\boldsymbol{x} \geq \boldsymbol{y}$ to mean $\boldsymbol{x} - \boldsymbol{y} \geq \mathbf{0}$. The transpose of a matrix $\boldsymbol{x} \in \mathbb{R}^N$ is denoted by $\boldsymbol{x}^\top$. The standard inner product on $\mathbb{R}^N$ is given by $\boldsymbol{x}^\top \boldsymbol{y} = \sum_{n \in [N]} x_n y_n$ for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$. The ℓ_p-norm of the vector $\boldsymbol{x}$ is denoted by $\|\boldsymbol{x}\|_p := \left(\sum_{n \in [N]} |x_n|^p \right)^{1/p}$.

We denote by $\boldsymbol{X} := (X_{nm}) \in \mathbb{R}^{N \times M}$ an $N \times M$ real matrix. The zero and identity matrices of appropriate sizes are written as $\boldsymbol{O}$ and $\boldsymbol{I}$, respectively. The trace $\mathrm{Tr}(\cdot)$ is the sum of the diagonal elements of a square matrix. The $\mathrm{diag}(\cdot)$ operator extracts the main diagonal from the matrix as a vector, and the $\mathrm{Diag}(\cdot)$ operator maps the vector to the diagonal matrix. The transpose of a matrix $\boldsymbol{X} \in \mathbb{R}^{N \times M}$ is denoted by $\boldsymbol{X}^\top$.

The set of all symmetric $N \times N$ real matrices is denoted by $\mathcal{S}^N$. The standard inner product on $\mathcal{S}^N$ is given by $\boldsymbol{X} \bullet \boldsymbol{Y} = \sum_{m \in [N]} \sum_{n \in [N]} X_{mn} Y_{mn}$ for $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{S}^N$. The minimum and maximum eigenvalues of matrix $\boldsymbol{X} \in \mathcal{S}^N$ are denoted by $\lambda_{\min}(\boldsymbol{X})$ and $\lambda_{\max}(\boldsymbol{X})$, respectively.

A matrix $\boldsymbol{X} \in \mathcal{S}^N$ is positive semidefinite (PSD) if

$$\boldsymbol{v}^\top \boldsymbol{X} \boldsymbol{v} \geq 0$$

for all $\boldsymbol{v} \in \mathbb{R}^N$. We write $\boldsymbol{X} \succeq \boldsymbol{O}$ if the matrix $\boldsymbol{X}$ is PSD. For $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{S}^N$, we use $\boldsymbol{X} \succeq \boldsymbol{Y}$ to mean $\boldsymbol{X} - \boldsymbol{Y} \succeq \boldsymbol{O}$. A matrix $\boldsymbol{X} \in \mathcal{S}^N$ is positive definite if

$$\boldsymbol{v}^\top \boldsymbol{X} \boldsymbol{v} > 0$$

for all $\boldsymbol{v} \neq \mathbf{0}$. We write $\boldsymbol{X} \succ \mathbf{0}$ if the matrix $\boldsymbol{X}$ is positive definite. For a symmetric matrix $\boldsymbol{X} \in \mathcal{S}^N$, its positive semidefiniteness and nonnegativity of its eigenvalues are equivalent as follows:

$$\boldsymbol{X} \succeq \boldsymbol{O} \Longleftrightarrow \lambda_{\min}(\boldsymbol{X}) \geq 0.$$

Chapter 2

Branch-and-cut Algorithm for Mixed-integer Semidefinite Optimization

In this chapter, we study the general-purpose cutting-plane algorithms for solving MISDO problems. In particular, we focus on the standard form of MISDO problems and develop a cutting-plane algorithm for solving it by extending the existing algorithm for solving SDO problems proposed by Konno *et al.* [100]. We prove the convergence properties of the algorithm. In addition, to speed up the computation, we devise a branch-and-cut algorithm, in which cutting-planes are dynamically added during a branch-and-bound procedure. The content of this chapter is included in Kobayashi *et al.* [94].

We give an introduction to this study and summarize our contribution in Section 2.1. In Section 2.2, we show the definition of the MISDO problems again. In Section 2.3, we present our cutting-plane algorithm and analyze its convergence properties. In Section 2.4, we describe our branch-and-cut algorithm for solving MISDO problems. In Section 2.5, we report the computational results of tests of our algorithms, and in Section 2.6, we conclude with a brief summary.

2.1 Introduction

In this chapter, we focus on the standard form of MISDO problems. As we mentioned in Chapter 1, the standard MISDO problem includes various kinds of optimization problems appearing in diverse fields, such as combinatorial optimization [134, 147], architecture [40, 171], control systems [89, 111, 153, 158], graph theory [10, 11, 135], signal processing [61, 130], surgery planning [172], and statistical data analysis [8, 127, 151]. Thus, developing an efficient general-purpose algorithm for solving the standard MISDO formulation is of great significance in terms of providing a framework for handling these various optimization problems in a unified manner.

A common way to solve MISDO problems is to use a branch-and-bound (B&B) algorithm. Gally et al. [62] developed a general-purpose MISDO solver, SCIP-SDP, by combining the B&B framework of SCIP [1] with SDO solvers that use interior-point methods. Also, the existing specialized algorithms for specific MISDO applications are mainly based on the B&B algorithm [8, 10, 11, 40, 130, 135, 139, 171]. These B&B algorithms involve solving a continuous relaxation (i.e., SDO) problem at each node of the enumeration tree. In this case, however, we cannot implement a warm-starting strategy for efficiently solving a series of SDO problems because the interior-point methods do not receive a given initial solution. Moreover, the progress of the B&B algorithm is sometimes disrupted and returns an incorrect solution due to numerical instability, especially when Slater's condition does not hold for some of the SDO problems.

As an alternative to the B&B framework, we focus on the cutting-plane algorithm [86, 162]. Specifically, we shall extend the cutting-plane algorithms [97, 102] that were originally developed for solving SDO problems. These algorithms can readily be extended to solving MISDO problems by removing the PSD constraint and repeatedly solving the resultant MILO problem in which a cutting plane is imposed at each iteration to exclude infeasible solutions for the removed PSD constraint. This approach has the advantage of being able to use state-of-the-art optimization software (e.g., Gurobi or CPLEX) when solving MILO problems. However, to solve an MISDO problem with a sufficient degree of accuracy, we must deal with an exponentially large number of MILO problems [38]. A similar cutting-plane algorithm for solving MISDO problems has previously been described [108] and implemented in YALMIP [106]. To the best of our knowledge, however, the theoretical properties of the algorithm have not been fully described, and no detailed computational results have been reported.

The purpose of this chapter is to describe an effective computational framework for solving MISDO problems. First, we formulate a cutting-plane algorithm for solving MISDO problems and prove its convergence properties. We then devise a high-performance branch-and-cut (B&C) algorithm on the basis of this cutting-plane algorithm. Specifically, we start by solving an MILO problem in which the PSD constraint has been relaxed, and then we include cutting planes for the constraint dynamically during the B&B procedure. We implement this B&C algorithm through the use of a callback function in the Gurobi Optimizer. We compare the computational performance of our algorithms with that of SCIP-SDP [62] and CUTSDP for three types of MISDO problems: random instances, computing restricted isometry constants [61], and robust truss topology design [171].

The main advantages of our B&C algorithm can be summarized as follows:

- Our B&C algorithm solves an MILO problem subject to a series of cutting planes. In contrast to the existing B&B algorithms that handle a series of SDO problems, a warm-starting strategy using the dual simplex method makes the B&C computation much faster.

- Our B&C algorithm can be implemented using sophisticated MILO software, such as Gurobi or CPLEX, and it deals with no SDO problems that may cause numerical instability. Hence, computation using our algorithm is very stable.

- Our B&C algorithm adds cutting planes dynamically during the MILO computation. As a result, our algorithm needs to execute the B&B procedure only once, in contrast with the cutting-plane algorithm, which repeats the B&B procedure.

2.2 Problem formulation

In this chapter, we focus on solving the dual standard form of MISDO problems. Recall that the dual standard dual form of MISDO problems is expressed as

follows:

$$\underset{\boldsymbol{y}}{\text{maximize}} \quad \boldsymbol{b}^\top \boldsymbol{y} \tag{2.1a}$$

$$\text{subject to} \quad \mathcal{A}^\top(\boldsymbol{y}) := \sum_{m\in[M]} \boldsymbol{A}_m y_m \preceq \boldsymbol{C}, \tag{2.1b}$$

$$\boldsymbol{y} \in \{0,1\}^{M_\mathrm{b}} \times \mathbb{R}^{M_\mathrm{c}}, \tag{2.1c}$$

where $\boldsymbol{b} := (b_m) \in \mathbb{R}^M$ and $\boldsymbol{A}_m \in \mathcal{S}^N$ $(m \in [M])$ are given constants, and $\boldsymbol{y}$ is a vector of binary and continuous decision variables. Note that $M = M_\mathrm{b} + M_\mathrm{c}$ holds, where M_b and M_c are, respectively, the numbers of binary and continuous decision variables. Note that we may also consider MISDO problems in the primal form in Eq. (1.4). In the following sections, we shall deal with the problem (2.1), but our algorithms can readily be applied to the primal standard form.

We assume that the feasible region of the problem (2.1) is bounded even if the PSD constraint (2.1b) is removed. Since

$$\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}) \succeq \boldsymbol{O} \;\Rightarrow\; \mathrm{diag}(\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y})) \geq \boldsymbol{0},$$

we define the relaxed feasible region

$$\mathcal{Y} := \{\boldsymbol{y} \in \{0,1\}^{M_\mathrm{b}} \times \mathbb{R}^{M_\mathrm{c}} \mid \mathrm{diag}(\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y})) \geq \boldsymbol{0}\}.$$

Assumption 2.1. The relaxed feasible region $\mathcal{Y}$ is bounded.

Let $\boldsymbol{y}^\star$ be an optimal solution to the problem (2.1). Note that Assumption 2.1 is fulfilled if the following constraint is incorporated into the definition of the feasible region:

$$-U \leq y_m \leq U, \quad M_\mathrm{b} + 1 \leq \forall i \leq M,$$

where U is a sufficiently large number such that this constraint is satisfied by $\boldsymbol{y}^\star$.

2.3 Cutting-plane algorithm

Our cutting-plane algorithm for solving MISDO problems is an extension of the cutting-plane algorithms [97, 102] for solving SDO problems. It can also be regarded as an improved version of the extended cutting-plane algorithm [162] in the sense that our algorithm makes it possible to handle PSD constraints.

Our algorithm starts with defining the initial feasible region as $\mathcal{F}_1 := \mathcal{Y}$ where the PSD constraint (2.1b) has been removed. At tth iteration $(t \geq 1)$, our algorithm solves the following relaxed MILO problem:

$$\underset{\boldsymbol{y}}{\text{maximize}} \quad \boldsymbol{b}^\top \boldsymbol{y} \quad \text{subject to} \quad \boldsymbol{y} \in \mathcal{F}_t, \tag{2.2}$$

where $\mathcal{F}_t$ is a relaxed feasible region at the tth iteration such that $\mathcal{F}_t \subseteq \mathcal{F}_1$. If $\mathcal{F}_t = \emptyset$, the problem (2.1) is infeasible because its feasible region is contained in $\mathcal{Y}$. Otherwise, since the feasible region $\mathcal{F}_t$ is bounded from Assumption 2.1 and $\mathcal{F}_t \subseteq \mathcal{F}_1$, there exists an optimal solution $\boldsymbol{y}_t$ to the problem (2.2).

We next check whether the PSD constraint (2.1b) is satisfied by the relaxed solution $\boldsymbol{y}_t$. If $\lambda_{\min}(\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}_t)) \geq 0$, then $\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}_t)$ is PSD and $\boldsymbol{y}_t$ is an optimal solution to the original problem (2.1), so we terminate the algorithm. Otherwise, let $\boldsymbol{d}_t \in \mathbb{R}^N$ be the normalized eigenvector of $\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}_t)$ corresponding to the minimum eigenvalue. It follows that

$$\boldsymbol{d}_t^\top (\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}_t))\boldsymbol{d}_t = \lambda_{\min}(\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}_t)) < 0.$$

Accordingly, we can exclude the solution $\boldsymbol{y}_t$ from the feasible region by incorporating the constraint

$$\boldsymbol{d}_t^\top (\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}))\boldsymbol{d}_t \geq 0. \tag{2.3}$$

Note that this constraint is a valid inequality for the PSD constraint (2.1b) because

$$\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}) \succeq \boldsymbol{O} \iff \left\langle \boldsymbol{d}^\top (\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}))\boldsymbol{d} \geq 0,\ \forall \boldsymbol{d} \in \mathbb{R}^n \right\rangle.$$

After constraint (2.3) is included, we set $t \leftarrow t+1$ and solve the MILO problem (2.2) again. We repeat this process (i.e., solving the MILO problem and adding a cutting plane) until $\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}_t)$ becomes nearly PSD:

$$\lambda_{\min}(\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}_t)) \geq -\varepsilon, \tag{2.4}$$

where ε is a sufficiently small non-negative number representing a tolerance for feasibility. In this case, constraints (2.1c) and (2.4) are satisfied by $\boldsymbol{y}_t$, and the value of the objective function is not less than the maximum (i.e., $\boldsymbol{b}^\top \boldsymbol{y}_t \geq \boldsymbol{b}^\top \boldsymbol{y}^\star$) because $\boldsymbol{y}_t$ solves the relaxed problem (2.2). Here, we call such a solution optimal within the feasibility tolerance ε. Our cutting-plane algorithm is summarized as Algorithm 2.1; if the algorithm terminates, it proves the infeasibility of the problem (2.1) or yields an optimal solution with the feasibility tolerance ε.

Algorithm 2.1 Cutting-plane algorithm for solving MISDO Problems

Step 0 (Initialization) Let $\varepsilon \geq 0$ be a tolerance for feasibility. Define the initial feasible region as $\mathcal{F}_1 := \mathcal{Y}$. Set $t \leftarrow 1$.

Step 1 (Relaxed Problem) Solve Problem (2.2). Let $\boldsymbol{y}_t$ be an optimal solution.

Step 2 (Termination Condition)

- **(a)** If $\mathcal{F}_t = \emptyset$, then the problem (2.1) is infeasible.
- **(b)** If $\lambda_{\min}(\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}_t)) \geq -\varepsilon$, then $\boldsymbol{y}_t$ is an optimal solution within the feasibility tolerance ε.

Step 3 (Cut Generation) Update the feasible region,

$$\mathcal{F}_{t+1} \leftarrow \mathcal{F}_t \cap \left\{ \boldsymbol{y} \in \{0,1\}^{M_{\mathrm{b}}} \times \mathbb{R}^{M_{\mathrm{c}}} \mid \text{Eq.}(2.3) \right\}.$$

Step 4 Set $t \leftarrow t+1$ and return to Step 1.

Following the approach used in the case of solving SDO problems [97, 98], we prove convergence of the algorithm.

Theorem 2.2. *For any $\varepsilon > 0$, Algorithm 2.1 terminates in a finite number of iterations.*

Proof. Suppose that the algorithm does not terminate. The infinite sequence $\{(\boldsymbol{d}_t, \boldsymbol{y}_t)\}$ generated by the algorithm is bounded because $\|\boldsymbol{d}_t\| = 1$ and $\boldsymbol{y}_t \in \mathcal{Y}$ for all t. Therefore, we can choose a subsequence $\{(\boldsymbol{d}_t, \boldsymbol{y}_t) \mid t \in \mathcal{T}\}$ that converges to an accumulation point $(\bar{\boldsymbol{d}}, \bar{\boldsymbol{y}})$.

Since the termination condition is never satisfied, we have that for all $t \in \mathcal{T}$,

$$\boldsymbol{d}_t^\top (\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}_t))\boldsymbol{d}_t = \lambda_{\min}(\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}_t)) < -\varepsilon.$$

It follows that

$$\lim_{t\to\infty\,(t\in\mathcal{T})} (\boldsymbol{d}_t)^\top (\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}_t))\boldsymbol{d}_t = \bar{\boldsymbol{d}}^\top (\boldsymbol{C} - \mathcal{A}^\top(\bar{\boldsymbol{y}}))\bar{\boldsymbol{d}} \leq -\varepsilon. \tag{2.5}$$

For each $t < \ell$, $\boldsymbol{y}_\ell$ satisfies the tth cutting plane

$$\boldsymbol{d}_t^\top (\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}_\ell))\boldsymbol{d}_t \geq 0.$$

It follows that

$$\lim_{t\to\infty\,(t\in\mathcal{T})} \lim_{\ell\to\infty\,(\ell\in\mathcal{T})} \boldsymbol{d}_t^\top (\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}_\ell))\boldsymbol{d}_t = \bar{\boldsymbol{d}}^\top (\boldsymbol{C} - \mathcal{A}^\top(\bar{\boldsymbol{y}}))\bar{\boldsymbol{d}} \geq 0. \tag{2.6}$$

Eqs. (2.5) and (2.6) imply that $\varepsilon = 0$, which contradicts the hypothesis. □

Theorem 2.3. *Suppose that $\varepsilon = 0$. Even if Algorithm 2.1 does not terminate, every accumulation point of the sequence of solutions $\{\boldsymbol{y}_t\}$ is an optimal solution to the problem (2.1a)–(2.1c).*

Proof. Suppose that $\{(\boldsymbol{d}_t, \boldsymbol{y}_t) \mid t \in \mathcal{T}\}$ is a sequence that converges to an accumulation point $(\bar{\boldsymbol{d}}, \bar{\boldsymbol{y}})$. Note that $\lambda_{\min}(\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}_t))$ converges to $\lambda_{\min}(\boldsymbol{C} - \mathcal{A}^\top(\bar{\boldsymbol{y}}))$ as $t \to \infty$ $(t \in \mathcal{T})$ (see, e.g., Theorem 2.4.9.2 [84]). Due to Eq. (2.6), we have

$$\begin{aligned}
\lambda_{\min}(\boldsymbol{C} - \mathcal{A}^\top(\bar{\boldsymbol{y}})) &= \lim_{t\to\infty\,(t\in\mathcal{T})} \lambda_{\min}(\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}_t)) \\
&= \lim_{t\to\infty\,(t\in\mathcal{T})} (\boldsymbol{d}_t)^\top (\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y}_t))\boldsymbol{d}_t \\
&= \bar{\boldsymbol{d}}^\top (\boldsymbol{C} - \mathcal{A}^\top(\bar{\boldsymbol{y}}))\bar{\boldsymbol{d}} \geq 0,
\end{aligned}$$

which means that the matrix $\boldsymbol{C} - \mathcal{A}^\top(\bar{\boldsymbol{y}})$ is PSD. Since $\boldsymbol{b}^\top \boldsymbol{y}_t \geq \boldsymbol{b}^\top \boldsymbol{y}^\star$ for all t, we have that $\boldsymbol{b}^\top \bar{\boldsymbol{y}} \geq \boldsymbol{b}^\top \boldsymbol{y}^\star$. This implies that the accumulation point $\bar{\boldsymbol{y}}$ is an optimal solution to the problem (2.1). □

When all of the decision variables are binary, we can prove finite convergence of the algorithm even in the case $\varepsilon = 0$.

Theorem 2.4. *If $M_{\mathrm{c}} = 0$, Algorithm 2.1 terminates in a finite number of iterations.*

Proof. Recall that Step 3 excludes the solution $\boldsymbol{y}_t$ from the feasible region $\mathcal{F}_t$ in the tth iteration. Since the number of possible solutions $\boldsymbol{y} \in \{0,1\}^M$ is 2^M, the algorithm terminates with $\mathcal{F}_t = \emptyset$ after at most 2^M iterations. □

2.4 Branch-and-cut algorithm

Note that Step 1 of the cutting-plane algorithm solves an MILO problem at every iteration, and therefore we must execute the B&B algorithm many times from scratch. To resolve this issue, we develop a B&C algorithm that generates cutting planes dynamically during the B&B procedure. Such dynamic constraint generation is known as *lazy constraint callback* in the optimization literature. This functionality is offered by modern optimization software (e.g., CPLEX or Gurobi) as a state-of-the-art computational feature. Lazy constraints have been employed to speed up the computation for various applications, including harvest scheduling [159], energy supply systems [170], distribution network design [124], robust optimization [28], and statistical model selection [31]. To the best of our

knowledge, however, we are the first to use this approach to dealing with PSD constraints.

Our method starts with the B&B algorithm for solving the relaxed MILO problem (2.2). Once a feasible solution $\hat{\boldsymbol{y}} \in \mathcal{Y}$ is found, the callback procedure is initiated; if the PSD constraint (2.1b) is violated by $\hat{\boldsymbol{y}}$, a cutting plane is appended to exclude $\hat{\boldsymbol{y}}$ from the feasible region. After that, we proceed with the B&B algorithm from the middle of the MILO computation. We continue this process until the optimality of the obtained solution is proved through the B&B procedure. This B&C algorithm is summarized as Algorithm 2.2. Although our algorithms are designed for solving the problem (2.1) of the dual form, they can readily be modified to solve the primal standard form.

Algorithm 2.2 B&C algorithm for solving MISDO problem (2.1)

Step 0 (Initialization) Let $\varepsilon \geq 0$ be a tolerance for feasibility. Set the feasible region as $\mathcal{F} := \mathcal{Y}$.

Step 1 (B&B Procedure) Start (or continue) the B&B algorithm for solving the MILO problem,

$$\underset{\boldsymbol{y}}{\text{maximize}} \quad \boldsymbol{b}^\top \boldsymbol{y} \quad \text{subject to} \quad \boldsymbol{y} \in \mathcal{F}.$$

Step 2 (Callback Procedure) Once a feasible solution $\hat{\boldsymbol{y}} \in \mathcal{F}$ is found, go to the following steps:

Step 2.1 (Cut Generation) If $\lambda_{\min}(\boldsymbol{C} - \mathcal{A}^\top(\hat{\boldsymbol{y}})) < -\varepsilon$, update the feasible region,

$$\mathcal{F} \leftarrow \mathcal{F} \cap \left\{ \boldsymbol{y} \in \{0,1\}^{M_\mathrm{b}} \times \mathbb{R}^{M_\mathrm{c}} \mid \hat{\boldsymbol{d}}^\top (\boldsymbol{C} - \mathcal{A}^\top(\boldsymbol{y})) \hat{\boldsymbol{d}} \geq 0 \right\},$$

where $\hat{\boldsymbol{d}}$ is the normalized eigenvector of $\boldsymbol{C} - \mathcal{A}^\top(\hat{\boldsymbol{y}})$ corresponding to the minimum eigenvalue.

Step 2.2 Return to Step 1.

2.5 Numerical experiments

We performed several computational experiments to evaluate the effectiveness of our algorithms for solving MISDO problems.

2.5.1 Experimental design

We compare the computational performance of the following methods:

SCIP: MISDO solver SCIP-SDP[1] [62],

CUTSDP: MISDO solver CUTSDP implemented in YALMIP[2] [106],

CPA: cutting-plane algorithm (Algorithm 2.1),

B&C: B&C algorithm (Algorithm 2.2).

[1] http://www.opt.tu-darmstadt.de/scipsdp/
[2] https://yalmip.github.io/solver/cutsdp/

We also solved some instances by Pajarito [45] to evaluate its performance. Our preliminary experiments showed that Pajarito was very slow and failed to complete the computations even for small-sized instances. Therefore, we omitted Pajarito from our target solvers for the comparison in the following numerical experiments.

We used SCIP-SDP 3.1.0 on the NEOS Server[3] [46], where the B&B framework of SCIP[4] 5.0.0 [63] and the SDO solver DSDP[5] 5.8 [21] were combined in the default configuration. When we ran CUTSDP, we used the MATLAB R2015 b and Gurobi Optimizer[6] 8.0 to solve relaxed linear optimization problems and MILO problems. CPA and B&C were implemented in the Python language; MILO problems were solved using the Gurobi Optimizer 8.0, and its callback function was employed for Step 2 of Algorithm 2.2. The tolerance for feasibility was set as $\varepsilon := 10^{-5} \cdot (1 + \|\boldsymbol{b}\|_1)$ on the basis of the DIMACS error [117], where $\boldsymbol{b}$ is the coefficient vector of the objective function. These CUTSDP, CPA, and B&C computations were performed on a Windows 7 PC with an Intel Core i7-4790 CPU (3.60 GHz) and 16 GB of memory. The computation of each method was terminated if it did not finish by itself within 7200 s. In these cases, the results obtained at 7200 s were taken as the final outcome.

The column labels used in the tables of experimental results are defined as follows.

Time: computation time in seconds,

#Abort: number of times that the computation was aborted by a numerical issue,

#Limit: number of times that the computation reached the time limit of 7200 s,

#Cuts: number of cutting planes generated by CUTSDP, CPA, and B&C,

#Nodes: number of nodes in the enumeration tree formed by SCIP and B&C.

Following Gally et al. [62], we use the shifted geometric mean to aggregate results of random instances. The shifted geometric mean of values $x_1, x_2, \ldots, x_N$ is defined as

$$\left(\prod_{n \in [N]} (x_n + s) \right)^{1/N} - s,$$

where the shift s was set to 10, 1000, and 100 for "Time," "#Cuts," and "#Nodes," respectively, as in Gally et al. [62].

2.5.2 Random instances

In this subsection, we report experimental results for solving random instances of the problem (2.1a)–(2.1c).

Instance generation

Following Yamashita et al. [169], we generated random instances of the problem (2.1). First, we constructed $\tilde{\boldsymbol{y}}$ as a feasible solution: $\tilde{y}_m$ was randomly chosen

[3] https://neos-server.org/neos/
[4] http://scip.zib.de/
[5] http://www.mcs.anl.gov/hs/software/DSDP/
[6] http://www.gurobi.com/

from $\{0,1\}$ for $m \leq M_{\mathrm{b}}$ and was drawn from a uniform distribution on the interval $[0,1]$ for $m \geq M_{\mathrm{b}}+1$. The elements of each of the matrices $\boldsymbol{A}_m$ $(m \in [M])$ were drawn from a uniform distribution on the interval $[-1,1]$. We next set $\boldsymbol{C} := \alpha \boldsymbol{I} + \sum_{m\in[M]} \boldsymbol{A}_m \tilde{y}_m$, where α is a positive-valued parameter ensuring that the matrix $\boldsymbol{C} - \mathcal{A}^\top(\tilde{\boldsymbol{y}})$ is PSD. We then set $b_m := \boldsymbol{A}_m \bullet \boldsymbol{I}$ for $m \in [M]$.

Experimental results

Table 2.1 gives the computational results for the case of the random instances. Five instances of the problem (2.1a)–(2.1c) were generated for each tuple $(N, M_{\mathrm{b}}, M_{\mathrm{c}}, \alpha)$, and the shifted geometric means of "Time," "#Cuts," and "#Nodes" were computed.

We can see from Table 2.1 that our algorithm B&C was always the fastest among the four methods. The number of generated nodes in an enumeration tree was much larger for B&C than for SCIP; however, the series of continuous relaxation problems at such nodes can be solved efficiently in the case of B&C computation. CPA was often slower than SCIP, but it is noteworthy that the SCIP computations were aborted in some instances by numerical issues. In contrast to SCIP, our algorithms do not handle SDO problems, so they can solve MISDO problems without encountering numerical instability. A typical example is the case of $(N, M_{\mathrm{b}}, M_{\mathrm{c}}, \alpha) = (30, 30, 60, 1.0)$: the computations of CPA and B&C finished on average in 2069.8 s and 18.8 s, respectively, but the SCIP computations were aborted in all five instances.

Table 2.1 showed that B&C was always faster than CUTSDP. The main difference between B&C and CUTSDP is that CUTSDP does not implement the lazy constraints callback function. Thus, CUTSDP does not employ a warm-starting strategy unlike B&C, and this is considered to be the reason why CUTSDP was not efficient compared with our B&C algorithm. In addition, comparing CPA and CUTSDP, CPA was faster than CUTSDP except for the case of $(N, M_{\mathrm{b}}, N_{\mathrm{c}}, \alpha) = (15, 30, 30, 1.0)$. While CPA solves the relaxed MILOs from the beginning, CUTSDP first solves a sequence of relaxed continuous linear optimization problems iteratively with adding cuts at each iteration. Then, CUTSDP keeps the added constraints and moves into the next phase, imposing integer constraints and solving MILOs iteratively as our CPA does. The results that CUTSDP was less efficient than CPA suggest that CUTSDP would add too many constraints during the first phase and make solving the relaxed MILOs hard in the second phase.

2.5.3 Computing restricted isometry constants

Here, we compare the performance of the algorithms for the problem of computing restricted isometry constants [61].

Problem formulation

Let us consider a system of linear equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where the coefficient matrix $\boldsymbol{A} \in \mathbb{R}^{\mu\times N}$ and the right-hand side vector $\boldsymbol{b} \in \mathbb{R}^{\mu}$ are given. Compressed sensing [13] focuses on finding the sparsest solution $\hat{\boldsymbol{x}}^0$ to an undetermined linear system with $\mu \leq N$:

$$\hat{\boldsymbol{x}}^0 := \arg\min\{\|\boldsymbol{x}\|_0 \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\},$$

where $\|\boldsymbol{x}\|_0$ denotes the number of nonzero elements of vector $\boldsymbol{x} \in \mathbb{R}^N$. This problem is known to be NP-hard [13].

Table 2.1: Results for random instances of the problem (2.1a)–(2.1c)

N	M_b	M_c	α	Method	Time	#Abort	#Limit	#Cuts	#Nodes
30	30	30	1.0	SCIP	61.5	0	0	—	25.4
				CUTSDP	378.4	0	0	1541.2	—
				CPA	131.0	0	0	543.6	—
				B&C	4.2	0	0	556.2	1071.7
15	30	30	1.0	SCIP	35.9	1	0	—	63.4
				CUTSDP	253.3	0	0	1279.7	—
				CPA	315.8	0	0	524.8	—
				B&C	4.4	0	0	770.8	5270.1
60	30	30	1.0	SCIP	142.2	1	0	—	23.0
				CUTSDP	1086.7	0	0	2235.7	—
				CPA	186.3	0	0	633.1	—
				B&C	4.7	0	0	628.8	1010.8
30	15	30	1.0	SCIP	34.6	2	0	—	15.7
				CUTSDP	158.1	0	0	1478.7	—
				CPA	78.0	0	0	540.3	—
				B&C	2.1	0	0	546.6	629.2
30	60	30	1.0	SCIP	128.7	0	0	—	51.6
				CUTSDP	978.8	0	0	1624.6	—
				CPA	439.8	0	0	553.8	—
				B&C	5.3	0	0	584.2	5183.2
30	30	15	1.0	SCIP	30.6	0	0	—	24.1
				CUTSDP	83.5	0	0	690.7	—
				CPA	14.4	0	0	185.7	—
				B&C	0.7	0	0	199.1	446.6
30	30	60	1.0	SCIP	—	5	—	—	—
				CUTSDP	2637.0	0	0	3858.8	—
				CPA	2069.8	0	0	1559.3	—
				B&C	18.8	0	0	1617.3	2977.7
30	30	30	0.1	SCIP	22.8	3	0	—	1.0
				CUTSDP	282.5	0	0	1273.0	—
				CPA	145.1	0	0	445.7	—
				B&C	2.0	0	0	431.1	474.2
30	30	30	10.0	SCIP	75.3	2	0	—	47.2
				CUTSDP	144.0	0	0	995.1	—
				CPA	97.8	0	0	401.8	—
				B&C	55.1	0	0	3947.6	13111.2
45	45	45	1	SCIP	164.8	0	0	—	38.8
				CUTSDP	3046.6	0	0	3107.7	—
				CPA	1123.8	0	0	1066.5	—
				B&C	11.8	0	0	1112.2	2403.8
60	60	60	1	SCIP	558.0	0	0	—	50.0
				CUTSDP	7169.6	0	4	4468.7	—
				CPA	5069.8	0	0	1644.0	—
				B&C	31.0	0	0	1663.5	4125.5

The lower and upper restricted isometry constants (RICs) [55] of order κ are defined as

$$\alpha_\kappa := \min\{\|\boldsymbol{A}\boldsymbol{x}\|_2^2 \mid \|\boldsymbol{x}\|_2^2 = 1,\ \|\boldsymbol{x}\|_0 \leq \kappa\}, \tag{2.7}$$

$$\beta_\kappa := \max\{\|\boldsymbol{A}\boldsymbol{x}\|_2^2 \mid \|\boldsymbol{x}\|_2^2 = 1,\ \|\boldsymbol{x}\|_0 \leq \kappa\}. \tag{2.8}$$

When certain conditions on RICs are fulfilled, the sparsest solution $\hat{\boldsymbol{x}}^0$ coincides with the following ℓ_1-norm minimizer [55]:

$$\hat{\boldsymbol{x}}^1 := \arg\min\{\|\boldsymbol{x}\|_1 \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\}.$$

Gally and Pfetsch [61] proved that exact values of α_κ and β_κ can be computed by solving the following MISDO problem:

$$\begin{aligned}
\underset{\boldsymbol{X},\boldsymbol{z}}{\text{minimize/maximize}} \quad & \mathrm{Tr}(\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{X}) && \text{(2.9a)}\\
\text{subject to} \quad & \mathrm{Tr}(\boldsymbol{X}) = 1, && \text{(2.9b)}\\
& \sum_{n\in[N]} z_n \leq \kappa, && \text{(2.9c)}\\
& -\frac{1}{2} z_n \leq X_{nm} \leq \frac{1}{2} z_n, \quad (\forall (n,m) \in [N]\times[N]), && \text{(2.9d)}\\
& \boldsymbol{X} \succeq \boldsymbol{O}, && \text{(2.9e)}\\
& \boldsymbol{z} \in \{0,1\}^N, && \text{(2.9f)}
\end{aligned}$$

where $\boldsymbol{X} := (X_{nm}) \in \mathcal{S}^N$ and $\boldsymbol{z} := (z_n) \in \{0,1\}^N$ are decision variables.

Experimental results

Tables 2.2 and 2.3 give the results for computing lower and upper RICs, respectively. The size of the problem (2.9) is represented by $N \in \{10, 15, 20, 25\}$, and the order of RICs is $\kappa \in \{3, 5\}$. Note that the number of linear equations in the system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ was fixed to $\mu = 10$ because it is independent of the size of the problem (2.9). As in the case of Gaussian-distributed matrices [61], each element of matrix $\boldsymbol{A}$ was drawn from the standard normal distribution. Five instances of matrix $\boldsymbol{A}$ were generated for each pair (N, κ), and the shifted geometric means of "Time," "#Cuts," and "#Nodes" were computed. In Tables 2.2 and 2.3, each column "M_b" and "M_c" are the number of integer variables and that of continuous ones, respectively.

We can see from Tables 2.2 and 2.3 that, as was the case for random instances, B&C was the fastest among the four methods. CPA was also faster than SCIP and CUTSDP in the case of computing RICs. In addition, we notice that when the problem size (i.e., N) was large, the SCIP computation was frequently aborted due to a numerical issue. Indeed, when $N \geq 20$, SCIP failed to complete the computation for 19 out of 20 instances of computing lower RICs (Table 2.2) and for 11 out of 20 instances of computing upper RICs (Table 2.3). Also, CUTSDP did not solve all instances of computing lower RICs (Table 2.2) and upper RICs (Table 2.3) within the time limit of 7200 s for $N \geq 25$.

2.5.4 Robust truss topology design

In this subsection, we report the experimental results for our tests of the three algorithms on the problem of robust truss topology design [171].

Table 2.2: Results for computing lower RICs ($\mu = 10$)

N	κ	M_{b}	M_{c}	Method	Time	#Abort	#Limit	#Cuts	#Nodes
10	3	45	15	SCIP	6.1	0	0	—	27.6
				CUTSDP	21.9	0	0	730.5	—
				CPA	1.7	0	0	55.4	—
				B&C	1.0	0	0	479.2	751.6
10	5	45	15	SCIP	16.6	2	0	—	91.4
				CUTSDP	50.7	0	0	945.9	—
				CPA	14.0	0	0	183.1	—
				B&C	2.3	0	0	1087.0	2827.2
15	3	105	15	SCIP	140.3	0	0	—	187.6
				CUTSDP	2468.1	0	1	4369.6	—
				CPA	11.5	0	0	107.6	—
				B&C	3.8	0	0	943.5	1864.8
15	5	105	15	SCIP	693.4	0	0	—	891.4
				CUTSDP	7200.0	0	5	4296.1	—
				CPA	237.9	0	0	342.4	—
				B&C	14.1	0	0	1718.1	6494.2
20	3	190	20	SCIP	1037.6	4	0	—	373.0
				CUTSDP	7200.0	0	5	5478.4	—
				CPA	58.7	0	0	191.7	—
				B&C	8.9	0	0	1155.7	3031.0
20	5	190	20	SCIP	—	5	—	—	—
				CUTSDP	7200.0	0	5	5153.0	—
				CPA	1938.7	0	0	669.5	—
				B&C	112.5	0	0	533.3	21239.5
25	3	300	25	SCIP	—	5	—	—	—
				CUTSDP	7200.0	0	5	5149.2	—
				CPA	188.4	0	0	294.4	—
				B&C	16.9	0	0	1407.6	5435.7
25	5	300	25	SCIP	—	5	—	—	—
				CUTSDP	7200.0	0	5	5332.3	—
				CPA	7200.0	0	5	1178.9	—
				B&C	373.0	0	0	1008.0	60996.7
30	3	435	30	SCIP	—	5	—	—	—
				CUTSDP	7200.0	0	5	5116.3	—
				CPA	616.48	0	0	449.1	—
				B&C	45.33	0	0	1123.2	9828.7
30	5	435	30	SCIP	—	5	—	—	—
				CUTSDP	7200.0	0	5	5619.5	—
				CPA	7200.0	0	5	2104.3	—
				B&C	1343.6	0	0	2272.5	136486.1
35	3	595	35	SCIP	—	5	—	—	—
				CUTSDP	7200.0	0	5	6635.7	—
				CPA	2257.4	0	0	652.3	—
				B&C	102.1	0	0	609.7	16269.0
35	5	595	35	SCIP	—	5	—	—	—
				CUTSDP	7200.0	0	5	6696.3	—
				CPA	7200.0	0	5	2776.7	—
				B&C	3802.6	0	0	3336.7	225854.9

Table 2.3: Results for computing upper RICs ($\mu = 10$)

N	κ	M_{b}	M_{c}	Method	Time	#Abort	#Limit	#Cuts	#Nodes
10	3	10	45	SCIP	5.4	0	0	—	26.3
				CUTSDP	13.5	0	0	416.7	—
				CPA	1.5	0	0	30.2	—
				B&C	0.6	0	0	308.4	429.1
10	5	10	45	SCIP	9.7	0	0	—	74.3
				CUTSDP	53.9	0	0	826.5	—
				CPA	4.8	0	0	77.8	—
				B&C	2.0	0	0	880.4	2233.6
15	3	15	105	SCIP	41.3	0	0	—	51.7
				CUTSDP	231.3	0	0	1091.7	—
				CPA	3.8	0	0	34.2	—
				B&C	2.7	0	0	683.0	1465.0
15	5	15	105	SCIP	54.5	0	0	—	76.6
				CUTSDP	2657.9	0	3	2028.3	—
				CPA	22.8	0	0	93.7	—
				B&C	6.2	0	0	1084.2	2608.2
20	3	20	190	SCIP	322.0	0	0	—	93.1
				CUTSDP	1479.7	0	1	1177.4	—
				CPA	8.1	0	0	34.8	—
				B&C	3.8	0	0	511.9	1662.8
20	5	20	190	SCIP	605.9	2	0	—	130.8
				CUTSDP	3651.5	0	3	922.2	—
				CPA	129.2	0	0	121.9	—
				B&C	14.6	0	0	997.3	6100.2
25	3	25	300	SCIP	1224.0	4	0	—	137.0
				CUTSDP	7200.0	0	5	1072.6	—
				CPA	23.9	0	0	46.4	—
				B&C	7.9	0	0	681.2	3211.8
25	5	25	300	SCIP	—	5	—	—	—
				CUTSDP	7200.0	0	5	1106.3	—
				CPA	1285.2	0	0	165.4	—
				B&C	68.9	0	0	922.4	13854.4
30	3	30	435	SCIP	—	5	—	—	—
				CUTSDP	7200.0	0	5	1185.4	—
				CPA	40.8	0	0	44.6	—
				B&C	14.7	0	0	654.5	4552.8
30	5	30	435	SCIP	—	5	—	—	—
				CUTSDP	7200.0	0	5	1424.9	—
				CPA	2380.4	0	0	168.9	—
				B&C	177.9	0	0	1199.2	24939.4
35	3	35	595	SCIP	—	5	—	—	—
				CUTSDP	7200.0	0	5	1519.3	—
				CPA	92.4	0	0	46.8	—
				B&C	26.5	0	0	301.3	5424.4
35	5	35	595	SCIP	—	5	—	—	—
				CUTSDP	7200.0	0	5	1887.3	—
				CPA	6426.1	0	4	158.4	—
				B&C	318.0	0	0	938.1	56816.4

Problem formulation

Let us consider an undirected graph called the *ground structure* in d-dimensional space. We suppose that some nodes are fixed, and d-dimensional forces act on the q unfixed nodes; so the external load vector is defined as $\boldsymbol{f} \in \mathbb{R}^{\nu}$ with $\nu := d \cdot q$. The index set in the external load vector is denoted by $\mathcal{J} := \{1, 2, \ldots, \nu\}$. The index set of edges in the ground structure is denoted by $\mathcal{I} := \{1, 2, \ldots, \mu\}$, which represents candidate members (or bars) of a truss. Truss topology design involves choosing a subset of members and determining their cross-sectional areas so that the truss structure will be lightweight but sufficiently rigid.

We begin by introducing the decision variables. We denote by $\boldsymbol{a} := (a_i)_{i \in \mathcal{I}} \in \mathbb{R}^{\mu}$ the vector of member cross-sectional areas. Additionally, the vector $\boldsymbol{p} := (p_j)_{j \in \mathcal{J}} \in \{0, 1\}^{\nu}$ corresponds to the existence of each node, and the vector $\boldsymbol{t} := (t_i)_{i \in \mathcal{I}} \in \{0, 1\}^{\mu}$ represents the selection of members in the truss structure.

Given matrices $\boldsymbol{K}_i \in \mathcal{S}^{\nu}$ $(i \in \mathcal{I})$, the stiffness matrix is defined as $\boldsymbol{K}(\boldsymbol{a}) := \sum_{i \in \mathcal{I}} a_i \boldsymbol{K}_i$. The following matrix is used to express uncertainty about the external loads applied at each node in the truss structure:

$$\boldsymbol{Q} := (\tilde{\boldsymbol{f}}, r\boldsymbol{v}_1, r\boldsymbol{v}_2, \ldots, r\boldsymbol{v}_{\nu-1}) \in \mathbb{R}^{\nu \times \nu},$$

where $\tilde{\boldsymbol{f}} \in \mathbb{R}^{\nu}$ is the nominal external load, r is the level of uncertainty, and $\boldsymbol{v}_j \in \mathbb{R}^{\nu}$ $(j = 1, 2, \ldots, \nu - 1)$ are orthonormal basis vectors of the orthogonal complement of $\tilde{\boldsymbol{f}}$. When an ellipsoidal uncertainty set of external loads [19] is employed, the worst-case compliance constraint is posed as

$$\begin{pmatrix} \bar{\tau}\boldsymbol{I} & (\mathrm{Diag}(\boldsymbol{p})\boldsymbol{Q})^{\top} \\ \mathrm{Diag}(\boldsymbol{p})\boldsymbol{Q} & \boldsymbol{K}(\boldsymbol{a}) \end{pmatrix} \succeq \boldsymbol{O}, \tag{2.10}$$

where $\bar{\tau}$ is the upper-bound parameter for the compliance; see Yonekura and Kanno [171] for the details.

We define $\mathcal{J}_{\mathrm{f}} \subseteq \mathcal{J}$ to represent the nodes that should remain in the truss structure. Other nodes can be removed, but we must maintain the nodes that are connected to at least one member. Such constraints can be expressed as

$$0 \leq p_j \leq 1, \quad (\forall j \in \mathcal{J} \setminus \mathcal{J}_{\mathrm{f}}), \tag{2.11}$$

$$t_i \leq p_j, \quad (\forall i \in \mathcal{I}_{\hat{k}(j)}, \forall j \in \mathcal{J} \setminus \mathcal{J}_{\mathrm{f}}), \tag{2.12}$$

$$p_j = 1, \quad (\forall j \in \mathcal{J}_{\mathrm{f}}), \tag{2.13}$$

where $\mathcal{I}_{\hat{k}(j)} \subseteq \mathcal{I}$ is the set of members that are connected to the node associated with the jth element of $\tilde{\boldsymbol{f}}$.

The following constraints are imposed on the member cross-sectional areas:

$$a_{\min} t_i \leq a_i \leq a_{\max} t_i, \quad (\forall i \in \mathcal{I}), \tag{2.14}$$

$$t_i \in \{0, 1\}, \quad (\forall i \in \mathcal{I}), \tag{2.15}$$

where $a_{\min}$ and $a_{\max}$ are, respectively, lower and upper bounds on cross-sectional areas.

The robust truss topology design minimizing the structural volume is formulated as the following MISDO problem:

$$\underset{\boldsymbol{a},\, \boldsymbol{p},\, \boldsymbol{t}}{\text{minimize}} \quad \sum_{i \in \mathcal{I}} \ell_i a_i \tag{2.16a}$$

$$\text{subject to} \quad \text{Eqs. (2.10)–(2.15)}, \tag{2.16b}$$

where ℓ_i is the length of the ith member. Although $\boldsymbol{p}$ is treated as a vector of continuous variables, the binary constraint $\boldsymbol{p} \in \{0, 1\}^{\nu}$ can be satisfied.

Experimental results

We considered a two-dimensional lattice as the ground structure; nodes were placed on lattice points, and every pair of distinct nodes was connected by a unique edge. Note that the longer edge was removed if two edges overlapped along the same straight line, and the remaining edges were treated as candidate members.

As explained below, the settings of parameters were the same as those used by Yonekura and Kanno [171]. Specifically, the lengths of horizontal and vertical members were, respectively, 100 cm and 50 cm, the lower and upper bounds on cross-sectional areas were, respectively, $a_{\min} := 10\,\mathrm{cm}^2$ and $a_{\max} := 1000\,\mathrm{cm}^2$, and the elastic modulus was 200 GPa. The leftmost nodes of trusses were fixed, and the bottom-right node was loaded with a vertical force of 1 kN as a nominal external load $\tilde{\boldsymbol{f}}$. The level of uncertainty was $r := 0.1$, and the upper-bound parameter for compliance was $\bar{\tau} := 10\,\mathrm{kN}$. Note that all lengths were measured in centimeters, and that the value of each member length, ℓ_i, was divided by 1000 to reduce numerical error.

Table 2.4 gives the computational results for the robust truss topology design problem. In Tables 2.4, each column "N", "M_{b}", and "M_{c}" are the matrix size in Eq. (2.10), the number of integer variables, and that of continuous ones, respectively. For instance, Lat(3,3) is a problem instance corresponding to 3×3 lattice points. Also, 22mem and 51mem stand for the 22- and 51-member truss instances defined in Yonekura and Kanno [171]. The column labeled "Obj" shows the value of the objective function (2.16a). If the computation reached the time limit of 7200 s, the obtained lower and upper bounds on the objective function are shown.

We can see from Table 2.4 that there is no clear inferior-to-superior relationship between B&C and SCIP for small-sized instances. For example, B&C was three times faster than SCIP for the Lat(3,4) instance, whereas B&C was four times slower than SCIP for the 22mem instances. For large-sized instances, the computation time of B&C was relatively long. Indeed, B&C took much longer to solve the Lat(4,4) and 51mem instances than SCIP did. However, it is noteworthy that B&C successfully found solutions of good quality to such large-sized instances. For the Lat(4,4) instance, although the B&C computation was terminated due to the time limit, its computed upper-bound value (i.e., the objective value of the obtained feasible solution) was 71.01, which is very close to the optimal value (i.e., 69.34) obtained by SCIP. For the Lat(5,3) instance, B&C computed an upper-bound value of 294.49, which is very close to the optimal value (i.e., 292.68) provided by CPA and CUTSDP, whereas SCIP failed to solve this instance due to a numerical issue. On the other hand, CPA showed poorer computational performance than both SCIP and B&C for robust truss topology design.

Comparing B&C and CUTSDP, Table 2.4 showed that B&C was faster than CUSTDP except for the cases of the two instances, 22mem and Lat(5,3). For the 22mem instance, B&C was slower than CUTSDP, but the difference in the computational time between these methods was small. Moreover, as we mentioned before, B&C obtained a good upper-bound value for the Lat(5,3) instance. In terms of a comparison between CPA and CUTSDP, Table 2.4 showed that CPA was faster then CUTSDP except for the cases of the three instances, Lat(3,3), Lat(4,3) and 22mem. For each of the three instances, while CPA took longer to solve than CUTSDP did, the computational time of CPA and CUTSDP were comparable.

Table 2.4: Results for robust truss topology design

Instance	ν	μ	N	M_{b}	M_{c}	Method	Time	Obj	#Cuts	#Nodes
Lat(3,3)	12	26	24	26	38	SCIP	18.6	32.40	—	71
						CUTSDP	35.2	32.40	497	—
						CPA	55.1	32.40	347	—
						B&C	24.4	32.40	2612	17115
Lat(3,4)	16	46	32	46	65	SCIP	45.2	21.24	—	101
						CUTSDP	1590.3	21.24	764	—
						CPA	943.1	21.24	479	—
						B&C	12.9	21.24	1111	14594
Lat(4,3)	18	47	36	47	96	SCIP	326.1	114.07	—	755
						CUTSDP	629.8	114.07	1384	—
						CPA	741.3	114.07	982	—
						B&C	324.8	114.07	8855	85147
Lat(3,5)	20	70	40	70	90	SCIP	593.0	17.32	—	354
						CUTSDP	>7200	[15.92, —]	>962	—
						CPA	>7200	[16.14, —]	>433	—
						B&C	424.8	17.32	3902	149298
Lat(4,4)	24	83	48	83	107	SCIP	850.8	69.34	—	525
						CUTSDP	>7200	[64.82, —]	>1690	—
						CPA	>7200	[64.61, —]	>996	—
						B&C	>7200	[69.31, 71.01]	>10059	>423279
Lat(5,3)	24	72	48	72	96	SCIP	Abort	—	—	—
						CUTSDP	6852.7	292.68	2963	—
						CPA	6324.3	292.68	2127	—
						B&C	>7200	[292.67, 294.49]	>22398	>336055
22mem	12	22	24	22	34	SCIP	7.7	33.27	—	23
						CUTSDP	24.1	33.27	467	—
						CPA	39.2	33.27	321	—
						B&C	36.3	33.27	7507	15343
51mem	18	51	36	51	69	SCIP	336.5	75.05	—	236
						CUTSDP	>7200	[74.95, —]	>1999	—
						CPA	>7200	[74.86, —]	>1375	—
						B&C	4308.7	75.05	17680	267973

2.6 Conclusion

In this chapter, we have described general-purpose algorithms to solve such problems. Firstly, we formulated a cutting-plane algorithm to solve MISDO problems and proved its convergence properties. We then developed a B&C algorithm, where cutting planes are added dynamically to the relaxed MILO problem during a B&B procedure. The experimental results show that our B&C algorithm was effective in solving three different MISDO problems: random instances, computing restricted isometry constants, and robust truss topology design. Our algorithm is based on the B&B procedure for solving MILO problems, so it can use a warm-starting strategy to solve a series of continuous relaxation problems efficiently. Moreover, our algorithms do not solve any MISDO problems, which makes their computation very stable and practical.

Chapter 3

Cutting-plane Algorithm for Best Subset Selection for Eliminating Multicollinearity

In this chapter, we study the best subset selection problem for eliminating multicollinearity from linear regression models. Specifically, we focus on the problem of selecting the best subset of explanatory variables subject to an upper bound on the condition number of the correlation matrix of the selected variables. First, we formulate this problem as an MISDO problem. Second, we develop a specialized cutting-plane algorithm that, to approximate the condition number constraint, iteratively appends cutting planes to the mixed-integer quadratic optimization problem. Here, to improve the computational efficiency of our algorithm, we propose to generate strong cutting planes by effectively using a heuristic search. The content of this chapter is included in Tamura *et al.* [151].

We give an introduction to this study and summarize our contribution in Section 3.1. In Section 3.2, we give a brief review of linear regression and subset selection for eliminating multicollinearity. In Section 3.3, we formulate the best subset selection under the condition number constraint as an MISDO problem. In Section 3.4, we describe our specialized cutting-plane algorithm for solving the best subset selection problem. In Section 3.5, we show the computational results, and in Section 3.6, we conclude with a brief summary.

3.1 Introduction

Multicollinearity, which exists when two or more explanatory variables in a regression model are highly correlated, is a frequently encountered problem in multiple regression analysis [54, 76, 112]. Such an interrelationship among explanatory variables obscures their relationship with the explained variable, leading to computational instability in model estimation. Moreover, the reliability of the regression analysis is decreased in the presence of multicollinearity by the low quality of the resultant estimates.

Several approaches can be used to avoid the deleterious effects of multicollinearity [42]. One approach is orthogonal transformation through procedures such as principal component regression [88, 115] and partial least squares regression [164, 165]. In this approach, a set of correlated variables is transformed into a set of linearly uncorrelated variables (i.e., principal components) for use in a regression model. Orthogonal transformation can enhance the computational stability of model estimation but often leads to worse predictive performance and results that are strongly influenced by the presence of outliers [58, 78].

Another approach is penalized regression, such as ridge regression [83], lasso [154], and elastic net [174]. This approach introduces a penalty function to shrink regression coefficient estimates toward zero. Penalized regression helps prevent

regression models from overfitting noisy datasets and, accordingly, is effective for achieving high predictive performance. However, the penalty functions produce biased estimates, which are undesirable from the standpoint of model interpretation [30, 32].

In this chapter, we focus on subset selection, which is a simple but effective approach for eliminating multicollinearity. Conventionally in this approach, explanatory variables are deleted iteratively through the use of indicators for detecting multicollinearity, such as condition number of the correlation matrix and variance inflation factor [42]. Chong and Jun [44] have compared the performance of subset selection methods with that of partial least squares regression and suggested that a goodness-of-fit measure for evaluating a subset regression model should be chosen carefully when multicollinearity is present. For subset selection in the presence of multicollinearity, several researchers [49, 87] have proposed goodness-of-fit measures based on ridge regression. It is notable, however, that commonly used subset selection methods are heuristic algorithms, such as stepwise regression [50]; hence, they do not necessarily find the best subset of variables in terms of a given goodness-of-fit measure.

The mixed-integer optimization (MIO) approach to subset selection has recently received much attention due to advances in algorithms and hardware [30, 32, 92, 99, 101, 118, 119, 140, 141]. In contrast to heuristic algorithms, the MIO approach has the potential to provide the best subset of variables under a given goodness-of-fit measure. To deal with multicollinearity, Bertsimas and King [30] use a cutting-plane strategy, which iteratively adds constraints for deleting subsets of collinear variables. However, this strategy requires solving an enormous number of mixed-integer quadratic optimization (MIQO) problems when multicollinearity exists in many different sets of variables.

The aim of this chapter is to devise a more sophisticated MIO approach to best subset selection for eliminating multicollinearity. In particular, this chapter addresses the following problem: Find a subset of variables that minimizes the residual sum of squares under the constraint that the condition number of the associated correlation matrix is bounded. We first formulate this subset selection problem as an MISDO problem. Furthermore, to increase computational efficiency, we consider incorporating constraints based on the normal equations into the MISDO problem. Regrettably, as we show later, current general-purpose MISDO solvers do not deliver high computational performance and the problem size to be handled is limited. Thus, we secondly propose to develop a specialized cutting-plane algorithm for subset selection that works well in practice. Theoretically, this cutting-plane algorithm must solve an exponential number of MIQO problems, which are NP-hard, in a worst-case situation. However, it has a practical advantage that we can use sophisticated MIO solvers such as Gurobi and CPLEX to solve MIQO problems. In addition, we propose to use a backward elimination method that searches a smaller subset of collinear variables to generate strong cutting planes, which improves the efficiency of the cutting-plane algorithm.

The effectiveness of our MIO approaches is assessed through computational experiments using several datasets from the UCI Machine Learning Repository [109]. We succeeded in solving some of our MISDO problems for subset selection when the number of candidate explanatory variables was less than 26. In addition, the computational results demonstrate that our specialized cutting-plane algorithm yielded optimal solutions when the number of variables was 65, and it frequently gave a better subset of variables than did the conventional local search algorithms for large-sized datasets.

3.2 Linear regression and multicollinearity

This section contains a brief review of linear regression and subset selection for eliminating multicollinearity.

3.2.1 Linear regression

Let us suppose that we are given N samples, $(\boldsymbol{x}_n, y_n)$ for $n \in [N]$. Here, $\boldsymbol{x}_n := (x_{n1}, x_{n2}, \ldots, x_{nP})^\top$ is a vector composed of P explanatory variables, and y_n is an explained variable for each sample $n \in [N]$.

We focus on the following linear regression model:

$$y_n = \boldsymbol{a}^\top \boldsymbol{x}_n + \varepsilon_n = \sum_{p \in [P]} a_p x_{np} + \varepsilon_n \quad (\forall n \in [N]),$$

where $\boldsymbol{a} := (a_1, a_2, \ldots, a_P)^\top$ is a vector of regression coefficients to be estimated and ε_i is a prediction residual for each sample $n \in [N]$. We assume here that all explanatory and explained variables are standardized so that $(\sum_{n \in [N]} x_{np})/N = (\sum_{n \in [N]} y_n)/N = 0$ and $(\sum_{n \in [N]} (x_{np})^2)/N = (\sum_{n \in [N]} (y_n)^2)/N = 1$ for all $p \in [P]$. Therefore, no intercept (constant term) is present in the regression model.

The above linear regression model can be rewritten as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{a} + \boldsymbol{\varepsilon},$$

where $\boldsymbol{y} := (y_1, y_2, \ldots, y_N)^\top$, $\boldsymbol{\varepsilon} := (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_N)^\top$, and

$$\boldsymbol{X} := \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1P} \\ x_{21} & x_{22} & \cdots & x_{2P} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{NP} \end{pmatrix}.$$

To estimate the regression coefficients, $\boldsymbol{a}$, the ordinary least squares method minimizes the residual sum of squares (RSS):

$$\boldsymbol{\varepsilon}^\top \boldsymbol{\varepsilon} = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a}). \tag{3.1}$$

After partial differentiation, the ordinary least squares method is equivalent to solving a system of linear equations:

$$\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{a} = \boldsymbol{X}^\top \boldsymbol{y}. \tag{3.2}$$

This is the well-known normal equation.

3.2.2 Subset selection for eliminating multicollinearity

We use the condition number of the correlation matrix $\boldsymbol{R} := (r_{pq}) \in \mathcal{S}^P$ to detect multicollinearity. Because the explanatory variables are standardized, the correlation matrix is calculated as

$$\boldsymbol{R} = \frac{\boldsymbol{X}^\top \boldsymbol{X}}{N},$$

and its condition number is defined as

$$\mathrm{cond}(\boldsymbol{R}) := \begin{cases} \dfrac{\lambda_{\max}(\boldsymbol{R})}{\lambda_{\min}(\boldsymbol{R})} & (\lambda_{\min}(\boldsymbol{R}) > 0), \\ +\infty & (\lambda_{\min}(\boldsymbol{R}) = 0), \end{cases} \tag{3.3}$$

where $\lambda_{\min}(\boldsymbol{R})$ and $\lambda_{\max}(\boldsymbol{R})$ are the minimum and maximum eigenvalues of matrix $\boldsymbol{R}$, respectively. It follows from $\mathrm{cond}(\boldsymbol{R}) = \mathrm{cond}(\boldsymbol{X}^\top \boldsymbol{X})$ that when $\mathrm{cond}(\boldsymbol{R})$ is very large, the coefficient matrix of equations (3.2) is ill-conditioned, which implies that the regression coefficient estimates are subject to large numerical errors.

To compute accurate estimates, we consider selecting a subset $S \subseteq [P]$ of candidate explanatory variables. Let us denote by $\boldsymbol{R}_S$ the correlation sub-matrix of a subset of variables, that is, $\boldsymbol{R}_S := (r_{pq};\ p, q \in S)$. To avoid multicollinearity, the condition number of the sub-matrix should not exceed a user-defined parameter κ (> 1). The subset selection problem determines a subset S of explanatory variables so that the residual sum of squares of a subset regression model is minimized:

$$\begin{aligned}
\underset{\boldsymbol{a},\, S}{\text{minimize}} \quad & \sum_{n\in[N]} \left(y_n - \sum_{p\in S} a_p x_{np} \right)^2 && \text{(3.4a)}\\
\text{subject to} \quad & \mathrm{cond}(\boldsymbol{R}_S) \leq \kappa, && \text{(3.4b)}\\
& S \subseteq [P]. && \text{(3.4c)}
\end{aligned}$$

The subset selection problem (3.4) can be converted into an MIO problem. Let $\boldsymbol{z} = (z_1, z_2, \ldots, z_P)^\top$ be a vector of 0-1 decision variables for selecting explanatory variables; that is, $z_p = 1$ if $p \in S$; otherwise, $z_p = 0$. The correlation sub-matrix is then written as $\boldsymbol{R}(\boldsymbol{z}) := (r_{pq};\ z_p = z_q = 1)$. Consequently, the subset selection problem is formulated as an MIO problem,

$$\begin{aligned}
\underset{\boldsymbol{a},\, \boldsymbol{z}}{\text{minimize}} \quad & (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a}) && \text{(3.5a)}\\
\text{subject to} \quad & z_p = 0 \ \Rightarrow \ a_p = 0 \quad (\forall p \in [P]), && \text{(3.5b)}\\
& \mathrm{cond}(\boldsymbol{R}(\boldsymbol{z})) \leq \kappa, && \text{(3.5c)}\\
& \boldsymbol{z} \in \{0,1\}^P. && \text{(3.5d)}
\end{aligned}$$

Here, if $z_p = 0$, then the pth explanatory variable is deleted from the regression model because its coefficient is set to zero by the logical implications (3.5b). It is known that these logical implications can be represented by using a big-M method or a special ordered set type 1 (SOS1) constraint [14, 15].

3.3 Mixed-integer semidefinite optimization Approach

This section presents an MISDO approach to best subset selection for eliminating multicollinearity.

3.3.1 Formulation

A convex quadratic objective function (3.5a) is expressed as a linear objective function with a PSD constraint [155, 161]. We begin by computing a decomposition of the form:

$$\boldsymbol{X}^\top \boldsymbol{X} = N\boldsymbol{R} = \boldsymbol{V}\boldsymbol{V}^\top,$$

where the square matrix $\boldsymbol{V} \in \mathbb{R}^{P\times P}$ can be created, e.g., by the Cholesky/eigenvalue decomposition. Introducing a scalar decision variable f to be minimized, we rewrite the associated constraint as a PSD constraint as follows:

$$(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a}) \leq f \iff \begin{pmatrix} \boldsymbol{I}_P & \boldsymbol{V}^\top \boldsymbol{a} \\ \boldsymbol{a}^\top \boldsymbol{V} & 2\boldsymbol{y}^\top \boldsymbol{X}\boldsymbol{a} - \boldsymbol{y}^\top \boldsymbol{y} + f \end{pmatrix} \succeq \boldsymbol{O},$$

where $\boldsymbol{I}$ is the identity matrix of size P, $\boldsymbol{O}$ is a zero matrix of appropriate size, and $\boldsymbol{A} \succeq \boldsymbol{B}$ means that $\boldsymbol{A} - \boldsymbol{B}$ is a PSD matrix.

We denote by $\mathrm{Diag}(\boldsymbol{x})$ the diagonal matrix whose diagonal entries are components of vector $\boldsymbol{x}$. We also denote by $\boldsymbol{A} \circ \boldsymbol{B}$ the Hadamard product of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. The next theorem shows that the condition number constraint (3.5c) is expressed as PSD constraints based on the following matrices:

$$\mathrm{Diag}(\boldsymbol{1} - \boldsymbol{z}) = \begin{pmatrix} 1 - z_1 & 0 & \cdots & 0 \\ 0 & 1 - z_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 - z_P \end{pmatrix},$$

$$\boldsymbol{R} \circ \boldsymbol{z}\boldsymbol{z}^\top = \begin{pmatrix} r_{11} z_1 z_1 & r_{12} z_1 z_2 & \cdots & r_{1P} z_1 z_P \\ r_{21} z_2 z_1 & r_{22} z_2 z_2 & \cdots & r_{2P} z_2 z_P \\ \vdots & \vdots & \ddots & \vdots \\ r_{P1} z_P z_1 & r_{p2} z_P z_2 & \cdots & r_{PP} z_P z_P \end{pmatrix}.$$

Theorem 3.1. *Suppose that $\boldsymbol{z} \in \{0,1\}^P$. Then, $\mathrm{cond}(\boldsymbol{R}(\boldsymbol{z})) \leq \kappa$ holds if and only if there exists $\lambda \in [1/\kappa, 1]$ such that*

$$\lambda \boldsymbol{I}_P - \mathrm{Diag}(\boldsymbol{1} - \boldsymbol{z}) \preceq \boldsymbol{R} \circ \boldsymbol{z}\boldsymbol{z}^\top \preceq \kappa \lambda \boldsymbol{I}_P. \tag{3.6}$$

Proof. Let Q be the number of nonzero elements of $\boldsymbol{z}$. Without loss of generality, we assume that

$$\begin{cases} z_p = 1 & \text{for } p \in [Q], \\ z_p = 0 & \text{for } p \in [P] \setminus [Q] \end{cases} \tag{3.7}$$

Since $\boldsymbol{R}(\boldsymbol{z})$ is a PSD matrix whose diagonal entries are all one, $0 \leq \lambda_{\min}(\boldsymbol{R}(\boldsymbol{z})) \leq 1 \leq \lambda_{\max}(\boldsymbol{R}(\boldsymbol{z}))$ holds. It then follows from the definition (3.3) that $\mathrm{cond}(\boldsymbol{R}(\boldsymbol{z})) \leq \kappa$ is equivalent to

$$\lambda_{\max}(\boldsymbol{R}(\boldsymbol{z})) \leq \kappa \lambda_{\min}(\boldsymbol{R}(\boldsymbol{z})).$$

This also implies that $1/\kappa \leq \lambda_{\max}(\boldsymbol{R}(\boldsymbol{z}))/\kappa \leq \lambda_{\min}(\boldsymbol{R}(\boldsymbol{z}))$. Therefore, it is necessary to consider only $\lambda_{\min}(\boldsymbol{R}(\boldsymbol{z})) \in [1/\kappa, 1]$.

Using a PSD constraint for minimizing the maximal eigenvalue [155, 161], the condition number constraint can be converted as follows:

$$\begin{aligned} & \lambda_{\max}(\boldsymbol{R}(\boldsymbol{z})) \leq \kappa \lambda_{\min}(\boldsymbol{R}(\boldsymbol{z})) \\ \iff & \exists \lambda \in [1/\kappa, 1],\ \lambda \leq \lambda_{\min}(\boldsymbol{R}(\boldsymbol{z})) \text{ and } \lambda_{\max}(\boldsymbol{R}(\boldsymbol{z})) \leq \kappa \lambda \\ \iff & \exists \lambda \in [1/\kappa, 1],\ \lambda \boldsymbol{I}_Q \preceq \boldsymbol{R}(\boldsymbol{z}) \preceq \kappa \lambda \boldsymbol{I}_Q \\ \iff & \exists \lambda \in [1/\kappa, 1],\ \begin{pmatrix} \lambda \boldsymbol{I}_Q & \boldsymbol{O} \\ \boldsymbol{O} & (\lambda - 1)\boldsymbol{I}_{P-Q} \end{pmatrix} \preceq \begin{pmatrix} \boldsymbol{R}(\boldsymbol{z}) & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} \end{pmatrix} \preceq \kappa \lambda \begin{pmatrix} \boldsymbol{I}_Q & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I}_{P-Q} \end{pmatrix} \\ \iff & \exists \lambda \in [1/\kappa, 1],\ \lambda \boldsymbol{I}_P - \mathrm{Diag}(\boldsymbol{1} - \boldsymbol{z}) \preceq \boldsymbol{R} \circ \boldsymbol{z}\boldsymbol{z}^\top \preceq \kappa \lambda \boldsymbol{I}_P. \end{aligned}$$

□

To linearize the bilinear term $\boldsymbol{z}\boldsymbol{z}^\top$ in constraint (3.6), we introduce a symmetric matrix of decision variables:

$$\boldsymbol{W} = \begin{pmatrix} w_{11} & w_{21} & \cdots & w_{P1} \\ w_{21} & w_{22} & \cdots & w_{P2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{P1} & w_{P2} & \cdots & w_{PP} \end{pmatrix}.$$

It is known that when $\boldsymbol{z} \in \{0,1\}^P$, $w_{pq} = z_p z_q$ can be rewritten by means of its convex and concave envelopes as follows [5, 116]:

$$w_{pq} \geq 0,\ w_{pq} \geq z_p + z_q - 1,\ w_{pq} \leq z_p,\ w_{pq} \leq z_q.$$

Consequently, the subset selection problem is cast into an MISDO problem,

$$\begin{aligned}
\underset{\boldsymbol{a}, f, \lambda, \boldsymbol{W}, \boldsymbol{z}}{\text{minimize}} \quad & f && \text{(3.8a)} \\
\text{subject to} \quad & \begin{pmatrix} \boldsymbol{I}_P & \boldsymbol{V}^\top \boldsymbol{a} \\ \boldsymbol{a}^\top \boldsymbol{V} & 2\boldsymbol{y}^\top \boldsymbol{X}\boldsymbol{a} - \boldsymbol{y}^\top \boldsymbol{y} + f \end{pmatrix} \succeq \boldsymbol{O}, && \text{(3.8b)} \\
& z_p = 0 \ \Rightarrow\ a_p = 0 \quad (\forall p \in [p]), && \text{(3.8c)} \\
& \lambda \boldsymbol{I}_P - \mathrm{Diag}(\boldsymbol{1} - \boldsymbol{z}) \preceq \boldsymbol{R} \circ \boldsymbol{W} \preceq \kappa\lambda \boldsymbol{I}_P, && \text{(3.8d)} \\
& w_{pp} = z_p \quad (\forall p \in [P]), && \text{(3.8e)} \\
& w_{pq} \geq 0,\ w_{pq} \geq z_p + z_q - 1,\ w_{pq} \leq z_p,\ w_{pq} \leq z_q \\
& \qquad\qquad (\forall (p,q) \in [P] \times [P]), && \text{(3.8f)} \\
& 1/\kappa \leq \lambda \leq 1, \quad \boldsymbol{z} \in \{0,1\}^P. && \text{(3.8g)}
\end{aligned}$$

3.3.2 Normal-equation-based constraints

MISDO problems can be handled by a branch-and-bound procedure, but it involves solving a large number of relaxed semidefinite optimization problems. To improve computational efficiency, we consider including the normal equations (3.2) as the constraints of the MISDO problem. More precisely, when the pth explanatory variable is selected, the pth normal equation is placed as follows:

$$\begin{aligned}
& \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{a} + \boldsymbol{s} = \boldsymbol{X}^\top \boldsymbol{y}, && \text{(3.9)} \\
& z_p = 1 \ \Rightarrow\ s_p = 0 \quad (\forall p \in [P]), && \text{(3.10)}
\end{aligned}$$

where $\boldsymbol{s} = (s_1, s_2, \ldots, s_P)^\top$ is a vector of auxiliary decision variables.

The next theorem shows that constraints (3.9) and (3.10) are necessary optimality conditions for problem (3.5).

Theorem 3.2. *Let $(\boldsymbol{a}^\star, \boldsymbol{z}^\star)$ be an optimal solution to problem (3.5). There exists $\boldsymbol{s}^\star = (s_1^\star, s_2^\star, \ldots, s_P^\star)^\top$ such that*

$$\begin{aligned}
& \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{a}^\star + \boldsymbol{s}^\star = \boldsymbol{X}^\top \boldsymbol{y}, \\
& z_p^\star = 1 \ \Rightarrow\ s_p^\star = 0 \quad (\forall p \in [P]).
\end{aligned}$$

Proof. Without loss of generality, we can assume that

$$\boldsymbol{z}^\star = \begin{pmatrix} \boldsymbol{1} \\ \boldsymbol{0} \end{pmatrix},\ \boldsymbol{1} \in \mathbb{R}^Q,\ \boldsymbol{0} \in \mathbb{R}^{P-Q}.$$

According to $\boldsymbol{z}^\star$, we partition $\boldsymbol{X}$ and $\boldsymbol{a}^\star$ as follows:

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{X}_1 & \boldsymbol{X}_2 \end{pmatrix}, \ \boldsymbol{X}_1 \in \mathbb{R}^{N \times Q}, \ \boldsymbol{X}_2 \in \mathbb{R}^{N \times (P-Q)},$$
$$\boldsymbol{a}^\star = \begin{pmatrix} \boldsymbol{a}_1^\star \\ \boldsymbol{a}_2^\star \end{pmatrix}, \ \boldsymbol{a}_1^\star \in \mathbb{R}^Q, \ \boldsymbol{a}_2^\star \in \mathbb{R}^{P-Q}.$$

Because $(\boldsymbol{a}^\star, \boldsymbol{z}^\star)$ is an optimal solution to problem (3.5), we have $\boldsymbol{a}_2^\star = \boldsymbol{0}$. Moreover, $\boldsymbol{a}_1^\star$ minimizes RSS (3.1); that is, the following holds for $\boldsymbol{a}_1^\star$ and the associated normal equation:

$$\boldsymbol{X}_1^\top \boldsymbol{X}_1 \boldsymbol{a}_1^\star = \boldsymbol{X}_1^\top \boldsymbol{y}.$$

Now we define $\boldsymbol{s}^\star$ as follows:

$$\boldsymbol{s}^\star = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{X}_2^\top \boldsymbol{y} - \boldsymbol{X}_2^\top \boldsymbol{X}_1 \boldsymbol{a}_1^\star \end{pmatrix}.$$

It then follows that

$$\boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{a}^\star + \boldsymbol{s}^\star = \begin{pmatrix} \boldsymbol{X}_1^\top \boldsymbol{X}_1 \boldsymbol{a}_1^\star \\ \boldsymbol{X}_2^\top \boldsymbol{X}_1 \boldsymbol{a}_1^\star \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{X}_2^\top \boldsymbol{y} - \boldsymbol{X}_2^\top \boldsymbol{X}_1 \boldsymbol{a}_1^\star \end{pmatrix} = \begin{pmatrix} \boldsymbol{X}_1^\top \boldsymbol{y} \\ \boldsymbol{X}_2^\top \boldsymbol{y} \end{pmatrix} = \boldsymbol{X}^\top \boldsymbol{y},$$
$$z_p^\star = 1 \ \Rightarrow \ s_p^\star = 0 \quad (\forall p \in [P]).$$

□

3.4 Cutting-plane algorithm

This section presents a cutting-plane algorithm for solving Problem (3.5). The motivation behind this approach is that, as we see later in numerical experiments, the current general-purpose MISDO solver could solve only small-sized instances of Problem (3.8) and suffer from numerical instability. In contrast to the MISDO approach, where we can obtain an optimal solution by solving a single MIO problem, the cutting-plane algorithm has a theoretical drawback: it must solve an exponential number of MIQO problems, which are NP-hard, in a worst-case situation. However, it has a practical advantage in using sophisticated MIO solvers such as Gurobi and CPLEX to solve MIQO problems. In addition, we can improve its efficiency by generating strong cutting planes by exploiting the structure of the condition number constraint Eq. (3.5c).

The cutting-plane algorithm first removes the condition number constraint (3.5c) from the problem. Hence its feasible set is expressed as

$$\mathcal{F}_1 := \{(\boldsymbol{a}, \boldsymbol{z}) \in \mathbb{R}^P \times \{0,1\}^P \mid \text{Eqs. (3.5b) and (3.5d)}\}, \tag{3.11}$$

and the problem reduces to an MIQO problem, which can be handled by sophisticated MIO solvers such as Gurobi and CPLEX. The basic strategy of the algorithm involves repeatedly solving such relaxed MIQO problems and iteratively adding valid inequalities, instead of imposing the condition number constraint (3.5c), to the MIQO problems.

At the tth iteration ($t \geq 1$), our algorithm solves the follwing MIQO problem:

$$\underset{\boldsymbol{a}, \boldsymbol{z}}{\text{minimize}} \quad (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a}) \tag{3.12a}$$
$$\text{subject to} \quad (\boldsymbol{a}, \boldsymbol{z}) \in \mathcal{F}_t, \tag{3.12b}$$

where $\mathcal{F}_t$ is a relaxed feasible set at the tth iteration such that $\mathcal{F}_t \subseteq \mathcal{F}_1$, which will be updated according to Eq. (3.13) at the tth iteration. Let $(\boldsymbol{a}_t, \boldsymbol{z}_t)$ be an optimal solution to Problem (3.12). If $\mathrm{cond}(\boldsymbol{R}(\boldsymbol{z}_t)) \leq \kappa$, then $(\boldsymbol{a}_t, \boldsymbol{z}_t)$ is an optimal solution to Problem (3.5). In this case, we terminate the algorithm. Otherwise, we update the feasible region to cut off the solution $\boldsymbol{z}_t$ as follows:

$$\mathcal{F}_{t+1} \leftarrow \mathcal{F}_t \cap \{(\boldsymbol{a}, \boldsymbol{z}) \in \mathbb{R}^P \times \{0,1\}^P \mid \boldsymbol{z}_t^\top \boldsymbol{z} \leq \boldsymbol{z}_t^\top \boldsymbol{z}_t - 1\}. \tag{3.13}$$

After that, we set $t \leftarrow t+1$ and solve the MIQO problem (3.12) again. We repeat this process until we obtain a solution that satisfies the condition number constraint (3.5c).

We next show that the condition number is not increased by deleting explanatory variables.

Lemma 3.3. *Suppose that* $\boldsymbol{z}, \bar{\boldsymbol{z}} \in \{0,1\}^P$. *If* $\boldsymbol{z} \geq \bar{\boldsymbol{z}}$, *then* $\mathrm{cond}(\boldsymbol{R}(\boldsymbol{z})) \geq \mathrm{cond}(\boldsymbol{R}(\bar{\boldsymbol{z}}))$.

Proof. It follows from the Cauchy's interlace theorem (see, e.g., Theorem 4.3.17 [84]) that

$$0 \leq \lambda_{\min}(\boldsymbol{R}(\boldsymbol{z})) \leq \lambda_{\min}(\boldsymbol{R}(\bar{\boldsymbol{z}})) \leq \lambda_{\max}(\boldsymbol{R}(\bar{\boldsymbol{z}})) \leq \lambda_{\max}(\boldsymbol{R}(\boldsymbol{z})),$$

which completes the proof. □

The next theorem states that the feasible set of the original problem (3.5) is contained in $\mathcal{F}_t$ for all t.

Theorem 3.4. *Suppose that* $\bar{\boldsymbol{z}} \in \{0,1\}^P$ *and* $\mathrm{cond}(\boldsymbol{R}(\bar{\boldsymbol{z}})) > \kappa$. *If* $\boldsymbol{z} \in \{0,1\}^P$ *satisfies* $\mathrm{cond}(\boldsymbol{R}(\boldsymbol{z})) \leq \kappa$, *then it also satisfies* $\bar{\boldsymbol{z}}^\top \boldsymbol{z} \leq \bar{\boldsymbol{z}}^\top \bar{\boldsymbol{z}} - 1$.

Proof. We prove the proposition by contradiction:

$$\begin{aligned}
\bar{\boldsymbol{z}}^\top \boldsymbol{z} > \bar{\boldsymbol{z}}^\top \bar{\boldsymbol{z}} - 1 &\Rightarrow \bar{\boldsymbol{z}}^\top (\boldsymbol{z} - \bar{\boldsymbol{z}}) \geq 0 \quad \because \boldsymbol{z}, \bar{\boldsymbol{z}} \in \{0,1\}^P \\
&\Rightarrow \boldsymbol{z} \geq \bar{\boldsymbol{z}} \\
&\Rightarrow \mathrm{cond}(\boldsymbol{R}(\boldsymbol{z})) \geq \mathrm{cond}(\boldsymbol{R}(\bar{\boldsymbol{z}})) \quad \because \text{Lemma 3.3} \\
&\Rightarrow \mathrm{cond}(\boldsymbol{R}(\boldsymbol{z})) > \kappa.
\end{aligned}$$

□

This cutting-plane algorithm is developed on the basis of Bertsimas and King [30] and requires solving a large number of MIQO problems. To reduce the number of MIQO problems to be solved, we develop stronger valid inequalities for approximating the condition number constraint (3.5c). To this end, we employ a backward elimination method that searches a smaller subset of collinear variables. Specifically, it starts with an incumbent solution (e.g., $\boldsymbol{z}_t$) and deletes explanatory variables one by one on the basis of the RSS (3.1). Finally, we obtain $\bar{\boldsymbol{z}}$ $(\leq \boldsymbol{z}_t)$ such that $\mathrm{cond}(\boldsymbol{R}(\bar{\boldsymbol{z}})) > \kappa$. The feasible set is then updated:

$$\mathcal{F}_{t+1} \leftarrow \mathcal{F}_t \cap \{(\boldsymbol{a}, \boldsymbol{z}) \in \mathbb{R}^P \times \{0,1\}^P \mid \bar{\boldsymbol{z}}^\top \boldsymbol{z} \leq \bar{\boldsymbol{z}}^\top \bar{\boldsymbol{z}} - 1\}. \tag{3.14}$$

This cutting plane cuts off all $\boldsymbol{z} \in \{0,1\}^P$ satisfying $\boldsymbol{z} \geq \bar{\boldsymbol{z}}$; therefore, it is stronger than the previous one (3.13) because $\boldsymbol{z}_t \geq \bar{\boldsymbol{z}}$. Our cutting-plane algorithm is summarized by Algorithm 3.1

We next prove the finite convergence of the algorithm.

Algorithm 3.1 Cutting-plane algorithms for solving Problem (3.5)

Step 0 (Initialization) Set $t \leftarrow 1$. Let $\mathcal{F}_1$ be a feasible set (3.11).

Step 1 (Relaxed Problem) Solve Problem (3.12). Let $(\boldsymbol{a}, \boldsymbol{z}_t)$ be an optimal solution.

Step 2 (Muliticollinearity Detection) If $\mathrm{cond}(\boldsymbol{R}(\boldsymbol{z}_t)) \leq \kappa$, terminate the algorithm with the solution $\boldsymbol{z}_t$.

Step 3 (Backward Elimination) Find a solution $\bar{\boldsymbol{z}} \in \{0,1\}^P$ such that $\bar{\boldsymbol{z}} \leq \boldsymbol{z}_t$ and $\mathrm{cond}(\boldsymbol{R}(\bar{\boldsymbol{z}})) > \kappa$ by using a backward elimination method starting with $\boldsymbol{z}_t$.

Step 4 (Cut Generation) Add cut (3.14) to update the feasible set $\mathcal{F}_t$.

Step 5 Set $t \leftarrow t+1$ and return to Step 1.

Theorem 3.5. *Algorithm 3.1 provides an optimal solution to problem (3.5) in a finite number of iterations.*

Proof. Step 3 in Algorithm 3.1 removes $\boldsymbol{z}_t$ from $\mathcal{F}_t$ in each iteration; therefore, the algorithm terminates with a feasible solution (e.g., $\boldsymbol{z} = (1,0,0,\ldots,0)^\top$) after at most 2^P iterations.

Let $(\boldsymbol{a}^\star, \boldsymbol{z}^\star)$ be an optimal solution to problem (3.5). Theorem 3.4 guarantees that the feasible set of problem (3.5) is contained in $\mathcal{F}_t$ and hence that $(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a}_t)^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a}_t) \leq (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a}^\star)^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a}^\star)$ for all t. Therefore, the algorithm provides an optimal solution to problem (3.5). □

3.5 Numerical experiments

In this section, we assess the computational performance of our mixed-integer optimization approaches to best subset selection for eliminating multicollinearity.

We downloaded six datasets for regression analysis from the UCI Machine Learning Repository [109]. Tables 3.1 lists the instances used for computational experiments, where N and P are the number of samples and number of candidate explanatory variables, respectively. In the `SolarFlareC` instance, C-class flares production was employed as an explained variable. In the `ForestFires` instance, interaction terms were created from the variables of the x-axis and y-axis spatial coordinates. Each categorical variable was transformed into one or more dummy variables. All explanatory and explained variables were standardized to a mean of zero and standard deviation of one, as mentioned in Section 2.1. Samples containing missing values and redundant variables having the same value in all samples were eliminated.

3.5.1 Computational performance of MISDO approaches

We first evaluate the computational performance of the following MISDO approaches:

MISDO MISDO formulation (3.8)

MISDO$_{\mathbf{NE}}$ MISDO formulation (3.8) with the normal-equation-based constraints (3.9) and (3.10)

Table 3.1: List of instances

Abbreviation	N	P	Original dataset [109]
`Servo`	167	19	Servo
`AutoMPG`	392	25	Auto MPG
`SolarFlareC`	1066	26	Solar Flare (C-class flares production)
`BreastCancer`	194	32	Breast Cancer Wisconsin
`ForestFires`	517	63	Forest Fires
`Automobile`	159	65	Automobile

Table 3.2: Values of big-M for constraints (3.16)

Instance	N	P	Method	M ($\kappa = 100$)	M ($\kappa = 225$)
`Servo`	167	19	MISDO	2.25	2.25
			MISDO_{NE}	2.50	2.50
`AutoMPG`	392	25	MISDO	2.50	5.00
			MISDO_{NE}	1.05	1.50

Here the logical implications (3.10) was represented by means of the big-M method,

$$-M(1-z_p) \leq s_p \leq M(1-z_p) \quad (\forall p \in [P]), \tag{3.15}$$

where M was set to 1000 in all instances of our experiments. Similarly, the implications (3.8c) were represented with the big-M method,

$$-Mz_p \leq a_p \leq Mz_p \quad (\forall p \in [P]). \tag{3.16}$$

However, we had difficulty in finding a unified value of M for constraints (3.16) such that MISDO computation was not aborted due to numerical instability. Hence, we tuned the values of M through preliminary experiments as shown in Table 3.2.

These computations were performed on a Linux computer with an Intel Core2 Quad CPU (2.66 GHz) and 4 GB memory. MISDO problems were solved by using SCIP-SDP[1] 2.0.0 [61] combined with SCIP[2] 3.2.0 [1] and SDPA[3] 7.3.8 [168]. The computation for solving the MISDO problem was terminated if it did not finish by itself within 10000 s. In this case, the best feasible solution obtained within 10000 s was taken as the result. Chatterjee and Hadi [42] mention that when the value of the condition number exceeds 225, the deleterious effects of multicollinearity in the data become strong. Hence, the upper bound on the condition number was set as $\kappa = 100$ and 225.

Remark 3.1. We can use the cutting-plane and branch-and-cut algorithms in Chapter 2 for solving MISDO and MISDO_{NE}. However, the study in this chapter was conducted chronologically prior to that in Chapter 2, and thus, we did not used them in this experiment.

[1] http://www.opt.tu-darmstadt.de/scipsdp/
[2] http://scip.zib.de/
[3] http://sdpa.sourceforge.net/

Table 3.3: Results of solving MISDO problems ($\kappa = 100$)

Instance	N	P	Method	R^2	Cond	$\lvert S\rvert$	Time (s)
`Servo`	167	19	MISDO	**0.75877**	78.4	15	60.19
			MISDO_{NE}	**0.75877**	78.3	15	17.74
`AutoMPG`	392	25	MISDO	0.87429	76.9	19	>10000.00
			MISDO_{NE}	**0.87430**	92.8	21	5563.34

Table 3.4: Results of solving MISDO problems ($\kappa = 225$)

Instance	N	P	Method	R^2	Cond	$\lvert S\rvert$	Time (s)
`Servo`	167	19	MISDO	**0.75877**	97.1	15	4.99
			MISDO_{NE}	**0.75877**	104.1	15	5.49
`AutoMPG`	392	25	MISDO	0.87438	142.0	21	>10000.00
			MISDO_{NE}	**0.87438**	184.3	22	336.14

Tables 3.3 and 3.4 show the computational results of solving MISDO problems with $\kappa = 100$ and 225, respectively. The results for only the small-sized instances `Servo` and `AutoMPG` are shown in the tables because many of the large-scale MISDO problems could not be solved because of numerical instability[4] (i.e., violation of Slater's condition). The column labeled "R^2" shows the value of the coefficient of determination of a subset regression model built by each method, i.e.,

$$R^2 = 1 - \frac{\sum_{n\in[N]}(y_n - \sum_{p\in S} a_p x_{np})^2}{\sum_{n\in[N]}(y_n - \bar{y})^2},$$

where $\bar{y} := (\sum_{n\in[N]} y_n)/N$. Note here that the largest R^2 values for each instance are indicated in bold. The column labeled "Cond" shows the condition number of the correlation sub-matrix $\boldsymbol{R}_S$, and the column labeled "$|S|$" shows the number of selected explanatory variables. The column labeled "Time (s)" shows the computation time in seconds. Note that the computation of the cutting-plane algorithm was terminated if it did not finish by itself within 10000 s. In such cases, the best feasible solution obtained within 10000 s was used as the result, and "N/A" means that no feasible solution was found.

Tables 3.3 and 3.4 show that all the MISDO problems for `Servo` were solved within 61 s, and they were solved faster when $\kappa = 225$ than when $\kappa = 100$. Moreover, the normal-equation-based constraints worked effectively in speeding up the computations. For instance, in Table ,3.3 the computation time of the MISDO formulation was reduced from 60.19 s to 17.74 s by incorporating the normal-equation-based constraints into the problem.

In the case of `AutoMPG`, only the computations of MISDO_{NE} finished within 10000 s for both $\kappa = 100$ and 225. Furthermore, MISDO_{NE} attained the largest R^2 value for every instance in Tables 3.3 and 3.4. These results demonstrate the effectiveness of the normal-equation-based constraints in the MISDO formulation

[4]Using SCIP-SDP 2.1.0 instead of 2.0.0 and softening feasibility tolerances of SCIP and SCIP-SDP solvers (`feastol` and `sdpsolverfeastol`, respectively) from 10^{-6} (default) to 10^{-4} could resolve numerical issues, but it missed true optimal solutions to some of the instances. Hence, we used SCIP-SDP 2.0.0 with its default parameters.

for best subset selection to eliminate multicollinearity.

3.5.2 Computational performance of cutting-plane algorithms

Next, we compare the computational performance of the cutting-plane algorithms with that of conventional local search algorithms for subset selection. The algorithms used in the comparison are listed below:

FwS Forward selection method: Starts with $S = \emptyset$ and iteratively adds the variable p (i.e., $S \leftarrow S \cup \{p\}$) that leads to the largest decrease in RSS (3.1); this operation is repeated while $\mathrm{cond}(\boldsymbol{R}_S) \leq \kappa$ is satisfied.

BwE Backward elimination method: Starts with $S = [P]$ and iteratively eliminates the variable p (i.e., $S \leftarrow S \setminus \{p\}$) that leads to the smallest increase in RSS (3.1); this operation is repeated until $\mathrm{cond}(\boldsymbol{R}_S) \leq \kappa$ holds.

CPA Cutting-plane algorithm that omits Step 2 (Backward elimination) and appends cut (3.13).

CPA$_{\mathbf{BwE}}$ Cutting-plane algorithm that appends cut (3.14) strengthened by means of the backward elimination method.

These computations were performed on a Linux computer with an Intel Xeon W3520 CPU (2.66 GHz) and 12 GB memory. The algorithms FwS and BwE were implemented in MATLAB[5] R2013a ; the algorithms CPA and CPA$_{\mathrm{BwE}}$ were implemented in Python 2.7.3, with Gurobi Optimizer[6] 5.6.0 used to solve relaxed MIQO problems (3.12a) and (3.12b). Here the logical implications (3.5b) were incorporated in the form of SOS1 constraints, which imply that at most one element in the set can have a nonzero value. Specifically, the SOS1 constraint is imposed on $\{1 - z_p, a_p\}$ ($p \in [P]$) with the SOS type 1 function implemented in Gurobi Optimizer. Therefore, if $z_p = 0$, then $1 - z_p$ has a nonzero value and a_p must be zero from the SOS1 constraints. The upper bound on the condition number was set as $\kappa = 100$ or 225.

Tables 3.5 and 3.6 show the computational results obtained using the four algorithms with $\kappa = 100$ and 225, respectively. The column labeled "#Iter" shows the number of iterations in the cutting-plane algorithms. Note that the computation of the cutting-plane algorithm was terminated if it did not finish by itself within 10000 s. In such cases, the best feasible solution obtained within 10000 s was used as the result and "N/A" means that no feasible solution was found.

We can see from Tables 3.5 and 3.6 that the local search algorithms FwS and BwE finished their computations within 2 s for all the instances. Their obtained solutions were, however, frequently inferior to those of the cutting-plane algorithms, especially when $\kappa = 100$ (see Table 3.5); for the `BreastCancer` instance, CPA$_{\mathrm{BwE}}$ gave an R^2 value of 0.29040, whereas FwS and BwE gave R^2 values of 0.27580 and 0.26572, respectively.

We can also see that the computation finished much faster for CPA$_{\mathrm{BwE}}$ than for CPA. The main reason for this is that the number of iterations required for CPA$_{\mathrm{BwE}}$ was significantly reduced by the strengthened cut (3.14). Indeed, in the case of `SolarFlareC` in Table 3.5, CPA arrived at an optimal solution in 1246.97 s after 4209 iterations, whereas CPA$_{\mathrm{BwE}}$ terminated with an optimal solution in 84.59 s after 331 iterations.

[5] http://www.mathworks.com/products/matlab/
[6] http://www.gurobi.com

Table 3.5: Results of local search algorithms and cutting-plane algorithms ($\kappa = 100$)

Instance	N	P	Method	R^2	Cond	$\lvert S\rvert$	#Iter	Time (s)
`Servo`	167	19	FwS	0.75862	63.2	14	—	0.09
			BwE	**0.75877**	39.0	15	—	0.04
			CPA	**0.75877**	38.4	15	76	0.93
			CPA_{BwE}	**0.75877**	39.9	15	16	0.42
`AutoMPG`	392	25	FwS	0.87335	87.6	21	—	0.17
			BwE	0.87429	86.2	19	—	0.09
			CPA	**0.87430**	93.2	21	1284	325.20
			CPA_{BwE}	**0.87430**	92.8	21	84	14.11
`SolarFlareC`	1066	26	FwS	0.19713	4.3	19	—	0.23
			BwE	0.19705	2.0	15	—	0.87
			CPA	**0.19715**	18.7	19	4209	1246.97
			CPA_{BwE}	**0.19715**	34.3	19	331	84.59
`BreastCancer`	194	32	FwS	0.27580	91.8	15	—	0.17
			BwE	0.26572	61.8	9	—	0.27
			CPA	N/A			>4364	>10000.00
			CPA_{BwE}	**0.29040**	95.8	14	>1044	>10000.00
`ForestFires`	517	63	FwS	0.16399	99.9	52	—	1.68
			BwE	0.16481	53.1	50	—	1.32
			CPA	N/A			>5901	>10000.00
			CPA_{BwE}	**0.16537**	96.5	59	53	301.07
`Automobile`	159	65	FwS	0.96447	99.9	27	—	0.76
			BwE	0.96571	67.3	24	—	1.89
			CPA	N/A			>6960	>10000.00
			CPA_{BwE}	**0.96908**	72.9	25	>277	>10000.00

Note that when the computation is terminated due to the time limit of 10000 s, CPA does not provide a feasible solution because it is found only at the end of the algorithm. In contrast, CPA_{BwE} can find a feasible solution of good quality in the early stage of the algorithm by means of the backward elimination method. For this reason, CPA_{BwE} always provided the best solution among the four methods for all the instances in Tables 3.5 and 3.6.

Comparing the MISDO approaches and the cutting-plane algorithms, it needs to be said that the computational performance of the MISDO approaches was much lower than that of the cutting-plane algorithms. For instance in the case of `AutoMPG` with $\kappa = 225$, MISDO_{NE} took 336.14 s to solve the problem (Table 3.4), but CPA_{BwE} required only 1.76 s to solve the same problem (Table 3.6). These results imply that our MISDO approach was not effective for the subset selection problem at the current moment; however, since the computational performance of MISDO algorithms is being improved, our MISDO formulation will work better in the future.

Table 3.6: Results of local search algorithms and cutting-plane algorithms ($\kappa = 225$)

Instance	N	P	Method	R^2	Cond	$\lvert S\rvert$	#Iter	Time (s)
`Servo`	167	19	FwS	**0.75877**	146.3	15	—	0.11
			BwE	**0.75877**	39.0	15	—	0.04
			CPA	**0.75877**	102.5	15	56	0.62
			CPA_{BwE}	**0.75877**	39.9	15	15	0.40
`AutoMPG`	392	25	FwS	**0.87438**	185.3	22	—	0.18
			BwE	**0.87438**	181.1	22	—	0.05
			CPA	**0.87438**	157.1	22	60	2.04
			CPA_{BwE}	**0.87438**	173.2	22	18	1.76
`SolarFlareC`	1066	26	FwS	0.19713	4.3	19	—	0.22
			BwE	0.19705	2.0	15	—	0.48
			CPA	**0.19715**	18.7	19	4209	1244.26
			CPA_{BwE}	**0.19715**	169.4	19	750	189.48
`BreastCancer`	194	32	FwS	0.30010	217.9	19	—	0.21
			BwE	0.26572	61.8	9	—	0.27
			CPA	N/A			>4369	>10000.00
			CPA_{BwE}	**0.30513**	215.6	20	287	288.59
`ForestFires`	517	63	FwS	**0.16556**	214.2	60	—	1.74
			BwE	**0.16556**	212.6	60	—	0.42
			CPA	**0.16556**	209.5	60	59	7.60
			CPA_{BwE}	**0.16556**	209.0	60	12	27.37
`Automobile`	159	65	FwS	0.97124	224.4	39	—	1.14
			BwE	0.97153	183.5	29	—	1.85
			CPA	N/A			>6973	>10000.00
			CPA_{BwE}	**0.97391**	224.4	36	>960	>10000.00

3.6 Conclusion

This chapter addressed the problem of selecting the best subset of explanatory variables subject to an upper bound on the condition number for eliminating multicollinearity from linear regression models. The first contribution of this study is a novel computational framework for eliminating multicollinearity based on the MISDO formulation. This framework reformulates the subset selection problem as a single MISDO problem. The second contribution is the establishment of a high-performance cutting-plane algorithm. In this algorithm, we generate strong cutting-planes with backward elimination and reduce the number of mixed-integer quadratic optimization problems to be solved. While numerical experiments show that our MISDO formulation could only be applied to small-sized instances at present, this chapter provides a new statistical application of mixed-integer semidefinite optimization formulation. Moreover, we found that our cutting-plane algorithm frequently provided a better subset of variables than did the common local search algorithms. This finding demonstrates the effectiveness of the specialized cutting-plane algorithm in eliminating multicollinearity from a linear regression model.

Chapter 4

Cutting-plane Algorithm for Cardinality-constrained Distributionally Robust Portfolio Optimization

This chapter focuses on portfolio selection problems. In particular, we treat a distributionally robust portfolio optimization model with a cardinality constraint, which limits the number of invested assets. This problem is formulated as a MISDO problem, and thus, solving it exactly is computationally challenging when the number of investable assets is large. To overcome this issue, we propose a specialized cutting-plane algorithm to solve the cardinality-constrained distributionally robust optimization problem. We first reformulate the problem as a bilevel optimization problem and design a cutting-plane algorithm for solving the upper-level problem. Then, to generate cutting planes efficiently, we apply the technique of positive semidefinite matrix completion to the lower-level problem and reduce its problem size. The content of this chapter is included in Kobayashi *et al.* [96].

We give an introduction to this study and summarize our contribution in Section 4.1 In Section 4.2, we give a formulation of the cardinality-constrained distributionally robust portfolio optimization problem. In Section 4.3, we explain our cutting-plane algorithm for solving the problem. In Section 4.4, we describe the reduction of the lower-level SDO problem. We report computational results in Section 4.5 and conclude in Section 4.6.

4.1 Introduction

Portfolio optimization models, originating from the mean-variance portfolio selection pioneered by Markowitz [114], have been actively studied by academic researchers and institutional investors. In real-world portfolio optimization problems, we must depend on inaccurate estimates of asset returns, which can lead to lower investment performance. Accordingly, robust optimization [18, 20], which copes with uncertainty about input data, has played an important role in many practical situations [48, 53, 72]. This chapter focuses on a distributionally robust optimization model developed by Delage and Ye [47] for portfolio optimization.

In distributionally robust optimization, optimal solutions are evaluated under the worst-case expectation with respect to a set of probability distributions of uncertain parameters. This model was first introduced by Scarf [142] for an inventory problem. Distributionally robust optimization can be viewed as a framework unifying traditional stochastic programming based on sample average approximation [93, 144, 145] and conventional robust optimization based on uncertainty sets of possible realizations of random variables [18, 20]. Moreover, some theoretical

analyses support the effectiveness of distributionally robust optimization in data-driven decision-making under uncertainty [69, 126].

Moment-based uncertainty sets, which are sets of probability distributions of asset returns whose moments satisfy certain conditions, are commonly adopted in distributionally robust portfolio optimization [51, 66, 107, 131, 160]. El Ghaoui *et al.* [51] considered minimizing the worst-case value-at-risk over an uncertainty set of probability distributions such that the mean and covariance matrix are elementwise bounded. Tütüncü and Koeing [160] used a similar uncertainty set for solving robust mean-variance portfolio optimization problems. Goldfarb and Iyengar [66] studied a distributionally robust portfolio optimization problem in which asset returns are formed by a linear factor model. Popescu [131] assumed that the mean and covariance matrix are exactly known and employed an uncertainty set of probability distributions whose first and second moments are equal to them. Similar assumptions were also made in other distributionally robust optimization models [122, 175]. Goh and Sim [65] studied an uncertainty set of probability distributions whose mean belongs to a convex set, and covariance matrix is given. As an extension of the elementwise uncertainty sets [51, 160], Lotfi and Zenios [107] considered an ellipsoidal uncertainty set of the tuple of mean and covariance matrix when minimizing the worst-case value-at-risk and conditional value-at-risk.

Unlike these models that require prior knowledge of the region including the mean and/or covariance matrix, Delage and Ye [47] proposed a data-driven method for defining a moment-based uncertainty set and gave its probabilistic guarantee. They also formulated the worst-case loss minimization problem over this uncertainty set as an SDO problem, which can be solved in polynomial time by interior-point methods [123]. Bertsimas *et al.* [29] proposed a practical framework based on a bootstrap hypothesis test to determine the uncertainty set of probability distributions from historical data. See Rahimian and Mehrotra [133] for a comprehensive survey on distributionally robust optimization models.

This chapter focuses on solving distributionally robust portfolio optimization problems based on the moment-based uncertainty set [47] with a cardinality constraint to limit the number of invested assets. In real-world practice, when the number of invested assets is large, it is difficult for investors to keep track of each asset, and substantial transaction costs are required [113, 128]. Thus, there is a need to control the number of invested assets by introducing the cardinality constraint. However, solving such a cardinality-constrained distributionally robust model is computationally challenging. Due to the cardinality constraint and the moment-based uncertainty set, the proposed model is formulated as a MISDO problem.

Recently, Bertsimas *et al.* [26] proposed a general framework of cutting-plane algorithms to exactly solve mixed-integer convex optimization problems with logical constraints. They reformulated this problem as a bilevel optimization problem composed of lower- and upper-level problems. To solve the upper-level problem, they devised a cutting-plane algorithm, which iteratively approximates the objective function by generating cutting planes from the solution to the lower-level problem with the strong duality theory. Bertsimas and Cory-Wright [24] applied this algorithm to the cardinality-constrained mean-variance portfolio optimization. They demonstrated that their algorithm was much faster than state-of-the-art MIO methods when solving large-scale problem instances. The cutting-plane algorithm was also applied to sparse principal component analysis [25] and sparse inverse covariance estimation [33].

Motivated by these prior studies, we propose a specialized cutting-plane algo-

rithm to exactly solve the cardinality-constrained distributionally robust portfolio optimization problem. At each iteration of this cutting-plane algorithm, we must solve an SDO problem, which is the dual of the lower-level problem for generating cutting planes. However, the size of this SDO problem depends on the number of investable assets, so solving this problem at each iteration is computationally prohibitive when handling a large number of assets. To overcome this difficulty, we reduce the size of the lower-level SDO problem through the application of the matrix completion technique [60, 121]. Notably, the size of the reduced SDO problem depends not on the number of investable assets but on the cardinality parameter, which is usually set to a small number. To the best of our knowledge, we are the first to develop an effective algorithm for exactly solving cardinality-constrained distributionally robust portfolio optimization problems. Numerical experiments using real-world datasets [16, 41, 59, 90] demonstrate the effectiveness of our method in terms of both computational and out-of-sample investment performances.

4.2 Problem formulation

In this section, we formulate the cardinality-constrained distributionally robust portfolio optimization problem that we consider in this chapter.

4.2.1 Cardinality constraint

Let $\boldsymbol{x} := (x_1, x_2, \ldots, x_N)^\top$ be a portfolio, where x_n is the investment weight of the nth asset. Throughout this paper, we consider the set of feasible portfolios:

$$\mathcal{X} := \left\{ \boldsymbol{x} \in \mathbb{R}^N \;\middle|\; \sum_{n\in[N]} x_n = 1, \quad \boldsymbol{x} \geq 0 \right\}.$$

The nonnegativity constraint on $\boldsymbol{x}$ prohibits short selling.

Let $k \in [N]$ be a user-defined parameter for limiting the cardinality (i.e., the number of assets to be held). We then impose the following cardinality constraint [27, 34, 128] on portfolio $\boldsymbol{x}$:

$$\|\boldsymbol{x}\|_0 \leq k, \tag{4.1}$$

where $\|\cdot\|_0$ is the ℓ_0-norm (i.e., the number of nonzero entries). In practice, this constraint is required by investors to reduce their portfolio monitoring and transaction costs.

Let $\boldsymbol{z} := (z_1, z_2, \ldots, z_N)^\top$ be a vector composed of binary decision variables for selecting assets; that is, $z_n = 1$ if the nth asset is selected, and $z_n = 0$ otherwise. We also introduce the feasible set corresponding to the cardinality constraint (4.1):

$$\mathcal{Z}_N^k := \left\{ \boldsymbol{z} \in \{0,1\}^N \;\middle|\; \sum_{n\in[N]} z_n = k \right\}.$$

Then, the cardinality constraint (4.1) is represented by the following logical implication:

$$\begin{cases} z_n = 0 \;\Rightarrow\; x_n = 0 \quad (\forall n \in [N]), & (4.2) \\ \boldsymbol{z} \in \mathcal{Z}_N^k. & (4.3) \end{cases}$$

4.2.2 Moment-based uncertainty set

We consider a measurable space $(\Omega, \mathcal{F})$. Let $\tilde{\boldsymbol{\xi}} : \Omega \to \mathbb{R}^N$ be an $\mathcal{F}$-measurable function (N-dimensional random vector) representing the rate of random return of each asset. Suppose that $\boldsymbol{\xi}_m \in \mathbb{R}^N$ is an observed historical return for each period $m \in [M]$. Then, the sample mean vector and the sample covariance matrix are calculated as

$$\hat{\boldsymbol{\mu}} := \frac{1}{M} \sum_{m \in [M]} \boldsymbol{\xi}_m, \qquad \hat{\boldsymbol{\Sigma}} := \frac{1}{M} \sum_{m \in [M]} (\boldsymbol{\xi}_m - \hat{\boldsymbol{\mu}})(\boldsymbol{\xi}_m - \hat{\boldsymbol{\mu}})^\top. \tag{4.4}$$

Let $\mathcal{M}$ be the set of all probability measures in the measurable space $(\Omega, \mathcal{F})$, and $\mathbb{E}_F[\,\cdot\,]$ be the expectation under the probability measure $F \in \mathcal{M}$. Delage and Ye [47] considered the moment-based uncertainty set of probability distributions of $\tilde{\boldsymbol{\xi}}$ with the assumption $\hat{\boldsymbol{\Sigma}} \succ \boldsymbol{O}$. When the support of distributions is $\mathbb{R}^N$, this uncertainty set is given by

$$\mathcal{D}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}, \kappa_1, \kappa_2) := \left\{ F \in \mathcal{M} \;\middle|\; \begin{array}{l} \left(\mathbb{E}_F[\tilde{\boldsymbol{\xi}}] - \hat{\boldsymbol{\mu}}\right)^\top \hat{\boldsymbol{\Sigma}}^{-1} \left(\mathbb{E}_F[\tilde{\boldsymbol{\xi}}] - \hat{\boldsymbol{\mu}}\right) \le \kappa_1 \\ \mathbb{E}_F[(\tilde{\boldsymbol{\xi}} - \hat{\boldsymbol{\mu}})(\tilde{\boldsymbol{\xi}} - \hat{\boldsymbol{\mu}})^\top] \preceq \kappa_2 \hat{\boldsymbol{\Sigma}} \end{array} \right\}, \tag{4.5}$$

where $\kappa_1 \ge 0$ and $\kappa_2 \ge 1$ are user-defined uncertainty parameters about $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$, respectively. The first condition in Eq. (4.5) ensures that the expectation of $\tilde{\boldsymbol{\xi}}$ lies in an ellipsoid of size κ_1 centered at $\hat{\boldsymbol{\mu}}$. The second condition in Eq. (4.5) implies that the second central-moment matrix of $\tilde{\boldsymbol{\xi}}$ is bounded above by $\kappa_2 \hat{\boldsymbol{\Sigma}}$ in the sense of the matrix inequality.

Throughout this chapter, we make the following mild assumption about the moment-based uncertainty set (4.5).

Assumption 4.1. $\kappa_1 > 0$ and $\hat{\boldsymbol{\Sigma}} \succ \boldsymbol{O}$.

4.2.3 Piecewise-linear utility and loss functions

Let $\boldsymbol{\xi} \in \mathbb{R}^N$ be a realization of the random return $\tilde{\boldsymbol{\xi}}$. Following Delage and Ye [47], we employ the following utility function, which is piecewise-linear and concave in the portfolio net return $\boldsymbol{\xi}^\top \boldsymbol{x}$ as

$$u(\boldsymbol{x}, \boldsymbol{\xi}) := \min_{\ell \in [L]} \left\{ a^{(\ell)} \boldsymbol{\xi}^\top \boldsymbol{x} + b^{(\ell)} \right\}, \tag{4.6}$$

where $a^{(\ell)} \in \mathbb{R}$ and $b^{(\ell)} \in \mathbb{R}$ are respectively the slope and intercept of the ℓth linear function for $\ell \in [L]$. The loss function is then defined as the negative of the utility function (4.6):

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\xi}) := -\min_{\ell \in [L]} \left\{ a^{(\ell)} \boldsymbol{\xi}^\top \boldsymbol{x} + b^{(\ell)} \right\} = \max_{\ell \in [L]} \left\{ -a^{(\ell)} \boldsymbol{\xi}^\top \boldsymbol{x} - b^{(\ell)} \right\}. \tag{4.7}$$

4.2.4 Portfolio optimization model

Our objective is to minimize the following worst-case expected loss with respect to an underlying probability distribution $F \in \mathcal{D}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}, \kappa_1, \kappa_2)$:

$$\max_{F \in \mathcal{D}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}, \kappa_1, \kappa_2)} \left\{ \mathbb{E}_F \left[\mathcal{L}(\boldsymbol{x}, \tilde{\boldsymbol{\xi}}) \right] \right\}. \tag{4.8}$$

The ℓ_2-regularization term is also incorporated into the objective from a perspective of robust optimization [24, 48, 70, 71].

We are now in a position to formulate our cardinality-constrained distributionally robust portfolio optimization model as follows:

$$\begin{aligned}
\underset{\boldsymbol{x},\boldsymbol{z}}{\text{minimize}} \quad & \frac{1}{2\gamma}\boldsymbol{x}^\top\boldsymbol{x} + \max_{F\in\mathcal{D}(\hat{\boldsymbol{\mu}},\hat{\boldsymbol{\Sigma}},\kappa_1,\kappa_2)} \left\{\mathbb{E}_F\left[\mathcal{L}(\boldsymbol{x},\tilde{\boldsymbol{\xi}})\right]\right\} && \text{(4.9a)}\\
\text{subject to} \quad & z_n = 0 \;\Rightarrow\; x_n = 0 \quad (\forall n\in[N]), && \text{(4.9b)}\\
& \boldsymbol{x}\in\mathcal{X}, \quad \boldsymbol{z}\in\mathcal{Z}_N^k, && \text{(4.9c)}
\end{aligned}$$

where $\gamma > 0$ is a user-defined regularization parameter. By deriving the dual of the inner maximization problem (4.8) as in Delage and Ye [47], Problem (4.9) can equivalently be reformulated as the following MISDO problem:

$$\begin{aligned}
\underset{\boldsymbol{x},\boldsymbol{P},\boldsymbol{Q},\boldsymbol{p},\boldsymbol{q},r,s,\boldsymbol{z}}{\text{minimize}} \quad & \frac{1}{2\gamma}\boldsymbol{x}^\top\boldsymbol{x} + (\kappa_2\hat{\boldsymbol{\Sigma}} - \hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^\top)\bullet\boldsymbol{Q} + r + \hat{\boldsymbol{\Sigma}}\bullet\boldsymbol{P} - 2\hat{\boldsymbol{\mu}}^\top\boldsymbol{p} + \kappa_1 s && \text{(4.10a)}\\
\text{subject to} \quad & \boldsymbol{p} = -\boldsymbol{q}/2 - \boldsymbol{Q}\hat{\boldsymbol{\mu}}, && \text{(4.10b)}\\
& \begin{pmatrix} \boldsymbol{Q} & \boldsymbol{q}/2 + a^{(\ell)}\boldsymbol{x}/2 \\ (\boldsymbol{q}/2 + a^{(\ell)}\boldsymbol{x}/2)^\top & r + b^{(\ell)} \end{pmatrix} \succeq \boldsymbol{O} \quad (\forall \ell\in[L]), && \text{(4.10c)}\\
& \begin{pmatrix} \boldsymbol{P} & \boldsymbol{p} \\ \boldsymbol{p}^\top & s \end{pmatrix} \succeq \boldsymbol{O}, && \text{(4.10d)}\\
& z_n = 0 \;\Rightarrow\; x_n = 0 \quad (\forall n\in[N]), && \text{(4.10e)}\\
& \boldsymbol{x}\in\mathcal{X}, \quad \boldsymbol{z}\in\mathcal{Z}_N^k, && \text{(4.10f)}
\end{aligned}$$

where $\boldsymbol{P},\boldsymbol{Q}\in\mathcal{S}^N$, $\boldsymbol{p},\boldsymbol{q}\in\mathbb{R}^N$, and $r,s\in\mathbb{R}$ are dual decision variables of Problem (4.8).

4.3 Cutting-plane algorithm

In this section, we give our cutting-plane algorithm for solving the cardinality-constrained distributionally robust portfolio optimization problem (4.10).

4.3.1 Bilevel optimization reformulation

We extend the method of bilevel optimization reformulation [24] to the cardinality-constrained distributionally robust portfolio optimization problem (4.10). Let us denote by $\boldsymbol{Z} := \mathrm{Diag}(\boldsymbol{z})$ a diagonal matrix whose diagonal entries are given by $\boldsymbol{z}$. The logical implication (4.10e) are then incorporated by replacing $\boldsymbol{x}$ with $\boldsymbol{Z}\boldsymbol{x}$ in Problem (4.10). Now, we can reformulate Problem (4.10) as a bilevel optimization problem. Specifically, the *upper-level problem* is written as the following integer optimization problem:

$$\underset{\boldsymbol{z}}{\text{minimize}} \quad f(\boldsymbol{z}) \quad \text{subject to} \quad \boldsymbol{z}\in\mathcal{Z}_N^k, \qquad \text{(4.11)}$$

and the *lower-level problem* for calculating the objective function is expressed as the following SDO problem:

$$f(\boldsymbol{z}) = \underset{\boldsymbol{x},\boldsymbol{P},\boldsymbol{Q},\boldsymbol{p},\boldsymbol{q},r,s}{\text{minimize}} \quad \frac{1}{2\gamma}\boldsymbol{x}^\top\boldsymbol{x} + (\kappa_2\hat{\boldsymbol{\Sigma}} - \hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^\top) \bullet \boldsymbol{Q} + r + \hat{\boldsymbol{\Sigma}} \bullet \boldsymbol{P} - 2\hat{\boldsymbol{\mu}}^\top\boldsymbol{p} + \kappa_1 s \tag{4.12a}$$

$$\text{subject to} \quad \boldsymbol{p} = -\boldsymbol{q}/2 - \boldsymbol{Q}\hat{\boldsymbol{\mu}}, \tag{4.12b}$$

$$\begin{pmatrix} \boldsymbol{Q} & \boldsymbol{q}/2 + a^{(\ell)}\boldsymbol{Z}\boldsymbol{x}/2 \\ (\boldsymbol{q}/2 + a^{(\ell)}\boldsymbol{Z}\boldsymbol{x}/2)^\top & r + b^{(\ell)} \end{pmatrix} \succeq \boldsymbol{O} \quad (\forall \ell \in [L]), \tag{4.12c}$$

$$\begin{pmatrix} \boldsymbol{P} & \boldsymbol{p} \\ \boldsymbol{p}^\top & s \end{pmatrix} \succeq \boldsymbol{O}, \tag{4.12d}$$

$$\boldsymbol{Z}\boldsymbol{x} \in \mathcal{X}. \tag{4.12e}$$

Note that $(\boldsymbol{Z}\boldsymbol{x})^\top\boldsymbol{Z}\boldsymbol{x}$ has been replaced by $\boldsymbol{x}^\top\boldsymbol{x}$ in Eq. (4.12a) because $\boldsymbol{x} = \boldsymbol{Z}\boldsymbol{x}$ holds for $\boldsymbol{x}$ that minimizes Eq. (4.12a) as in Bertsimas and Cory-Wright [24].

We then derive cutting planes that underestimate $f(\boldsymbol{z})$ in the upper-level problem (4.11) by exploiting the dual formulation of the lower-level problem (4.12). The following theorem yields the dual formulation of Problem (4.12), where $\boldsymbol{\omega} \circ \boldsymbol{\omega}$ denotes the Hadamard product of the vector $\boldsymbol{\omega}$.

Theorem 4.2. *For all $\boldsymbol{z} \in \mathcal{Z}_N^k$, the strong duality holds for Problem (4.12), and the dual formulation of Problem (4.12) is represented as follows:*

$$f(\boldsymbol{z}) = \underset{\boldsymbol{\omega},\boldsymbol{B},\boldsymbol{\beta},\boldsymbol{\eta},\boldsymbol{\lambda},\pi}{\text{maximize}} \quad -\frac{\gamma}{2}\boldsymbol{z}^\top(\boldsymbol{\omega} \circ \boldsymbol{\omega}) - \sum_{\ell\in[L]} \eta^{(\ell)} b^{(\ell)} + \pi \tag{4.13a}$$

$$\text{subject to} \quad \boldsymbol{\omega} \geq \sum_{\ell\in[L]} a^{(\ell)}\boldsymbol{\beta}^{(\ell)} + \pi\mathbf{1}, \tag{4.13b}$$

$$\sum_{\ell\in[L]} \boldsymbol{B}^{(\ell)} = \kappa_2\hat{\boldsymbol{\Sigma}} - \hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^\top + \hat{\boldsymbol{\mu}}\left(\sum_{\ell\in[L]} \boldsymbol{\beta}^{(\ell)}\right)^\top + \left(\sum_{\ell\in[L]} \boldsymbol{\beta}^{(\ell)}\right)\hat{\boldsymbol{\mu}}^\top, \tag{4.13c}$$

$$\sum_{\ell\in[L]} \boldsymbol{\beta}^{(\ell)} = \boldsymbol{\lambda} + \hat{\boldsymbol{\mu}}, \tag{4.13d}$$

$$1 - \sum_{\ell\in[L]} \eta^{(\ell)} = 0, \tag{4.13e}$$

$$\begin{pmatrix} \boldsymbol{B}^{(\ell)} & \boldsymbol{\beta}^{(\ell)} \\ (\boldsymbol{\beta}^{(\ell)})^\top & \eta^{(\ell)} \end{pmatrix} \succeq \boldsymbol{O} \quad (\forall \ell \in [L]), \tag{4.13f}$$

$$\begin{pmatrix} \hat{\boldsymbol{\Sigma}} & \boldsymbol{\lambda} \\ \boldsymbol{\lambda}^\top & \kappa_1 \end{pmatrix} \succeq \boldsymbol{O}, \tag{4.13g}$$

where $\boldsymbol{\omega} \in \mathbb{R}^N$, $\boldsymbol{B} := (\boldsymbol{B}^{(\ell)}) \in \mathbb{R}^{N\times N\times L}$, $\boldsymbol{\beta} := (\boldsymbol{\beta}^{(\ell)}) \in \mathbb{R}^{N\times L}$, $\boldsymbol{\eta} := (\eta^{(\ell)}) \in \mathbb{R}^L$, $\boldsymbol{\lambda} \in \mathbb{R}^N$, and $\pi \in \mathbb{R}$ are dual decision variables.

Proof. See Appendix A.1. □

According to Theorem 4.2, we can extend the definition of $f(\boldsymbol{z})$ to the optimal objective value of Problem (4.13) for real-valued $\boldsymbol{z} \in [0,1]^N$. Then, we obtain two key properties of $f(\boldsymbol{z})$ as in Bertsimas and Cory-Wright [24].

Lemma 4.3 (Convexity). *The function $f(\boldsymbol{z})$ is convex in $\boldsymbol{z} \in [0,1]^N$.*

Proof. Since $f(\boldsymbol{z})$ is a pointwise maximum of linear functions in $\boldsymbol{z} \in [0,1]^N$, its epigraph, which is the intersection of epigraphs of the linear functions, is convex. □

Lemma 4.4 (Subgradient). *Let $\boldsymbol{\omega}^\star(\boldsymbol{z})$ be an optimal solution of $\boldsymbol{\omega}$ to Problem (4.13) with $\boldsymbol{z} \in \{0,1\}^N$. Then, a subgradient of the function $f(\boldsymbol{z})$ is given by*

$$\boldsymbol{g}(\boldsymbol{z}) := -\frac{\gamma}{2}\boldsymbol{\omega}^\star(\boldsymbol{z}) \circ \boldsymbol{\omega}^\star(\boldsymbol{z}) \in \partial f(\boldsymbol{z}). \tag{4.14}$$

Proof. See, for example, Proposition 8.1.1 in Bertsekas *et al.* [23]. □

Lemmas 4.3 and 4.4 verify that for each $\hat{\boldsymbol{z}} \in \mathcal{Z}_N^k$, Problem (4.13) with $\boldsymbol{z} = \hat{\boldsymbol{z}}$ yields a linear underestimator of $f(\boldsymbol{z})$ for $\boldsymbol{z} \in [0,1]^N$ as follows:

$$f(\boldsymbol{z}) \geq f(\hat{\boldsymbol{z}}) + \boldsymbol{g}(\hat{\boldsymbol{z}})^\top(\boldsymbol{z} - \hat{\boldsymbol{z}}). \tag{4.15}$$

4.3.2 Algorithm description

We revise the cutting-plane algorithm [24] with the aim of solving the upper-level problem (4.10). Let θ_{LB} be a lower bound of the optimal objective value of Problem (4.11); this bound can easily be calculated by solving a continuous relaxation version of Problem (4.11). Our cutting-plane algorithm starts with the following initial feasible region:

$$\mathcal{F}_1 := \{(\boldsymbol{z}, \theta) \in \mathcal{Z}_N^k \times \mathbb{R} \mid \theta \geq \theta_{\mathrm{LB}}\}, \tag{4.16}$$

where θ is an auxiliary decision variable that corresponds to a lower estimate of $f(\boldsymbol{z})$.

At the tth iteration ($t \geq 1$), our algorithm solves the surrogate upper-level problem:

$$\underset{\boldsymbol{z},\theta}{\text{minimize}} \;\; \theta \quad \text{subject to} \;\; (\boldsymbol{z}, \theta) \in \mathcal{F}_t, \tag{4.17}$$

where $\mathcal{F}_t$ is a feasible region at the tth iteration such that $\mathcal{F}_{t+1} \subseteq \mathcal{F}_t$. Eq. (4.16) ensures that the objective value of Problem (4.17) is bounded below, and thus, there exists an optimal solution $(\boldsymbol{z}_t, \theta_t)$ to Problem (4.17).

We next solve the dual lower-level problem (4.13) with $\boldsymbol{z} = \boldsymbol{z}_t$. We thus obtain the function value $f(\boldsymbol{z}_t)$ and its subgradient $\boldsymbol{g}(\boldsymbol{z}_t)$ through Eq. (4.14). If $f(\boldsymbol{z}_t) - \theta_t \leq \varepsilon$ with sufficiently small $\varepsilon \geq 0$, then $\boldsymbol{z}_t$ is an ε-optimal solution to Problem (4.11), which means that

$$f^\star \leq f(\boldsymbol{z}_t) \leq f^\star + \varepsilon, \tag{4.18}$$

where $f^\star$ is the optimal objective value of Problem (4.11). In this case, we terminate the algorithm with the ε-optimal solution $\boldsymbol{z}_t$. Otherwise, we add the constraint (4.15) with $\hat{\boldsymbol{z}} = \boldsymbol{z}_t$ to the feasible region as follows:

$$\mathcal{F}_{t+1} \leftarrow \mathcal{F}_t \cap \{(\boldsymbol{z}, \theta) \in \mathcal{Z}_N^k \times \mathbb{R} \mid \theta \geq f(\boldsymbol{z}_t) + \boldsymbol{g}(\boldsymbol{z}_t)^\top(\boldsymbol{z} - \boldsymbol{z}_t)\}. \tag{4.19}$$

Note that this update cuts off the solution $(\boldsymbol{z}_t, \theta_t)$ because $\theta_t < f(\boldsymbol{z}_t)$.

We then set $t \leftarrow t+1$ and solve the surrogate upper-level problem (4.17) again with the updated feasible region (4.19). We repeat this procedure until we find an ε-optimal solution $\hat{\boldsymbol{z}}$. After termination of the algorithm, we can compute the corresponding portfolio by solving the lower-level problem (4.12) with $\boldsymbol{z} = \hat{\boldsymbol{z}}$.

Our cutting-plane algorithm is summarized by Algorithm 4.1. we can prove the finite convergence of Algorithm 4.1 as follows.

Algorithm 4.1 Cutting-plane algorithm for solving Problem (4.11)

Step 0 (Initialization) Let $\varepsilon \geq 0$ be a tolerance for optimality. Define the feasible region $\mathcal{F}_1$ as in Eq. (4.16). Set $t \leftarrow 1$ and $\mathrm{UB}_0 \leftarrow \infty$.

Step 1 (Surrogate Upper-level Problem) Solve Problem (4.17). Let $(\boldsymbol{z}_t, \theta_t)$ be an optimal solution, and set $\mathrm{LB}_t \leftarrow \theta_t$.

Step 2 (Dual Lower-level Problem) Solve Problem (4.13) with $\boldsymbol{z} = \boldsymbol{z}_t$ to calculate $f(\boldsymbol{z}_t)$ and $\boldsymbol{\omega}^\star(\boldsymbol{z}_t)$. If $f(\boldsymbol{z}_t) < \mathrm{UB}_{t-1}$, set $\mathrm{UB}_t \leftarrow f(\boldsymbol{z}_t)$ and $\hat{\boldsymbol{z}} \leftarrow \boldsymbol{z}_t$; otherwise, set $\mathrm{UB}_t \leftarrow \mathrm{UB}_{t-1}$.

Step 3 (Termination Condition) If $\mathrm{UB}_t - \mathrm{LB}_t \leq \varepsilon$, terminate the algorithm with the ε-optimal solution $\hat{\boldsymbol{z}}$.

Step 4 (Cut Generation) Calculate $\boldsymbol{g}(\boldsymbol{z}_t)$ as in Eq. (4.14) and update the feasible region as in Eq. (4.19). Set $t \leftarrow t + 1$ and return to Step 1.

Theorem 4.5. *Algorithm 4.1 terminates in a finite number of iterations and outputs an ε-optimal solution to Problem (4.11).*

Proof. Let $\{(\boldsymbol{z}_t, \theta_t) \mid t \in [T]\}$ be a sequence of solutions generated by Algorithm 4.1. Suppose that there exists $u < T$ such that $\boldsymbol{z}_u = \boldsymbol{z}_T$. Since $(\boldsymbol{z}_T, \theta_T) \in \mathcal{F}_{u+1}$, it follows from Eq. (4.19) that

$$\mathrm{LB}_T = \theta_T \geq f(\boldsymbol{z}_u) + \boldsymbol{g}(\boldsymbol{z}_u)^\top (\boldsymbol{z}_T - \boldsymbol{z}_u) = f(\boldsymbol{z}_T),$$

which verifies that $\boldsymbol{z}_T$ is an optimal solution to Problem (4.11). Since there are at most a finite number of possible solutions $\boldsymbol{z} \in \mathcal{Z}_N^k$, the algorithm terminates with an ε-optimal solution after a finite number of iterations. □

4.4 Problem reduction of the dual lower-level problem

Recall that Step 2 of Algorithm 4.1 solves Problem (4.13), which is an SDO problem including positive semidefinite constraints (4.13f) and (4.13g) on $(N+1) \times (N+1)$ symmetric matrices. It is clearly difficult to directly solve Problem (4.13) when N is very large. To remedy this situation, we reduce its problem size by applying the technique of positive semidefinite matrix completion [60, 121] to the lower-level SDO problem (4.13).

4.4.1 Reduced problem formulation

For any vector $\boldsymbol{v} := (v_n) \in \mathbb{R}^N$ and matrix $\boldsymbol{M} := (m_{nn'}) \in \mathbb{R}^{N \times N}$, we write the subvector and submatrix corresponding to $\boldsymbol{z}, \boldsymbol{z}' \in \{0,1\}^N$ as

$$\begin{aligned} \boldsymbol{v}_{\boldsymbol{z}} &:= (v_n)_{n \in \mathcal{N}(\boldsymbol{z})} = (v_n \mid z_n = 1) \in \mathbb{R}^{|\mathcal{N}(\boldsymbol{z})|}, \\ \boldsymbol{M}_{\boldsymbol{z},\boldsymbol{z}'} &:= (M_{nn'})_{(n,n') \in \mathcal{N}(\boldsymbol{z}) \times \mathcal{N}(\boldsymbol{z}')} = (M_{nn'} \mid z_n = z_{n'} = 1) \in \mathbb{R}^{|\mathcal{N}(\boldsymbol{z})| \times |\mathcal{N}(\boldsymbol{z}')|}, \end{aligned}$$

where $\mathcal{N}(\boldsymbol{z}) := \{n \in [N] \mid z_n = 1\}$.

When $\boldsymbol{z} \in \mathcal{Z}_N^k$, we have $|\mathcal{N}(\boldsymbol{z})| = k$. Then $\boldsymbol{\omega} \in \mathbb{R}^N$ can be replaced with its subvector $\boldsymbol{\omega}_{\boldsymbol{z}} \in \mathbb{R}^k$ in the objective (4.13a) as follows:

$$\frac{\gamma}{2} \boldsymbol{z}^\top (\boldsymbol{\omega} \circ \boldsymbol{\omega}) = \frac{\gamma}{2} \boldsymbol{\omega}_{\boldsymbol{z}}^\top \boldsymbol{\omega}_{\boldsymbol{z}}. \tag{4.20}$$

The cardinality parameter k is usually much smaller than N. We exploit this problem structure to reduce the size of Problem (4.13).

A reduced version of Problem (4.13) for $\boldsymbol{z} \in \mathcal{Z}_N^k$ is formulated as

$$
\begin{aligned}
f'(\boldsymbol{z}) = \underset{\boldsymbol{\omega}_{\boldsymbol{z}}, \boldsymbol{B}_{\boldsymbol{z},\boldsymbol{z}}, \boldsymbol{\beta}_{\boldsymbol{z}}, \boldsymbol{\eta}, \boldsymbol{\lambda}_{\boldsymbol{z}}, \pi}{\text{maximize}} \quad & -\frac{\gamma}{2}\boldsymbol{\omega}_{\boldsymbol{z}}^\top \boldsymbol{\omega}_{\boldsymbol{z}} - \sum_{\ell \in [L]} \eta^{(\ell)} b^{(\ell)} + \pi && \text{(4.21a)} \\
\text{subject to} \quad & \boldsymbol{\omega}_{\boldsymbol{z}} \geq \sum_{\ell \in [L]} a^{(\ell)} \boldsymbol{\beta}_{\boldsymbol{z}}^{(\ell)} + \pi \mathbf{1}, && \text{(4.21b)} \\
& \sum_{\ell \in [L]} \boldsymbol{B}_{\boldsymbol{z},\boldsymbol{z}}^{(\ell)} = \kappa_2 \hat{\boldsymbol{\Sigma}}_{\boldsymbol{z},\boldsymbol{z}} - \hat{\boldsymbol{\mu}}_{\boldsymbol{z}} \hat{\boldsymbol{\mu}}_{\boldsymbol{z}}^\top + \hat{\boldsymbol{\mu}}_{\boldsymbol{z}} \left(\sum_{\ell \in [L]} \boldsymbol{\beta}_{\boldsymbol{z}}^{(\ell)} \right)^\top + \left(\sum_{\ell \in [L]} \boldsymbol{\beta}_{\boldsymbol{z}}^{(\ell)} \right) \hat{\boldsymbol{\mu}}_{\boldsymbol{z}}^\top, && \text{(4.21c)} \\
& \sum_{\ell \in [L]} \boldsymbol{\beta}_{\boldsymbol{z}}^{(\ell)} = \boldsymbol{\lambda}_{\boldsymbol{z}} + \hat{\boldsymbol{\mu}}_{\boldsymbol{z}}, && \text{(4.21d)} \\
& 1 - \sum_{\ell \in [L]} \eta^{(\ell)} = 0, && \text{(4.21e)} \\
& \begin{pmatrix} \boldsymbol{B}_{\boldsymbol{z},\boldsymbol{z}}^{(\ell)} & \boldsymbol{\beta}_{\boldsymbol{z}}^{(\ell)} \\ \left(\boldsymbol{\beta}_{\boldsymbol{z}}^{(\ell)}\right)^\top & \eta^{(\ell)} \end{pmatrix} \succeq \boldsymbol{O} \quad (\ell \in [L]), && \text{(4.21f)} \\
& \begin{pmatrix} \hat{\boldsymbol{\Sigma}}_{\boldsymbol{z},\boldsymbol{z}} & \boldsymbol{\lambda}_{\boldsymbol{z}} \\ \boldsymbol{\lambda}_{\boldsymbol{z}}^\top & \kappa_1 \end{pmatrix} \succeq \boldsymbol{O}, && \text{(4.21g)}
\end{aligned}
$$

where $\boldsymbol{\omega}_{\boldsymbol{z}} \in \mathbb{R}^k$, $\boldsymbol{B}_{\boldsymbol{z},\boldsymbol{z}} := (\boldsymbol{B}_{\boldsymbol{z},\boldsymbol{z}}^{(\ell)}) \in \mathbb{R}^{k \times k \times L}$, $\boldsymbol{\beta}_{\boldsymbol{z}} := (\boldsymbol{\beta}_{\boldsymbol{z}}^{(\ell)}) \in \mathbb{R}^{k \times L}$, and $\boldsymbol{\lambda}_{\boldsymbol{z}} \in \mathbb{R}^k$ are reduced versions of decision variables.

In the next subsection, we verify that the reduced problem (4.21) is equivalent to the original problem (4.13) in the sense that an optimal solution to the original problem (4.13) can be recovered from an optimal solution to the reduced problem (4.21). We also prove that $f'(\boldsymbol{z})$ defined by the reduced problem (4.21) is equal to $f(\boldsymbol{z})$ defined by the original problem (4.13).

4.4.2 Equivalence of the original and reduced problems

We first prove the following lemma.

Lemma 4.6. *Let $(\boldsymbol{\omega}_{\boldsymbol{z}}, \boldsymbol{B}_{\boldsymbol{z},\boldsymbol{z}}, \boldsymbol{\beta}_{\boldsymbol{z}}, \boldsymbol{\eta}, \boldsymbol{\lambda}_{\boldsymbol{z}}, \pi)$ be a feasible solution to Problem (4.21) for $\boldsymbol{z} \in \mathcal{Z}_N^k$. We also define $\bar{\boldsymbol{\beta}} := (\bar{\boldsymbol{\beta}}^{(\ell)}) \in \mathbb{R}^{N \times L}$ as*

$$
\begin{cases}
\bar{\boldsymbol{\beta}}_{\boldsymbol{z}}^{(\ell)} := \boldsymbol{\beta}_{\boldsymbol{z}}^{(\ell)}, \\
\bar{\boldsymbol{\beta}}_{\mathbf{1}-\boldsymbol{z}}^{(\ell)} := \hat{\boldsymbol{\Sigma}}_{\mathbf{1}-\boldsymbol{z},\boldsymbol{z}} (\hat{\boldsymbol{\Sigma}}_{\boldsymbol{z},\boldsymbol{z}})^{-1} (\boldsymbol{\beta}_{\boldsymbol{z}}^{(\ell)} - \eta^{(\ell)} \hat{\boldsymbol{\mu}}_{\boldsymbol{z}}) + \eta^{(\ell)} \hat{\boldsymbol{\mu}}_{\mathbf{1}-\boldsymbol{z}}
\end{cases}
\quad (\forall \ell \in [L]). \qquad \text{(4.22)}
$$

It then follows that

$$
\kappa_2 \hat{\boldsymbol{\Sigma}} \succeq \sum_{\substack{\ell \in [L] \\ \eta^{(\ell)} > 0}} \frac{1}{\eta^{(\ell)}} (\bar{\boldsymbol{\beta}}^{(\ell)} - \eta^{(\ell)} \hat{\boldsymbol{\mu}})(\bar{\boldsymbol{\beta}}^{(\ell)} - \eta^{(\ell)} \hat{\boldsymbol{\mu}})^\top. \qquad \text{(4.23)}
$$

Proof. See Appendix A.2. □

We next prove that feasible $\boldsymbol{B}$ and $\boldsymbol{\beta}$ for the original problem (4.13) can be completed from a feasible solution to the reduced problem (4.21).

Lemma 4.7. *Let* $(\boldsymbol{\omega}_{\boldsymbol{z}}, \boldsymbol{B}_{\boldsymbol{z},\boldsymbol{z}}, \boldsymbol{\beta}_{\boldsymbol{z}}, \boldsymbol{\eta}, \boldsymbol{\lambda}_{\boldsymbol{z}}, \pi)$ *be a feasible solution to Problem* (4.21) *for* $\boldsymbol{z} \in \mathcal{Z}_N^k$, *and* $\bar{\boldsymbol{\beta}}$ *is defined by Eq.* (4.22). *Then, there exists* $\bar{\boldsymbol{B}} := (\bar{\boldsymbol{B}}^{(\ell)}) \in \mathbb{R}^{N\times N\times L}$ *such that* $(\boldsymbol{B}, \boldsymbol{\beta}) = (\bar{\boldsymbol{B}}, \bar{\boldsymbol{\beta}})$ *satisfies Eqs.* (4.13c) *and* (4.13f).

Proof. For $\ell \in [L]$ such that $\eta^{(\ell)} = 0$, we have $\boldsymbol{\beta}_{\boldsymbol{z}}^{(\ell)} = \boldsymbol{0}$ from the constraint (4.21f), thus $\bar{\boldsymbol{\beta}}^{(\ell)} = \boldsymbol{0}$ from the definition (4.22). In this case, the solution $(\bar{\boldsymbol{B}}^{(\ell)}, \bar{\boldsymbol{\beta}}^{(\ell)}, \eta^{(\ell)}) = (\boldsymbol{O}, \boldsymbol{0}, 0)$ satisfies the constraint (4.13f) and can be left out of Eq. (4.13c). Therefore, we can assume without loss of generality that $\eta^{(\ell)} > 0$ for all $\ell \in [L]$ in Eqs. (4.13c) and (4.13f).

From Eqs. (4.21e) and (4.23), we can see that

$$\kappa_2 \hat{\boldsymbol{\Sigma}} - \hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^\top + \hat{\boldsymbol{\mu}} \left(\sum_{\ell\in[L]} \bar{\boldsymbol{\beta}}^{(\ell)} \right)^\top + \left(\sum_{\ell\in[L]} \bar{\boldsymbol{\beta}}^{(\ell)} \right) \hat{\boldsymbol{\mu}}^\top \succeq \sum_{\ell\in[L]} \widetilde{\boldsymbol{B}}^{(\ell)}, \tag{4.24}$$

where

$$\widetilde{\boldsymbol{B}}^{(\ell)} := \frac{1}{\eta^{(\ell)}} \bar{\boldsymbol{\beta}}^{(\ell)} (\bar{\boldsymbol{\beta}}^{(\ell)})^\top \quad (\forall \ell \in [L]).$$

Then we have

$$\begin{pmatrix} \widetilde{\boldsymbol{B}}^{(\ell)} & \bar{\boldsymbol{\beta}}^{(\ell)} \\ (\bar{\boldsymbol{\beta}}^{(\ell)})^\top & \eta^{(\ell)} \end{pmatrix} = \frac{1}{\eta^{(\ell)}} \begin{pmatrix} \bar{\boldsymbol{\beta}}^{(\ell)} \\ \eta^{(\ell)} \end{pmatrix} \begin{pmatrix} \bar{\boldsymbol{\beta}}^{(\ell)} \\ \eta^{(\ell)} \end{pmatrix}^\top \succeq \boldsymbol{O} \quad (\forall \ell \in [L]). \tag{4.25}$$

From Eqs. (4.24) and (4.25), we can see that $\bar{\boldsymbol{B}}$ satisfying Eqs. (4.13c) and (4.13f) is obtained by adding appropriate positive semidefinite matrices to $\widetilde{\boldsymbol{B}}^{(\ell)}$ for $\ell \in [L]$. □

Now, we can prove our main theorem.

Theorem 4.8. *Let* $(\boldsymbol{\omega}_{\boldsymbol{z}}, \boldsymbol{B}_{\boldsymbol{z},\boldsymbol{z}}, \boldsymbol{\beta}_{\boldsymbol{z}}, \boldsymbol{\eta}, \boldsymbol{\lambda}_{\boldsymbol{z}}, \pi)$ *be an optimal solution to Problem* (4.21) *for* $\boldsymbol{z} \in \mathcal{Z}_N^k$, *and* $(\bar{\boldsymbol{\omega}}, \bar{\boldsymbol{\beta}}, \bar{\boldsymbol{\lambda}})$ *be defined by Eq.* (4.22) *and*

$$\begin{cases} \bar{\boldsymbol{\omega}}_{\boldsymbol{z}} := \boldsymbol{\omega}_{\boldsymbol{z}}, \\ \bar{\boldsymbol{\omega}}_{\boldsymbol{1}-\boldsymbol{z}} := \left[\sum_{\ell\in[L]} a^{(\ell)} \bar{\boldsymbol{\beta}}_{\boldsymbol{1}-\boldsymbol{z}}^{(\ell)} + \pi \boldsymbol{1} \right]_+, \end{cases} \tag{4.26}$$

$$\begin{cases} \bar{\boldsymbol{\lambda}}_{\boldsymbol{z}} := \boldsymbol{\lambda}_{\boldsymbol{z}}, \\ \bar{\boldsymbol{\lambda}}_{\boldsymbol{1}-\boldsymbol{z}} := \hat{\boldsymbol{\Sigma}}_{\boldsymbol{1}-\boldsymbol{z},\boldsymbol{z}} (\hat{\boldsymbol{\Sigma}}_{\boldsymbol{z},\boldsymbol{z}})^{-1} \boldsymbol{\lambda}_{\boldsymbol{z}}, \end{cases} \tag{4.27}$$

where $[\boldsymbol{v}]_+ := (\max\{0, v_n\}) \in \mathbb{R}^N$ *for* $\boldsymbol{v} := (v_n) \in \mathbb{R}^N$. *Then, there exists* $\bar{\boldsymbol{B}}$ *such that* $(\bar{\boldsymbol{\omega}}, \bar{\boldsymbol{B}}, \bar{\boldsymbol{\beta}}, \boldsymbol{\eta}, \bar{\boldsymbol{\lambda}}, \pi)$ *is an optimal solution to Problem* (4.13). *In addition, we have* $f(\boldsymbol{z}) = f'(\boldsymbol{z})$ *for all* $\boldsymbol{z} \in \mathcal{Z}_N^k$.

Proof. We assume without loss of generality that

$$\boldsymbol{z} = \begin{pmatrix} \boldsymbol{1} \\ \boldsymbol{0} \end{pmatrix}, \ \boldsymbol{1} \in \mathbb{R}^k, \ \boldsymbol{0} \in \mathbb{R}^{N-k}. \tag{4.28}$$

Accordingly, we partition vector $\boldsymbol{v} \in \mathbb{R}^N$ and symmetric matrix $\boldsymbol{M} \in \mathcal{S}^N$ for notational simplicity as follows:

$$\boldsymbol{v} = \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{pmatrix}, \ \boldsymbol{v}_1 = \boldsymbol{v}_{\boldsymbol{z}}, \ \boldsymbol{v}_2 = \boldsymbol{v}_{\boldsymbol{1}-\boldsymbol{z}}, \tag{4.29}$$

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{M}_{11} & \boldsymbol{M}_{12} \\ \boldsymbol{M}_{21} & \boldsymbol{M}_{22} \end{pmatrix}, \ \boldsymbol{M}_{11} = \boldsymbol{M}_{\boldsymbol{z},\boldsymbol{z}}, \ \boldsymbol{M}_{12} = \boldsymbol{M}_{21}^\top = \boldsymbol{M}_{\boldsymbol{z},\boldsymbol{1}-\boldsymbol{z}}, \ \boldsymbol{M}_{22} = \boldsymbol{M}_{\boldsymbol{1}-\boldsymbol{z},\boldsymbol{1}-\boldsymbol{z}}. \tag{4.30}$$

Suppose that $(\boldsymbol{\omega}, \boldsymbol{B}, \boldsymbol{\beta}, \boldsymbol{\eta}, \boldsymbol{\lambda}, \pi)$ is an optimal solution to Problem (4.13). Then, $(\boldsymbol{\omega}_1, \boldsymbol{B}_{11}, \boldsymbol{\beta}_1, \boldsymbol{\eta}, \boldsymbol{\lambda}_1, \pi)$ is a feasible solution to Problem (4.21), and its objective value is equal to $f(\boldsymbol{z})$ due to Eq. (4.20). This proves that $f(\boldsymbol{z}) \leq f'(\boldsymbol{z})$.

Next, we show that $f(\boldsymbol{z}) \geq f'(\boldsymbol{z})$. Let $(\boldsymbol{\omega}_1, \boldsymbol{B}_{11}, \boldsymbol{\beta}_1, \boldsymbol{\eta}, \boldsymbol{\lambda}_1, \pi)$ be an optimal solution to Problem (4.21). From the definition (4.26), the constraint (4.13b) is satisfied by $(\bar{\boldsymbol{\omega}}, \bar{\boldsymbol{\beta}}, \pi)$. Since the constraints (4.21d) and (4.21e) are satisfied, the constraint (4.13d) is satisfied by $(\bar{\boldsymbol{\beta}}, \bar{\boldsymbol{\lambda}})$ as

$$\begin{aligned}
\sum_{\ell \in [L]} \bar{\boldsymbol{\beta}}^{(\ell)} &= \begin{pmatrix} \sum_{\ell \in [L]} \boldsymbol{\beta}_1^{(\ell)} \\ \hat{\boldsymbol{\Sigma}}_{21} (\hat{\boldsymbol{\Sigma}}_{11})^{-1} \left(\sum_{\ell \in [L]} \boldsymbol{\beta}_1^{(\ell)} - \hat{\boldsymbol{\mu}}_1 \right) + \hat{\boldsymbol{\mu}}_2 \end{pmatrix} \quad \because \text{Eqs. (4.21e), (4.22)} \\
&= \begin{pmatrix} \boldsymbol{\lambda}_1 + \hat{\boldsymbol{\mu}}_1 \\ \hat{\boldsymbol{\Sigma}}_{21} (\hat{\boldsymbol{\Sigma}}_{11})^{-1} \boldsymbol{\lambda}_1 + \hat{\boldsymbol{\mu}}_2 \end{pmatrix} \quad \because \text{Eq. (4.21d)} \\
&= \bar{\boldsymbol{\lambda}} + \hat{\boldsymbol{\mu}}. \quad \because \text{Eq. (4.27)}
\end{aligned}$$

Since the constraint (4.21g) is satisfied, we can use the following Schur complement property (see, e.g., Section A.5.5 [36]):

$$\begin{pmatrix} \hat{\boldsymbol{\Sigma}}_{11} & \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_1^\top & \kappa_1 \end{pmatrix} \succeq \boldsymbol{O} \;\Rightarrow\; \kappa_1 - \boldsymbol{\lambda}_1^\top \hat{\boldsymbol{\Sigma}}_{11}^{-1} \boldsymbol{\lambda}_1 \succeq \boldsymbol{O}. \tag{4.31}$$

From Eq. (4.27), we have

$$\begin{pmatrix} \hat{\boldsymbol{\Sigma}} & \bar{\boldsymbol{\lambda}} \\ \bar{\boldsymbol{\lambda}}^\top & \kappa_1 \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{\Sigma}}_{11} & \hat{\boldsymbol{\Sigma}}_{12} & \bar{\boldsymbol{\lambda}}_1 \\ \hat{\boldsymbol{\Sigma}}_{21} & \hat{\boldsymbol{\Sigma}}_{22} & \bar{\boldsymbol{\lambda}}_2 \\ \bar{\boldsymbol{\lambda}}_1^\top & \bar{\boldsymbol{\lambda}}_2^\top & \kappa_1 \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{\Sigma}}_{11} & \hat{\boldsymbol{\Sigma}}_{12} & \boldsymbol{\lambda}_1 \\ \hat{\boldsymbol{\Sigma}}_{21} & \hat{\boldsymbol{\Sigma}}_{22} & \hat{\boldsymbol{\Sigma}}_{21} \hat{\boldsymbol{\Sigma}}_{11}^{-1} \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_1^\top & (\hat{\boldsymbol{\Sigma}}_{21} \hat{\boldsymbol{\Sigma}}_{11}^{-1} \boldsymbol{\lambda}_1)^\top & \kappa_1 \end{pmatrix},$$

which is positive semidefinite because

$$\begin{aligned}
&\begin{pmatrix} \boldsymbol{I} & \boldsymbol{O} & \boldsymbol{0} \\ \boldsymbol{O} & \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{\lambda}_1^\top \hat{\boldsymbol{\Sigma}}_{11}^{-1} & \boldsymbol{0}^\top & 1 \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{\Sigma}}_{11} & \hat{\boldsymbol{\Sigma}}_{12} & \boldsymbol{\lambda}_1 \\ \hat{\boldsymbol{\Sigma}}_{21} & \hat{\boldsymbol{\Sigma}}_{22} & \hat{\boldsymbol{\Sigma}}_{21} \hat{\boldsymbol{\Sigma}}_{11}^{-1} \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_1^\top & (\hat{\boldsymbol{\Sigma}}_{21} \hat{\boldsymbol{\Sigma}}_{11}^{-1} \boldsymbol{\lambda}_1)^\top & \kappa_1 \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{O} & -\hat{\boldsymbol{\Sigma}}_{11}^{-1} \boldsymbol{\lambda}_1 \\ \boldsymbol{O} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0}^\top & \boldsymbol{0}^\top & 1 \end{pmatrix} \\
&= \begin{pmatrix} \hat{\boldsymbol{\Sigma}}_{11} & \hat{\boldsymbol{\Sigma}}_{12} & \boldsymbol{0} \\ \hat{\boldsymbol{\Sigma}}_{21} & \hat{\boldsymbol{\Sigma}}_{22} & \boldsymbol{0} \\ \boldsymbol{0}^\top & \boldsymbol{0}^\top & \kappa_1 - \boldsymbol{\lambda}_1^\top \hat{\boldsymbol{\Sigma}}_{11}^{-1} \boldsymbol{\lambda}_1 \end{pmatrix} \succeq \boldsymbol{O}. \quad \because \text{Eq. (4.31)}
\end{aligned}$$

Therefore, the constraint (4.13g) is satisfied by $\bar{\boldsymbol{\lambda}}$.

From Lemma 4.7, the constraints (4.13c) and (4.13f) are satisfied by $(\bar{\boldsymbol{B}}, \bar{\boldsymbol{\beta}})$. Therefore, $(\bar{\boldsymbol{\omega}}, \bar{\boldsymbol{B}}, \bar{\boldsymbol{\beta}}, \boldsymbol{\eta}, \bar{\boldsymbol{\lambda}}, \pi)$ is a feasible solution to Problem (4.13), and its objective value is equal to $f'(\boldsymbol{z})$ due to Eq. (4.20). This implies that $f(\boldsymbol{z}) \geq f'(\boldsymbol{z})$. As a result, we have $f(\boldsymbol{z}) = f'(\boldsymbol{z})$, and thus, $(\bar{\boldsymbol{\omega}}, \bar{\boldsymbol{B}}, \bar{\boldsymbol{\beta}}, \boldsymbol{\eta}, \bar{\boldsymbol{\lambda}}, \pi)$ is an optimal solution to Problem (4.13). □

Remark 4.1. Note that Eq. (4.26) defines a minimum-norm solution to Problem (4.13). This solution is known to generate strong cutting planes [24].

Remark 4.2. When $\kappa_1 = 0$, unlike in Assumption 4.1, we can set $\mathbb{E}_F[\tilde{\boldsymbol{\xi}}] = \hat{\boldsymbol{\mu}}$ and delete the first condition in Eq. (4.5). In this case, we can also prove the theorem corresponding to Theorem 4.8.

From Theorem 4.8, we can revise Step 2 of Algorithm 4.1 as follows:

Step 2 (Reduced Dual Lower-level Problem) Solve Problem (4.21) with $\boldsymbol{z} = \boldsymbol{z}_t$. Let $(\boldsymbol{\omega}_{\boldsymbol{z}}, \boldsymbol{B}_{\boldsymbol{z},\boldsymbol{z}}, \boldsymbol{\beta}_{\boldsymbol{z}}, \boldsymbol{\eta}, \boldsymbol{\lambda}_{\boldsymbol{z}}, \pi)$ be an optimal solution. Calculate $f(\boldsymbol{z}_t) = f'(\boldsymbol{z}_t)$ and $\boldsymbol{\omega}^\star(\boldsymbol{z}_t) = \bar{\boldsymbol{\omega}}$ from Eqs. (4.22) and (4.26). If $f(\boldsymbol{z}_t) < \mathrm{UB}_{t-1}$, set $\mathrm{UB}_t \leftarrow f(\boldsymbol{z}_t)$ and $\hat{\boldsymbol{z}} \leftarrow \boldsymbol{z}_t$; otherwise, set $\mathrm{UB}_t \leftarrow \mathrm{UB}_{t-1}$.

Remark 4.3. Our cutting-plane algorithm can be extended to a feasible set of portfolios with additional linear constraints:

$$\mathcal{X}' := \left\{ \boldsymbol{x} \in \mathbb{R}^N \;\middle|\; \sum_{n\in[N]} x_n = 1, \quad \boldsymbol{x} \geq 0, \quad \boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{d} \right\},$$

where $\boldsymbol{C} \in \mathbb{R}^{J\times N}$ and $\boldsymbol{d} \in \mathbb{R}^J$ are given constants (see, e.g., Bertsimas and Cory-Wright [24] and Kobayashi *et al.* [95] for details).

4.5 Numerical experiments

In this section, we report numerical results to evaluate the efficacy of our method for distributionally robust portfolio optimization with the cardinality constraint. We first examine the computational performance of our cutting-plane algorithm and then demonstrate the out-of-sample investment performance of our portfolio optimization model. All experiments were performed on a Windows 10 PC with an Intel Xeon E-2274G CPU (4.00 GHz) and 32 GB of memory.

4.5.1 Computational performance

In this subsection, we evaluate the computational performance of our cutting-plane algorithm by comparison with other MISDO algorithms.

Experimental design

Table 4.1 lists the datasets used in our experiments, where N is the number of assets. From the data library on the website of Kenneth R. French [59], we downloaded two historical datasets (i.e., `ind49` and `sbm100`) of US stock returns. We used monthly data from January 2010 to December 2019 to calculate the sample estimates (4.4), $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ of asset returns. From the OR-Library [16, 41], we downloaded two datasets (i.e., `port2` and `port5`) of the sample estimates $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$, which were multiplied by 100 to be consistent with the other datasets. From Kaggle datasets [90], we downloaded a historical dataset (i.e., `s&p500`) of US stock returns. To compute the sample estimates $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$, we used daily data from February 8th, 2013 to February 7th, 2018, where 32 companies including missing values were omitted from the S&P500 index.

Table 4.1: Dataset description

Abbr.	N	Original dataset
`ind49`	49	49 Industry Portfolios [59]
`port2`	89	port2 (Portfolio optimization: Single period) [16, 41]
`sbm100`	100	100 Portfolios Formed on Size and Book-to-Market [59]
`port5`	225	port5 (Portfolio optimization: Single period) [16, 41]
`s&p500`	468	468 companies in the S&P 500 index [90]

As the utility function (4.6), we used a piecewise-linear approximation of the following normalized exponential utility function:

$$\tilde{u}(y) := \frac{\mu_{\max}(1 - \exp(-\alpha y/\mu_{\max}))}{\alpha}, \tag{4.32}$$

where $\mu_{\max}$ is the maximum entry of the sample mean vector $\hat{\boldsymbol{\mu}}$, and $\alpha > 0$ is a risk-aversion parameter. As Figure 4.1 illustrates, we set $\alpha = 10$ and employed

three tangent lines (i.e., $L = 3$) at $y \in \{0, \mu_{\max}/2, \mu_{\max}\}$ for piecewise-linear approximation.

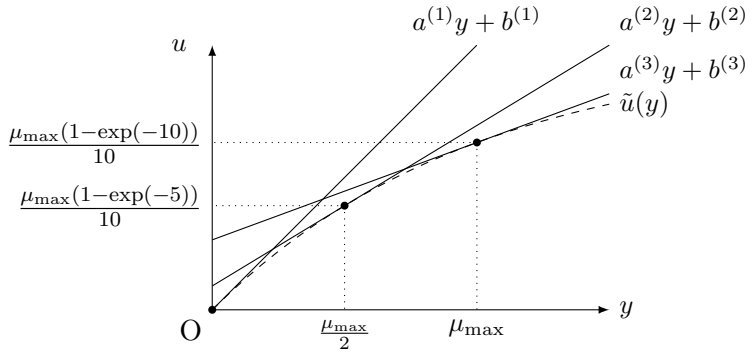


Figure 4.1: Piecewise-linear approximation of the utility function (4.32) with $L = 3$

We compare the computational performance of the following methods for solving the MISDO problem (4.10):

SCIP: MISDO solver SCIP-SDP[1] [62],

CPA: cutting-plane algorithm (Algorithm 4.1),

CPA$_+$: cutting-plane algorithm (Algorithm 4.1) using the matrix-completion-based problem reduction (i.e., Step 2 in Section 4.4.2).

These methods used Mosek[2] 9.2.40 to solve SDO problems. CPA and CPA$_+$ were implemented in Python 3.7 with Gurobi Optimizer[3] 8.1.11 to solve the surrogate upper-level problem (4.17). We also used the lazy constraint callback to add cutting planes during the branch-and-bound procedure. We set $\varepsilon = 10^{-5}$ as the tolerance for optimality. The computation of each method was terminated if it did not finish by itself within 3,600 s. In these cases, the results obtained within 3,600 s were taken as the final outcome.

We also applied the branch-and-cut algorithm in Chapter 2 for solving some instances. The preliminary experiments showed that it was very slow and failed to complete the computations even for small-sized instances. Therefore, we did not employ the branch-and-cut algorithm as a baseline in the numerical experiments.

The following column labels are used in Tables 4.2–4.4:

Obj: objective value of the obtained best feasible solution,

GAP(%): absolute difference between lower and upper bounds on the optimal objective value divided by the upper bound,

Time: computation time in seconds,

#Cuts: number of cutting planes generated by the cutting-plane algorithms,

#Nodes: number of nodes explored in the branch-and-bound procedure.

[1] http://www.opt.tu-darmstadt.de/scipsdp/
[2] https://www.mosek.com/
[3] https://www.gurobi.com/

Note that the best values of "Time" are indicated in bold for each problem instance, and those of "Obj" are also indicated in bold for the `port2` and `s&p500` datasets. Also, "OM" in the Obj column indicates that the computation did not start due to the out-of-memory condition.

Results for different values of the cardinality parameter k

Table 4.2 gives the numerical results of each method for the cardinality parameter $k \in \{5, 10, 15\}$. Here, we set the parameters $\gamma = 10/\sqrt{N}$ for the ℓ_2-regularization term and $(\kappa_1, \kappa_2) = (1, 4)$ for the uncertainty set.

First, we focus on our cutting-plane algorithms (i.e., CPA and CPA$_+$). Both CPA and CPA$_+$ failed to finish solving problem instances for the `port2` and `s&p500` datasets; however, CPA$_+$ solved other problem instances much faster than did CPA, especially for large N. In the case of the `sbm100` dataset with $k = 5$, although CPA was terminated due to the time limit, CPA$_+$ solved the problem instance completely in 1.9 s. These results verify the effectiveness of our matrix-completion-based problem reduction of the dual lower-level problem.

Next, we compare our cutting-plane algorithms with the MISDO solver SCIP. CPA$_+$ was the fastest when it finished computations within the time limit. SCIP returned an incorrect optimal objective value for the dataset `ind49` with $k \in \{10, 15\}$ due to the numerical instability. For the `port2` dataset, all three methods failed to complete the computations within the time limit, but CPA$_+$ found solutions of better quality than did SCIP and CPA for $k \in \{10, 15\}$.

For the `port5` dataset, SCIP did not finish solving even a first continuous relaxation problem within the time limit, and thus, failed to provide a feasible solution for all $k \in \{5, 10, 15\}$. CPA did not start computations due to the out-of-memory condition. In contrast, CPA$_+$ succeeded in computing solutions with guaranteed optimality for all $k \in \{5, 10, 15\}$.

For the `s&p500` dataset involving 468 investable assets, only CPA$_+$ worked normally to find feasible solutions. Optimality gaps attained by CPA$_+$ for $k \in \{5, 10, 15\}$ are 21.0%, 19.1%, and 9.9%, respectively, and this gap is expected to be smaller if longer time can be spent on the computation. These results support the potential of CPA$_+$ to give good-quality solutions to large problem instances with a limited memory capacity.

The computation time of CPA$_+$ tended to be shorter for larger k. For the `port5` dataset, CPA$_+$ finished the computations in 316.2 s, 84.6 s, and 16.16 s for $k \in \{5, 10, 15\}$, respectively. Here, both #Cuts and #Nodes were much smaller for $k = 15$ than for $k = 5$, which is part of the reason why CPA$_+$ got to be faster with larger k.

Results for different values of the uncertainty parameters (κ_1, κ_2)

Table 4.3 gives the numerical results of each method for the pair of uncertainty parameters $(\kappa_1, \kappa_2) \in \{(0.5, 2), (1, 4), (2, 8)\}$. Here, we set the parameters $k = 10$ for the cardinality constraint and $\gamma = 10/\sqrt{N}$ for the ℓ_2-reguralization term. Table 4.3 shows that CPA$_+$ consistently outperformed other methods regardless of values of the uncertainty parameters (κ_1, κ_2). For the `port2` dataset, all methods failed to complete the computations within the time limit for all (κ_1, κ_2), but CPA$_+$ found solutions of better quality than did other methods for $(\kappa_1, \kappa_2) \in \{(1, 4), (2, 8)\}$. Moreover, the computation time of CPA$_+$ was stable against the change in (κ_1, κ_2). Indeed, for the `port5` dataset, CPA$_+$ finished computations in 60.9 s, 84.6 s, and 62.6 s for $(\kappa_1, \kappa_2) \in \{(0.5, 2), (1, 4), (2, 8)\}$,

respectively. In addition, for the `s&p500` dataset, optimality gaps given by CPA$_+$ were 17.9%, 19.1%, and 20.1% for $(\kappa_1, \kappa_2) \in \{(0.5, 2), (1, 4), (2, 8)\}$, respectively. These results suggest that the computational performance of the CPA$_+$ is not greatly affected by the uncertainty parameters (κ_1, κ_2).

Results for different values of the ℓ_2-regularization parameter γ

Table 4.4 gives the numerical results of each method for the ℓ_2-regularization parameter $\gamma \in \{1/\sqrt{N}, 10/\sqrt{N}, 100/\sqrt{N}\}$. Here, we set the parameters $k = 10$ for the cardinality constraint and $(\kappa_1, \kappa_2) = (1, 4)$ for the uncertainty set. Table 4.4 shows that CPA$_+$ performed very well compared to other methods. CPA$_+$ was always faster than other methods for the `ind49`, `sbm100`, and `port5` datasets. For the `port2` dataset, all methods failed to solve problem instances to optimality, but CPA$_+$ found solutions of better quality than did other methods except for $\gamma = 100/\sqrt{N}$. For the `s&p500` dataset, only CPA$_+$ provided feasible solutions without causing an out-of-memory condition. The computation time of CPA$_+$ tended to be shorter for smaller γ. For the `port5` dataset, CPA$_+$ finished the computations in 66.0 s, 84.6 s, and 417.1 s for $\gamma \in \{1/\sqrt{N}, 10/\sqrt{N}, 100/\sqrt{N}\}$, respectively. A similar tendency was also shown by the results for the `s&p500` dataset, where optimality gaps obtained by CPA$_+$ were 4.2%, 19.1%, and 79.4% for $\gamma \in \{1/\sqrt{N}, 10/\sqrt{N}, 100/\sqrt{N}\}$, respectively.

4.5.2 Out-of-sample investment performance

Finally, we investigate the investment performance of our cardinality-constrained distributionally robust portfolio optimization model in a practical situation.

Experimental design

For the purpose of comparison, we consider the following cardinality-constrained mean-variance portfolio optimization model:

$$\begin{aligned}
\underset{\boldsymbol{x}}{\text{minimize}} \quad & \boldsymbol{x}^\top \hat{\boldsymbol{\Sigma}} \boldsymbol{x} && \text{(4.33a)}\\
\text{subject to} \quad & \hat{\boldsymbol{\mu}}^\top \boldsymbol{x} \geq \bar{r}, && \text{(4.33b)}\\
& z_n = 0 \;\Rightarrow\; x_n = 0 \quad (\forall n \in [N]), && \text{(4.33c)}\\
& \boldsymbol{x} \in \mathcal{X}, \quad \boldsymbol{z} \in \mathcal{Z}_N^k, && \text{(4.33d)}
\end{aligned}$$

where $\bar{r}$ is a user-defined parameter of the required return level.

We compare the out-of-sample investment performance of portfolios determined by the following optimization models:

MV: cardinality-constrained mean-variance optimization model (4.33), which was solved using Gurobi Optimizer 8.1.11;

DR: cardinality-constrained distributionally robust optimization model (4.10), which was solved by our cutting-plane algorithm CPA$_+$.

For the MV model, we set the required return level $\bar{r}$ to the first quartile of entries of the sample mean $\hat{\boldsymbol{\mu}}$. For the DR model, we tuned the uncertainty parameters (κ_1, κ_2) based on the 80% confidence region estimated by the tailored bootstrap method [29] with the bootstrap sample size of 100,000. We set $\gamma = 10/\sqrt{N}$ for the ℓ_2-regularization term, and the utility function (4.6) in the same way as in Section 4.5.1.

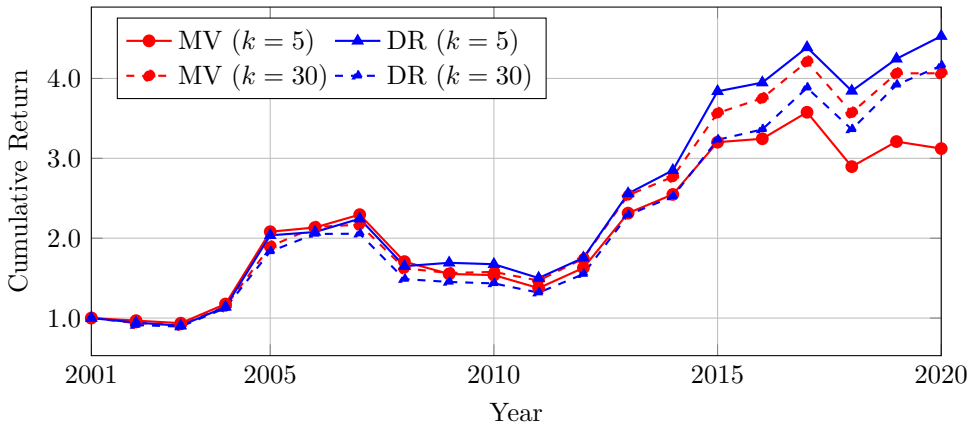


Figure 4.2: Evolution of cumulative total returns over the years 2000–2020

We evaluated the out-of-sample investment performance based on a rolling horizon strategy. From Yahoo! Finance [167], we downloaded weekly data of Japanese stock returns from January 2000 to December 2020, where top 30 companies were selected from the Nikkei 225 index according to market capitalization as of December 2020. We solved portfolio optimization models by calculating sample estimates $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ in the first training period (156 weeks). We next calculated weekly returns by applying the obtained portfolios to the subsequent testing period (52 weeks). We repeated this process at intervals of 52 weeks until the end of the entire data period.

We evaluate the out-of-sample investment performance based on the following cumulative total return:

$$\prod_{m \in [M]} (1 + R_m),$$

where R_m is the rate of portfolio return in the mth week during the testing period M.

Results of the investment performance

Figure 4.2 shows the evolution of (out-of-sample) cumulative total returns produced by each optimization model with the cardinality parameter $k \in \{5, 30\}$. The highest investment performance was achieved by the DR model with $k = 5$, which suggests that the out-of-sample investment performance can be improved by the interplay between the distributionally robust optimization and the cardinality constraint. In contrast, the investment performance of the MV model was lower with $k = 5$ than with $k = 30$ probably because portfolio diversification is prevented by the cardinality constraint.

A notable result is the investment performance in 2020, when the stock market was significantly affected by the spread of COVID-19. In this year, while the cumulative return was not increased by the MV model with $k \in \{5, 30\}$, that was improved by the DR model with $k \in \{5, 30\}$. This implies that distributionally robust optimization models are helpful in coping with an unexpected situation. These results demonstrate the practical effectiveness of our cardinality-constrained distributionally robust portfolio optimization model in terms of the out-of-sample investment performance.

4.6 Conclusion

In this chapter, we consider moment-based distributionally robust portfolio optimization problems [47] with the cardinality constraint. Due to the discreteness, this problem is formulated mixed-integer semidefinite optimization problem, which is very hard to exactly solve when the number of investable assets is large. We reformulated the problem as a bilevel optimization problem and devised a specialized cutting-plane algorithm for solving the upper-level problem. In addition, we apply the matrix completion [60, 121] to the lower-level problem to efficiently generate cutting planes.

The computational results indicate that our cutting-plane algorithm was very effective, especially when the number of investable assets is large. Our algorithm succeeded in obtaining an optimal solution within 3,600 seconds, to a problem including 225 assets. For a large-sized problem instance with 468 assets, while the existing method did not start its computation due to lack of memory, our algorithm attained a solution of good quality. In addition, the out-of-sample investment performances given by the cardinality-constrained distributionally robust model outperformed the cardinality-constrained mean-variance model.

Table 4.2: Numerical results with $\gamma = 10/\sqrt{N}$ and $(\kappa_1, \kappa_2) = (1, 4)$ for $k \in \{5, 10, 15\}$

Data	N	k	Method	Obj	Gap(%)	Time	#Cuts	#Nodes
ind49	49	5	SCIP	3.108	0.0	627.9	—	61
			CPA	3.108	2.2	>3600	>195	>940
			CPA$_+$	3.108	0.0	**26.5**	182	1047
		10	SCIP	3.036	0.0	303.2	—	17
			CPA	3.034	0.0	400.6	22	20
			CPA$_+$	3.034	0.0	**9.3**	22	20
		15	SCIP	3.037	0.0	408.6	—	43
			CPA	3.033	0.0	107.8	6	0
			CPA$_+$	3.033	0.0	**5.2**	6	0
port2	86	5	SCIP	**1.951**	15.1	>3600	—	>27
			CPA	2.181	374.1	>3600	>23	>71
			CPA$_+$	2.071	68.9	>3600	>20640	>196390
		10	SCIP	1.761	8.7	>3600	—	>20
			CPA	1.924	188.6	>3600	>20	>39
			CPA$_+$	**1.727**	51.6	>3600	>7937	>147890
		15	SCIP	1.749	8.0	>3600	—	>20
			CPA	2.009	98.1	>3600	>23	>87
			CPA$_+$	**1.635**	17.5	>3600	>4014	>76327
sbm100	100	5	SCIP	4.494	12.1	>3600	—	>5
			CPA	3.940	0.3	>3600	>12	>1
			CPA$_+$	3.940	0.0	**1.9**	13	8
		10	SCIP	3.955	0.4	>3600	—	>10
			CPA	3.935	0.0	1753.4	5	0
			CPA$_+$	3.935	0.0	**2.0**	5	0
		15	SCIP	3.952	0.3	>3600	—	>9
			CPA	3.935	0.0	1910.1	5	0
			CPA$_+$	3.935	0.0	**4.4**	5	0
port5	225	5	SCIP	∞	100.0	>3600	—	>1
			CPA	OM	—	—	—	—
			CPA$_+$	2.812	0.0	**316.2**	770	9402
		10	SCIP	∞	100.0	>3600	—	>1
			CPA	OM	—	—	—	—
			CPA$_+$	2.687	0.0	**84.6**	191	2077
		15	SCIP	∞	100.0	>3600	—	>1
			CPA	OM	—	—	—	—
			CPA$_+$	2.677	0.0	**16.6**	18	37
s&p500	468	5	SCIP	OM	—	—	—	—
			CPA	OM	—	—	—	—
			CPA$_+$	**1.051**	21.0	>3600	>9794	>477699
		10	SCIP	OM	—	—	—	—
			CPA	OM	—	—	—	—
			CPA$_+$	**0.910**	19.1	>3600	>1996	>118070
		15	SCIP	OM	—	—	—	—
			CPA	OM	—	—	—	—
			CPA$_+$	**0.830**	9.9	>3600	>3435	>270889

Table 4.3: Numerical results with $k = 10$ and $\gamma = 10/\sqrt{N}$ for $(\kappa_1, \kappa_2) \in \{(0.5, 2), (1, 4), (2, 8)\}$.

Data	N	(κ_1, κ_2)	Method	Obj	Gap(%)	Time	#Cuts	#Nodes
`ind49`	49	(0.5,2)	SCIP	1.948	0.0	476.6	—	29
			CPA	1.946	0.0	409.2	22	20
			CPA$_+$	1.946	0.0	**9.3**	22	20
		(1,4)	SCIP	3.036	0.0	303.2	—	17
			CPA	3.034	0.0	400.6	22	20
			CPA$_+$	3.034	0.0	**9.3**	22	20
		(2,8)	SCIP	4.599	0.0	289.6	—	19
			CPA	4.588	0.0	323.3	18	8
			CPA$_+$	4.588	0.0	**7.5**	18	8
`port2`	86	(0.5,2)	SCIP	**1.156**	5.9	>3600	—	>22
			CPA	1.336	135.1	>3600	>22	>11
			CPA$_+$	1.177	34.7	>3600	>8448	>158415
		(1,4)	SCIP	1.761	8.7	>3600	—	>20
			CPA	1.924	188.6	>3600	>20	>39
			CPA$_+$	**1.727**	51.6	>3600	>7937	>147890
		(2,8)	SCIP	2.549	7.8	>3600	—	>21
			CPA	2.766	255.2	>3600	>20	>59
			CPA$_+$	**2.484**	61.7	>3600	>8415	>139514
`sbm100`	100	(0.5,2)	SCIP	2.612	1.3	>3600	—	>8
			CPA	2.575	0.0	1929.4	5	0
			CPA$_+$	2.575	0.0	**2.0**	5	0
		(1,4)	SCIP	3.955	0.4	>3600	—	>10
			CPA	3.935	0.0	1753.4	5	0
			CPA$_+$	3.935	0.0	**2.0**	5	0
		(2,8)	SCIP	5.910	0.3	>3600	—	>10
			CPA	5.874	0.0	2169.4	6	0
			CPA$_+$	5.874	0.0	**2.0**	5	0
`port5`	225	(0.5,2)	SCIP	∞	100.0	>3600	—	>1
			CPA	OM	—	—	—	—
			CPA$_+$	1.915	0.0	**60.9**	144	2177
		(1,4)	SCIP	∞	100.0	>3600	—	>1
			CPA	OM	—	—	—	—
			CPA$_+$	2.687	0.0	**84.6**	191	2077
		(2,8)	SCIP	∞	100.0	>3600	—	>1
			CPA	OM	—	—	—	—
			CPA$_+$	3.776	0.0	**62.6**	142	1486
`s&p500`	468	(0.5,2)	SCIP	OM	—	—	—	—
			CPA	OM	—	—	—	—
			CPA$_+$	**0.659**	17.9	>3600	>648	>2558
		(1,4)	SCIP	OM	—	—	—	—
			CPA	OM	—	—	—	—
			CPA$_+$	**0.910**	19.1	>3600	>1996	>118070
		(2,8)	SCIP	OM	—	—	—	—
			CPA	OM	—	—	—	—
			CPA$_+$	**1.238**	20.1	>3600	>5944	>364115

Table 4.4: Numerical results with $k = 10$ and $(\kappa_1, \kappa_2) = (1, 4)$ for $\gamma \in \{1/\sqrt{N}, 10/\sqrt{N}, 100/\sqrt{N}\}$.

Data	N	γ	Method	Obj	Gap(%)	Time	#Cuts	#Nodes
ind49	49	$1/\sqrt{N}$	SCIP	3.415	0.0	433.2	—	30
			CPA	3.415	0.0	339.8	19	18
			CPA$_+$	3.415	0.0	**8.2**	19	18
		$10/\sqrt{N}$	SCIP	3.036	0.0	303.2	—	17
			CPA	3.034	0.0	400.6	22	20
			CPA$_+$	3.034	0.0	**9.3**	22	20
		$100/\sqrt{N}$	SCIP	2.984	0.0	323.9	—	20
			CPA	2.984	0.0	398.4	22	37
			CPA$_+$	2.984	0.0	**10.0**	22	42
port2	86	$1/\sqrt{N}$	SCIP	**2.154**	15.3	>3600	—	>22
			CPA	2.251	43.2	>3600	>23	>159
			CPA$_+$	**2.154**	6.2	>3600	>7591	>208109
		$10/\sqrt{N}$	SCIP	1.761	8.7	>3600	—	>20
			CPA	1.924	188.6	>3600	>20	>39
			CPA$_+$	**1.727**	51.6	>3600	>7937	>147890
		$100/\sqrt{N}$	SCIP	**1.699**	6.8	>3600	—	>18
			CPA	2.143	2805.9	>3600	>21	>131
			CPA$_+$	1.711	459.1	>3600	>7487	>82148
sbm100	100	$1/\sqrt{N}$	SCIP	5.029	9.2	>3600	—	>7
			CPA	4.594	0.0	2599.2	7	0
			CPA$_+$	4.594	0.0	**2.9**	7	0
		$10/\sqrt{N}$	SCIP	3.955	0.4	>3600	—	>10
			CPA	3.935	0.0	1753.4	5	0
			CPA$_+$	3.935	0.0	**2.0**	5	0
		$100/\sqrt{N}$	SCIP	3.806	0.0	2094.4	—	3
			CPA	3.801	0.0	2183.0	6	0
			CPA$_+$	3.801	0.0	**2.1**	5	0
port5	225	$1/\sqrt{N}$	SCIP	∞	100.0	>3600	—	>1
			CPA	OM	—	—	—	—
			CPA$_+$	3.380	0.0	**66.0**	144	2007
		$10/\sqrt{N}$	SCIP	∞	100.0	>3600	—	>1
			CPA	OM	—	—	—	—
			CPA$_+$	2.687	0.0	**84.6**	191	2077
		$100/\sqrt{N}$	SCIP	∞	100.0	>3600	—	>1
			CPA	OM	—	—	—	—
			CPA$_+$	2.611	0.0	**417.1**	968	8718
s&p500	468	$1/\sqrt{N}$	SCIP	OM	—	—	—	—
			CPA	OM	—	—	—	—
			CPA$_+$	**1.861**	4.2	>3600	>5452	>456468
		$10/\sqrt{N}$	SCIP	OM	—	—	—	—
			CPA	OM	—	—	—	—
			CPA$_+$	**0.910**	19.1	>3600	>1996	>118070
		$100/\sqrt{N}$	SCIP	OM	—	—	—	—
			CPA	OM	—	—	—	—
			CPA$_+$	**0.789**	79.4	>3600	>6580	>239418

Chapter 5

Bilevel Cutting-plane Algorithm for Cardinality-constrained Mean-CVaR Portfolio Optimization

In this chapter, we examine cardinality-constrained conditional value-at-risk (CVaR) minimization problems and extend the cutting-plane algorithm for the distributionally-robust portfolio optimization problem discussed in Chapter 4. While the cardinality-constrained CVaR minimization problem is formulated as an MILO problem, which can be handled by state-of-the-art MIO solvers, the problem size of CVaR minimization problems depends not only on the number of investable assets but also on the number of asset return scenarios. Thus, the computational efficiency decreases when the number of scenarios is large. To overcome this challenge, we propose a specialized cutting-plane algorithm named the *bilevel cutting-plane algorithm* for exactly solving the cardinality-constrained mean-CVaR portfolio optimization problem. Our proposal extends the cutting-plane algorithm discussed in Chapter 4 so that it incorporates another cutting-plane algorithm that efficiently minimizes CVaR. The content of this chapter is included in Kobayashi *et al.* [94].

We give an introduction to this study and summarize our contribution in Section 5.1. In Section 5.2, we show the definition of CVaR and formulate the cardinality-constrained mean-CVaR portfolio optimization problem as an MIO problem. In Section 5.3, we describe our bilevel cutting-plane algorithm for solving the problem. We report computational results in Section 5.4 and conclude in Section 5.5.

5.1 Introduction

This chapter focuses on mean-risk portfolio optimization models using the conditional value-at-risk (CVaR) [137, 138] as a risk measure. CVaR is a downside risk measure for evaluating a potential heavy loss. It is known as a coherent risk measure that has the desirable properties of monotonicity, translation invariance, positive homogeneity, and subadditivity [12, 129]. Additionally, CVaR is monotonic with respect to second-order stochastic dominance [129], which means that CVaR minimization is consistent with the preference of any rational risk-averse decision-maker. In the standard formulation based on sample average approximation [137, 138, 145], many scenarios are required to approximate CVaR accurately [91, 150]. To resolve this computational difficulty, various efficient algorithms have been proposed, including the dual solution method [125], nonsmooth optimization algorithms [17, 85, 104, 137], the factor model [100], cutting-plane algorithms [4, 80, 103, 149], the level method [52], smoothing methods [6, 157],

and successive regression approximations [3].

We consider solving mean-CVaR portfolio optimization problems with a cardinality constraint for limiting the number of invested assets. Because of the cardinality constraint, this model is formulated as an MILO problem, which is a special case of MISDO problems. While state-of-the-art MIO solvers such as Gurobi and CPLEX can be applied to the problem, the problem size of the cardinality-constrained mean-CVaR model depends not only on the number of investable assets but also on the number of asset return scenarios. When we aim to approximate CVaR accurately [91, 150], sufficiently many scenarios are required, which decreases computational efficiency even if we use the state-of-the-art MIO solvers. For this problem, some heuristic optimization algorithms have been developed, including continuous-relaxation-based heuristics [9], the ℓ_1-norm-based approximation of the cardinality constraint [43], and a Scholtes-type regularization method for complementarity constraints [37]. However, these algorithms cannot guarantee global optimality of obtained solutions.

In this chapter, we extend the cutting-plane algorithm for distributionally robust portfolio optimization in Chapter 4 and propose a high-performance algorithm named the *bilevel cutting-plane algorithm* for exactly solving the cardinality-constrained mean-CVaR portfolio optimization problem. Unlike the distributionally-robust portfolio optimization problems discussed in Chapter 4, the size of the lower-level problem of the mean-CVaR model depends on not only the number of investable assets but also that of scenarios, which degrades the computational performance of the algorithm. To speed up the computations of the lower-level problem, we apply the cutting-plane algorithm [4, 80, 103, 149] that was developed for efficiently minimizing CVaR. As a result, these two types of cutting-plane algorithms are integrated into our bilevel cutting-plane algorithm to achieve faster computations. To our knowledge, we are the first to develop an effective algorithm for exactly solving cardinality-constrained mean-CVaR portfolio optimization problems.

We conducted numerical experiments using some benchmark datasets [16, 41, 59, 90] to evaluate the efficiency of our bilevel cutting-plane algorithm. Numerical results demonstrate that our algorithm is faster than other MIO approaches, especially for large problem instances. Remarkably, our algorithm attained an optimal solution within 3600 s to a problem involving 225 assets and 100,000 scenarios.

5.2 Problem formulation

In this section, we formulate the cardinality-constrained mean-CVaR portfolio optimization problem that we consider in this chapter.

5.2.1 Conditional value-at-risk

As we did in Chapter 4, we define $\boldsymbol{x} := (x_1, x_2, \ldots, x_N)^\top$ to be a portfolio satisfying

$$\sum_{n\in[N]} x_n = 1, \quad \boldsymbol{x} \geq 0,$$

where x_n is the investment weight of the nth asset. We consider a probability space $(\Omega, \mathcal{F}, \mathcal{P})$. Let $\tilde{\boldsymbol{\xi}} : \Omega \to \mathbb{R}^N$ be an $\mathcal{F}$-measurable function (N-dimensional random vector) representing the rate of random return of each asset, and $G(\boldsymbol{r}) := \mathcal{P}(\{\omega \in \Omega \mid \tilde{\boldsymbol{\xi}}(\omega) \leq \boldsymbol{r}\})$ be the corresponding cumulative distribution function of

the asset return vector $\boldsymbol{r} \in \mathbb{R}^N$. In this chapter, we define the loss function as the negative of the portfolio net return:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{r}) := -\boldsymbol{r}^\top \boldsymbol{x}.$$

Let $\beta \in (0, 1)$ be a probability level parameter, which is frequently set close to one. Then, β-CVaR can be regarded as the approximate conditional expectation of a random loss exceeding the β-value-at-risk (β-VaR). To calculate CVaR, we use the following function:

$$F_\beta(a, \boldsymbol{x}) := a + \frac{1}{1-\beta} \int_{\mathbb{R}^N} [\mathcal{L}(\boldsymbol{x}, \boldsymbol{r}) - a]_+ \, \mathrm{d}G(\boldsymbol{r}), \tag{5.1}$$

where a is an auxiliary decision variable, and $[v]_+$ is a positive part of v (i.e., $[v]_+ := \max\{0, v\}$). Then, β-CVaR minimization is posed as follows [137, 138]:

$$\underset{a, \boldsymbol{x}}{\text{minimize}} \;\; F_\beta(a, \boldsymbol{x}) \quad \text{subject to} \;\; \boldsymbol{x} \in \mathcal{X},$$

where $\mathcal{X}$ is a set of feasible portfolios. Note that an optimal solution of a corresponds to β-VaR, which is a β-quantile of the probability distribution of the portfolio loss $\mathcal{L}(\boldsymbol{x}, \tilde{\boldsymbol{\xi}})$.

Because multiple integration in Eq. (5.1) is computationally expensive, the sample average approximation [137, 138, 145] is commonly used. Note also that even when the number of scenarios is small, the cardinality constraint helps a solution based on the sample average approximation to nearly converge to the exact solution with high probability [2, 105].

Accordingly, we consider a finite sample space $\Omega := \{\omega_s \mid s \in [S]\}$, where S is the number of scenarios. Here, $\boldsymbol{r}^{(s)} := \tilde{\boldsymbol{\xi}}(\omega_s)$ is the sth scenario of asset returns for $s \in [S]$, and $\boldsymbol{p} := (p_1, p_2, \ldots, p_S)^\top$ is a vector composed of occurrence probabilities $p_s := \mathcal{P}(\{\omega_s\})$. We then have

$$F_\beta(a, \boldsymbol{x}) \approx a + \frac{1}{1-\beta} \sum_{s \in [S]} p_s \left[-(\boldsymbol{r}^{(s)})^\top \boldsymbol{x} - a \right]_+ .$$

5.2.2 Portfolio optimization model

Throughout this chapter, we consider the following set of feasible portfolios:

$$\mathcal{X} := \{\boldsymbol{x} \in \mathbb{R}^N \mid \boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{d}, \; \mathbf{1}^\top \boldsymbol{x} = 1, \; \boldsymbol{x} \geq \boldsymbol{0}\},$$

where $\boldsymbol{C} \in \mathbb{R}^{M \times N}$ and $\boldsymbol{d} \in \mathbb{R}^M$ are given. It is supposed that the linear constraint $\boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{d}$ contains the expected return constraint:

$$\hat{\boldsymbol{\mu}}^\top \boldsymbol{x} \geq \bar{\mu}, \tag{5.2}$$

where $\hat{\boldsymbol{\mu}} := (\hat{\mu}_1, \hat{\mu}_2, \ldots, \hat{\mu}_N)^\top$ is a vector composed of expected returns of assets, and $\bar{\mu}$ is a parameter of the required return level.

As we did in Chapter 4, we also impose a cardinality constraint on the portfolio $\boldsymbol{x}$:

$$\|\boldsymbol{x}\|_0 \leq k, \tag{5.3}$$

where $k \geq 1$ be a user-defined parameter for limiting the cardinality and $\|\cdot\|_0$ is the ℓ_0-norm (i.e., the number of nonzero entries) [27, 34, 128]. Let $\boldsymbol{z} := (z_1, z_2, \ldots, z_N)^\top$ be a vector of binary decision variables for selecting assets; that

is, $z_n = 1$ if the nth asset is selected, and $z_n = 0$ otherwise. We also introduce the feasible set corresponding to the cardinality constraint (5.3):

$$\mathcal{Z}_N^k := \left\{ \boldsymbol{z} \in \{0,1\}^N \;\middle|\; \sum_{n\in[N]} z_n = k \right\}.$$

In line with previous studies [24, 48, 70, 71], we incorporate the ℓ_2-regularization term into the objective from a perspective of robust optimization. The cardinality-constrained mean-CVaR portfolio optimization model is formulated as follows:

$$\begin{aligned}
\underset{a,v,\boldsymbol{x},\boldsymbol{z}}{\text{minimize}} \quad & \frac{1}{2\gamma}\boldsymbol{x}^\top\boldsymbol{x} + a + v && \text{(5.4a)}\\
\text{subject to} \quad & v \geq \frac{1}{1-\beta}\sum_{s\in[S]} p_s \left[-(\boldsymbol{r}^{(s)})^\top\boldsymbol{x} - a\right]_+, && \text{(5.4b)}\\
& z_n = 0 \;\Rightarrow\; x_n = 0 \quad (\forall n \in [N]), && \text{(5.4c)}\\
& \boldsymbol{x} \in \mathcal{X}, \quad \boldsymbol{z} \in \mathcal{Z}_N^k, && \text{(5.4d)}
\end{aligned}$$

where v is an auxiliary decision variable, and $\gamma > 0$ is a regularization parameter. Throughout this chapter, we assume that Problem (5.4) is feasible.

5.3 Cutting-plane algorithms

In this section, we present our bilevel cutting-plane algorithm for solving the cardinality-constrained mean-CVaR portfolio optimization problem (5.4).

5.3.1 Bilevel optimization reformulation

Here, we extend the method of bilevel optimization reformulation in Chapter 4 to the mean-CVaR portfolio optimization problem (5.4). Let us define $\boldsymbol{Z} := \mathrm{Diag}(\boldsymbol{z})$ as a diagonal matrix whose diagonal entries are given by $\boldsymbol{z}$. We first eliminate the logical implication (5.4c) by replacing $\boldsymbol{x}$ with $\boldsymbol{Z}\boldsymbol{x}$ in Problem (5.4). We next reformulate Problem (5.4) as a bilevel optimization problem. Specifically, the *upper-level problem* is posed as the following integer optimization problem:

$$\underset{\boldsymbol{z}}{\text{minimize}} \;\; f(\boldsymbol{z}) \quad \text{subject to} \;\; \boldsymbol{z} \in \mathcal{Z}_N^k, \tag{5.5}$$

and its objective function is defined by the following *lower-level problem*:

$$\begin{aligned}
f(\boldsymbol{z}) = \underset{a,v,\boldsymbol{x}}{\text{minimize}} \quad & \frac{1}{2\gamma}\boldsymbol{x}^\top\boldsymbol{x} + a + v && \text{(5.6a)}\\
\text{subject to} \quad & v \geq \frac{1}{1-\beta}\sum_{s\in[S]} p_s \left[-(\boldsymbol{r}^{(s)})^\top\boldsymbol{Z}\boldsymbol{x} - a\right]_+, && \text{(5.6b)}\\
& \boldsymbol{Z}\boldsymbol{x} \in \mathcal{X}. && \text{(5.6c)}
\end{aligned}$$

Note that $(\boldsymbol{Z}\boldsymbol{x})^\top\boldsymbol{Z}\boldsymbol{x}$ has been replaced with $\boldsymbol{x}^\top\boldsymbol{x}$ in the objective (5.6a); this is because $\boldsymbol{x} = \boldsymbol{Z}\boldsymbol{x}$ holds after minimization (5.6a) as in Bertsimas and Cory-Wright [24].

We then derive cutting planes for approximating $f(\boldsymbol{z})$ in the upper-level problem (5.5) by exploiting the dual formulation of the lower-level problem (5.6). For this purpose, Constraint (5.6b), which is nonlinear and nondifferentiable, needs to be transformed into tractable constraints. A commonly used method is the

lifting representation [52, 137], which converts Problem (5.6) into the following optimization problem with linear constraints:

$$
\begin{aligned}
f(\boldsymbol{z}) = \underset{a,\boldsymbol{q},v,\boldsymbol{x}}{\text{minimize}} \quad & \frac{1}{2\gamma}\boldsymbol{x}^\top \boldsymbol{x} + a + v && && \text{(5.7a)}\\
\text{subject to} \quad & v \geq \frac{1}{1-\beta}\sum_{s\in[S]} p_s q_s, && && \text{(5.7b)}\\
& q_s \geq -(\boldsymbol{r}^{(s)})^\top \boldsymbol{Z}\boldsymbol{x} - a && (\forall s \in [S]), && \text{(5.7c)}\\
& q_s \geq 0 && (\forall s \in [S]), && \text{(5.7d)}\\
& \boldsymbol{Z}\boldsymbol{x} \in \mathcal{X}, && && \text{(5.7e)}
\end{aligned}
$$

where $\boldsymbol{q} := (q_1, q_2, \ldots, q_S)^\top$ is a vector of auxiliary decision variables. The following theorem gives the dual formulation of Problem (5.7).

Theorem 5.1. *Suppose that Problem (5.7) is feasible with $\boldsymbol{z} \in \{0,1\}^N$. Then, the strong duality holds, and the dual formulation of Problem (5.7) is represented as follows:*

$$
\begin{aligned}
f(\boldsymbol{z}) = \underset{\boldsymbol{\alpha},\boldsymbol{\zeta},\lambda,\boldsymbol{\omega}}{\text{maximize}} \quad & -\frac{\gamma}{2}\boldsymbol{z}^\top(\boldsymbol{\omega}\circ\boldsymbol{\omega}) - \boldsymbol{d}^\top\boldsymbol{\zeta} + \lambda && && \text{(5.8a)}\\
\text{subject to} \quad & \boldsymbol{\omega} \geq \sum_{s\in[S]} \alpha_s \boldsymbol{r}^{(s)} - \boldsymbol{C}^\top\boldsymbol{\zeta} + \lambda\boldsymbol{1}, && && \text{(5.8b)}\\
& \sum_{s\in[S]} \alpha_s = 1, && && \text{(5.8c)}\\
& \alpha_s \leq \frac{p_s}{1-\beta} && (\forall s \in [S]), && \text{(5.8d)}\\
& \boldsymbol{\alpha} \geq \boldsymbol{0}, \quad \boldsymbol{\zeta} \geq \boldsymbol{0}, && && \text{(5.8e)}
\end{aligned}
$$

where $\boldsymbol{\alpha} := (\alpha_1, \alpha_2, \ldots, \alpha_S)^\top$, $\boldsymbol{\zeta} \in \mathbb{R}^M$, $\lambda \in \mathbb{R}$, and $\boldsymbol{\omega} \in \mathbb{R}^N$ are dual decision variables, and $\boldsymbol{\omega}\circ\boldsymbol{\omega}$ denotes the Hadamard product of the vector $\boldsymbol{\omega}$.

Proof. See Appendix A.3. □

Remark 5.1. Note that the dual problem (5.8) is always feasible with $\boldsymbol{\alpha} = \boldsymbol{p}$, $\boldsymbol{\zeta} = \boldsymbol{0}$, $\lambda = 0$, and $\boldsymbol{\omega} = \sum_{s\in[S]} p_s \boldsymbol{r}^{(s)}$. From the strong duality, the dual problem (5.8) is unbounded (i.e., $f(\boldsymbol{z}) = \infty$) if and only if the corresponding primal problem (5.7) is infeasible. Our objective is to minimize $f(\boldsymbol{z})$, so we can assume without loss of generality that the primal problem (5.7) is feasible.

Remark 5.2. When $z_n = 0$, we can eliminate ω_n from the dual problem (5.8). Suppose that after the elimination, $(\boldsymbol{\alpha}^\star, \boldsymbol{\zeta}^\star, \lambda^\star)$ is an optimal solution of $(\boldsymbol{\alpha}, \boldsymbol{\zeta}, \lambda)$ to Problem (5.8). We then recover ω_n as

$$
\omega_n^\star = \left[\sum_{s\in[S]} \alpha_s^\star r_n^{(s)} - \boldsymbol{c}_n^\top\boldsymbol{\zeta}^\star + \lambda^\star\right]_+ \qquad (n \in [N] \text{ with } z_n = 0),
$$

where $\boldsymbol{c}_n$ is the nth column vector of $\boldsymbol{C}$. This solution satisfies Constraint (5.8b) and thus optimal because its coefficient is zero in the objective (5.8a).

Theorem 5.1 allows us to redefine $f(\boldsymbol{z})$ as the optimal objective value of Problem (5.8) for $\boldsymbol{z} \in [0,1]^N$. Then, as we showed Lemmas 4.3 and 4.4 in Chapter 4, we obtain two properties of $f(\boldsymbol{z})$.

Lemma 5.2 (Convexity)**.** *The function $f(\boldsymbol{z})$ is convex in $\boldsymbol{z} \in [0,1]^N$.*

Lemma 5.3 (Subgradient). *Suppose that Problem (5.7) is feasible with* $\boldsymbol{z} \in \{0,1\}^N$*, and that* $\boldsymbol{\omega}^\star(\boldsymbol{z})$ *is an optimal solution of* $\boldsymbol{\omega}$ *to Problem (5.8). Then, a subgradient of the function* $f(\boldsymbol{z})$ *is given by*

$$\boldsymbol{g}(\boldsymbol{z}) := -\frac{\gamma}{2}\boldsymbol{\omega}^\star(\boldsymbol{z}) \circ \boldsymbol{\omega}^\star(\boldsymbol{z}) \in \partial f(\boldsymbol{z}). \tag{5.9}$$

From Lemmas 5.2 and 5.3, we see that for each $\hat{\boldsymbol{z}} \in \mathcal{Z}_N^k$, Problem (5.8) with $\boldsymbol{z} = \hat{\boldsymbol{z}}$ yields a linear underestimator of $f(\boldsymbol{z})$ for $\boldsymbol{z} \in [0,1]^N$ as follows:

$$f(\boldsymbol{z}) \geq f(\hat{\boldsymbol{z}}) + \boldsymbol{g}(\hat{\boldsymbol{z}})^\top(\boldsymbol{z} - \hat{\boldsymbol{z}}). \tag{5.10}$$

5.3.2 Upper-level cutting-plane algorithm

Here, we extend the cutting-plane algorithm in Chapter 4 to the upper-level problem (5.5) for mean-CVaR portfolio optimization. We refer to this algorithm as the *upper-level cutting-plane algorithm*, which finds a sequence of solutions to relaxed versions of Problem (5.5).

We first define the initial feasible region as

$$\mathcal{F}_1 := \{(\boldsymbol{z}, \theta) \in \mathcal{Z}_N^k \times \mathbb{R} \mid \theta \geq \theta_{\mathrm{LB}}\}, \tag{5.11}$$

where θ is an auxiliary decision variable that serves as a lower bound on $f(\boldsymbol{z})$, and θ_{LB} is a lower bound of the optimal objective value of Problem (5.5). At the tth iteration ($t \geq 1$), our algorithm solves the following optimization problem:

$$\underset{\boldsymbol{z},\theta}{\text{minimize}} \;\; \theta \quad \text{subject to} \;\; (\boldsymbol{z}, \theta) \in \mathcal{F}_t, \tag{5.12}$$

where $\mathcal{F}_t$ is a relaxed feasible region at the tth iteration such that $\mathcal{F}_{t+1} \subseteq \mathcal{F}_t$. According to Eq. (5.11), the objective value of Problem (5.12) is bounded below. Therefore, unless $\mathcal{F}_t = \emptyset$, Problem (5.12) has an optimal solution, which is denoted by $(\boldsymbol{z}_t, \theta_t)$.

After obtaining $(\boldsymbol{z}_t, \theta_t)$, we solve the dual lower-level problem (5.8) with $\boldsymbol{z} = \boldsymbol{z}_t$. If Problem (5.8) is unbounded, Problem (5.7) is infeasible because of the strong duality. In this case, we update the feasible region to cut off the solution $\boldsymbol{z}_t$ as follows:

$$\mathcal{F}_{t+1} \leftarrow \mathcal{F}_t \cap \{(\boldsymbol{z}, \theta) \in \mathcal{Z}_N^k \times \mathbb{R} \mid \boldsymbol{z}_t^\top(\boldsymbol{1} - \boldsymbol{z}) + (\boldsymbol{1} - \boldsymbol{z}_t)^\top \boldsymbol{z} \geq 1\}. \tag{5.13}$$

If Problem (5.8) is bounded, we obtain the function value $f(\boldsymbol{z}_t)$ and its subgradient $\boldsymbol{g}(\boldsymbol{z}_t)$ as in Lemma 5.3.

If $f(\boldsymbol{z}_t) - \theta_t \leq \varepsilon$ with sufficiently small $\varepsilon \geq 0$, then $\boldsymbol{z}_t$ is an ε-optimal solution to Problem (5.5), which means that

$$f^\star \leq f(\boldsymbol{z}_t) \leq f^\star + \varepsilon, \tag{5.14}$$

where $f^\star$ is the optimal objective value of Problem (5.5). In this case, we terminate the algorithm with the ε-optimal solution $\boldsymbol{z}_t$. Otherwise, we add the constraint (5.10) to the feasible region:

$$\mathcal{F}_{t+1} \leftarrow \mathcal{F}_t \cap \{(\boldsymbol{z}, \theta) \in \mathcal{Z}_N^k \times \mathbb{R} \mid \theta \geq f(\boldsymbol{z}_t) + \boldsymbol{g}(\boldsymbol{z}_t)^\top(\boldsymbol{z} - \boldsymbol{z}_t)\}. \tag{5.15}$$

Note that this update cuts off the solution $(\boldsymbol{z}_t, \theta_t)$ because $\theta_t < f(\boldsymbol{z}_t)$.

After updating the feasible region, we set $t \leftarrow t + 1$ and solve Problem (5.12) again. This procedure is repeated until an ε-optimal solution $\hat{\boldsymbol{z}}$ is found. After termination of the algorithm, we can compute the corresponding portfolio by solving Problem (5.7) with $\boldsymbol{z} = \hat{\boldsymbol{z}}$. Our upper-level cutting-plane algorithm is summarized by Algorithm 5.1.

Algorithm 5.1 Upper-level cutting-plane algorithm for solving Problem (5.5)

Step 0 (Initialization) Let $\varepsilon \geq 0$ be a tolerance for optimality. Define the feasible region $\mathcal{F}_1$ as in Eq. (5.11). Set $t \leftarrow 1$ and $\mathrm{UB}_0 \leftarrow \infty$.

Step 1 (Relaxed Problem) Solve Problem (5.12). Let $(\boldsymbol{z}_t, \theta_t)$ be an optimal solution, and set $\mathrm{LB}_t \leftarrow \theta_t$.

Step 2 (Cut Generation) Solve Problem (5.8) with $\boldsymbol{z} = \boldsymbol{z}_t$ to calculate $f(\boldsymbol{z}_t)$ and $\boldsymbol{\omega}^\star(\boldsymbol{z}_t)$.

(a) If Problem (5.8) is unbounded, set $\mathrm{UB}_t \leftarrow \mathrm{UB}_{t-1}$ and update the feasible region as in Eq. (5.13).

(b) Otherwise, perform the following procedures:

i. Calculate $\boldsymbol{g}(\boldsymbol{z}_t)$ as in Eq. (5.9).

ii. Update the feasible region as in Eq. (5.15).

iii. If $f(\boldsymbol{z}_t) < \mathrm{UB}_{t-1}$, set $\mathrm{UB}_t \leftarrow f(\boldsymbol{z}_t)$ and $(\hat{\boldsymbol{z}}, \hat{\theta}) \leftarrow (\boldsymbol{z}_t, \theta_t)$; otherwise, set $\mathrm{UB}_t \leftarrow \mathrm{UB}_{t-1}$.

Step 3 (Termination Condition) If $\mathrm{UB}_t - \mathrm{LB}_t \leq \varepsilon$, terminate the algorithm with the ε-optimal solution $\hat{\boldsymbol{z}}$.

Step 4 Set $t \leftarrow t + 1$ and return to Step 1.

5.3.3 Lower-level cutting-plane algorithm

Note that Step 2 of Algorithm 5.1 solves Problem (5.8), which contains $\mathcal{O}(S)$ decision variables and $\mathcal{O}(S)$ constraints; so, solving it directly is computationally expensive especially when the number of scenarios is very large. To avoid this difficulty, we propose solving the primal lower-level problem (5.6) by instead using the other cutting-plane algorithm [4, 80, 103, 149]. Hereinafter, we refer to this algorithm as the *lower-level cutting-plane algorithm* to distinguish it from the upper-level cutting-plane algorithm (Algorithm 5.1). We will explain in Section 5.3.4 how to derive dual solutions for cut generation from Problem (5.6).

We first use the cutting-plane representation [52, 103] to reformulate Problem (5.6) as follows:

$$
\begin{aligned}
f(\boldsymbol{z}) = \underset{a,v,\boldsymbol{x}}{\text{minimize}} \quad & \frac{1}{2\gamma}\boldsymbol{x}^\top \boldsymbol{x} + a + v && \text{(5.16a)}\\
\text{subject to} \quad & v \geq \frac{1}{1-\beta}\sum_{s\in\mathcal{J}} p_s(-(\boldsymbol{r}^{(s)})^\top \boldsymbol{Z}\boldsymbol{x} - a) \quad (\forall \mathcal{J} \subseteq [S]), && \text{(5.16b)}\\
& \boldsymbol{Z}\boldsymbol{x} \in \mathcal{X}, && \text{(5.16c)}
\end{aligned}
$$

where $\mathcal{J}$ denotes a subset of the scenario set $[S]$. We pick out the constraint (5.16b) with $\mathcal{J} = [S]$ to define the initial relaxed feasible region:

$$
\mathcal{G}_1(\boldsymbol{z}) := \left\{ (a, v, \boldsymbol{x}) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^N \;\middle|\; \begin{array}{l} v \geq \dfrac{1}{1-\beta}\displaystyle\sum_{s\in[S]} p_s(-(\boldsymbol{r}^{(s)})^\top \boldsymbol{Z}\boldsymbol{x} - a), \\ v \geq 0,\ \boldsymbol{Z}\boldsymbol{x} \in \mathcal{X} \end{array} \right\}. \tag{5.17}
$$

It is clear from Eq. (5.4b) that $v \geq 0$ is a valid constraint. At the tth iteration ($t \geq 1$), our algorithm solves the following optimization problem:

$$\underset{a,v,\boldsymbol{x}}{\text{minimize}} \quad \frac{1}{2\gamma}\boldsymbol{x}^\top \boldsymbol{x} + a + v \tag{5.18a}$$

$$\text{subject to} \quad (a, v, \boldsymbol{x}) \in \mathcal{G}_t(\boldsymbol{z}), \tag{5.18b}$$

where $\mathcal{G}_t(\boldsymbol{z})$ is a relaxed feasible region at the tth iteration such that $\mathcal{G}_t(\boldsymbol{z}) \subseteq \mathcal{G}_1(\boldsymbol{z})$. The objective value of Problem (5.18) is bounded below by the initial feasible region (5.17). Therefore, unless $\mathcal{G}_t(\boldsymbol{z}) = \emptyset$, Problem (5.18) has an optimal solution, which is denoted by $(a_t, v_t, \boldsymbol{x}_t)$. Note that Problem (5.18) contains only $N + 2$ decision variables, which are independent of the number of scenarios.

After obtaining $(a_t, v_t, \boldsymbol{x}_t)$, we define a subset of scenarios

$$\mathcal{J}_t \leftarrow \{s \in [S] \mid -(\boldsymbol{r}^{(s)})^\top \boldsymbol{Z}\boldsymbol{x}_t - a_t > 0\} \tag{5.19}$$

and calculate

$$v'_t \leftarrow \frac{1}{1-\beta} \sum_{s \in \mathcal{J}_t} p_s(-(\boldsymbol{r}^{(s)})^\top \boldsymbol{Z}\boldsymbol{x}_t - a_t). \tag{5.20}$$

Because the revised solution $(a_t, v'_t, \boldsymbol{x}_t)$ satisfies all the constraints of Problem (5.16), this solution gives an upper bound of $f(\boldsymbol{z})$.

If $v'_t - v_t \leq \delta$ with sufficiently small $\delta \geq 0$, we terminate the algorithm with the δ-optimal solution $(a_t, v'_t, \boldsymbol{x}_t)$ to Problem (5.16). Otherwise, we update the feasible region as follows:

$$\mathcal{G}_{t+1}(\boldsymbol{z}) \leftarrow \mathcal{G}_t(\boldsymbol{z}) \cap \left\{ (a, v, \boldsymbol{x}) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^N \;\middle|\; v \geq \frac{1}{1-\beta} \sum_{s \in \mathcal{J}_t} p_s(-(\boldsymbol{r}^{(s)})^\top \boldsymbol{Z}\boldsymbol{x} - a) \right\}. \tag{5.21}$$

Because $v_t < v'_t$, this update cuts off the solution $(a_t, v_t, \boldsymbol{x}_t)$.

Next, we set $t \leftarrow t + 1$ and solve Problem (5.18) again. The lower-level cutting-plane algorithm is summarized by Algorithm 5.2. As shown in previous studies [4, 80, 103, 149], Algorithm 5.2 terminates in a finite number of iterations and outputs a δ-optimal solution $(\hat{a}, \hat{v}, \hat{\boldsymbol{x}})$ to the lower-level problem (5.6). This algorithm is empirically much faster than solving Problem (5.7) based on the lifting representation when the number of scenarios is very large.

Suppose that Algorithm 5.2 terminates at the Tth iteration and outputs a δ-optimal solution $(\hat{a}, \hat{v}, \hat{\boldsymbol{x}})$, where $(a_T, v_T, \boldsymbol{x}_T)$ is an optimal solution to Problem (5.18) with $t = T$. The resultant family of scenario subsets $\mathcal{K}_\delta(\boldsymbol{z}) = \{[S], \mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_T\}$ is also provided by Algorithm 5.2. We then obtain lower and upper bounds on $f(\boldsymbol{z})$ as follows:

$$f(\boldsymbol{z}) - \delta \leq f_\delta(\boldsymbol{z}) \leq f(\boldsymbol{z}) \leq \hat{f}_\delta(\boldsymbol{z}) \leq f(\boldsymbol{z}) + \delta, \tag{5.22}$$

where

$$f_\delta(\boldsymbol{z}) := \frac{1}{2\gamma}\boldsymbol{x}_T^\top \boldsymbol{x}_T + a_T + v_T, \quad \hat{f}_\delta(\boldsymbol{z}) := \frac{1}{2\gamma}\hat{\boldsymbol{x}}^\top \hat{\boldsymbol{x}} + \hat{a} + \hat{v}.$$

5.3.4 Efficient cut generation for the upper-level problem

Now, we turn to the computation of dual solutions from primal solutions provided by the lower-level cutting-plane algorithm (Algorithm 5.2). Such dual solutions

Algorithm 5.2 Lower-level cutting-plane algorithm for solving Problem (5.16)

Step 0 (Initialization) Let $\delta \geq 0$ be a tolerance for optimality. Define the feasible region $\mathcal{G}_1(\boldsymbol{z})$ as in Eq. (5.17). Set $t \leftarrow 1$.

Step 1 (Relaxed Problem) Solve Problem (5.18). Let $(a_t, v_t, \boldsymbol{x}_t)$ be an optimal solution.

Step 2 (Upper Bound) Calculate $\mathcal{J}_t$ and v'_t as in Eqs. (5.19) and (5.20).

Step 3 (Termination Condition) If $v'_t - v_t \leq \delta$, terminate the algorithm with the δ-optimal solution $(\hat{a}, \hat{v}, \hat{\boldsymbol{x}}) := (a_t, v'_t, \boldsymbol{x}_t)$ and the resultant family of scenario subsets $\mathcal{K}_\delta(\boldsymbol{z}) := \{[S], \mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_t\}$.

Step 4 (Cut Generation) Update the feasible region as in Eq. (5.21).

Step 5 Set $t \leftarrow t + 1$ and return to Step 1.

are required for cut generation in the upper-level cutting-plane algorithm (Algorithm 5.1).

Suppose that Algorithm 5.2 results in the following relaxed problem with $\mathcal{K} = \mathcal{K}_\delta(\boldsymbol{z})$:

$$
\begin{aligned}
f_{\mathcal{K}}(\boldsymbol{z}) := \underset{a,v,\boldsymbol{x}}{\text{minimize}} \quad & \frac{1}{2\gamma}\boldsymbol{x}^\top \boldsymbol{x} + a + v && \text{(5.23a)}\\
\text{subject to} \quad & v \geq \frac{1}{1-\beta}\sum_{s\in\mathcal{J}} p_s(-(\boldsymbol{r}^{(s)})^\top \boldsymbol{Z}\boldsymbol{x} - a) \quad (\forall \mathcal{J} \in \mathcal{K}), && \text{(5.23b)}\\
& v \geq 0, \quad \boldsymbol{Z}\boldsymbol{x} \in \mathcal{X}. && \text{(5.23c)}
\end{aligned}
$$

The following theorem derives the dual formulation of Problem (5.23).

Theorem 5.4. *Suppose that Problem (5.23) is feasible with $\boldsymbol{z} \in \{0,1\}^N$. Then, the strong duality holds, and the dual formulation of Problem (5.23) can be written as*

$$
\begin{aligned}
f_{\mathcal{K}}(\boldsymbol{z}) = \underset{\boldsymbol{\alpha},\boldsymbol{\zeta},\lambda,\boldsymbol{\omega}}{\text{maximize}} \quad & -\frac{\gamma}{2}\boldsymbol{z}^\top(\boldsymbol{\omega}\circ\boldsymbol{\omega}) - \boldsymbol{d}^\top\boldsymbol{\zeta} + \lambda && \text{(5.24a)}\\
\text{subject to} \quad & \boldsymbol{\omega} \geq \sum_{\mathcal{J}\in\mathcal{K}} \alpha_{\mathcal{J}} \sum_{s\in\mathcal{J}} p_s \boldsymbol{r}^{(s)} - \boldsymbol{C}^\top\boldsymbol{\zeta} + \lambda\mathbf{1}, && \text{(5.24b)}\\
& \sum_{\mathcal{J}\in\mathcal{K}} \alpha_{\mathcal{J}} \leq 1, && \text{(5.24c)}\\
& \sum_{\mathcal{J}\in\mathcal{K}} \alpha_{\mathcal{J}} \sum_{s\in\mathcal{J}} p_s = 1-\beta, && \text{(5.24d)}\\
& \boldsymbol{\alpha} \geq \mathbf{0}, \quad \boldsymbol{\zeta} \geq \mathbf{0}, && \text{(5.24e)}
\end{aligned}
$$

where $\boldsymbol{\alpha} := (\alpha_{\mathcal{J}})_{\mathcal{J}\in\mathcal{K}}$, $\boldsymbol{\zeta} \in \mathbb{R}^M$, $\lambda \in \mathbb{R}$, and $\boldsymbol{\omega} \in \mathbb{R}^N$ are dual decision variables.

Proof. See Appendix A.4. □

Note that $|\mathcal{K}_\delta(\boldsymbol{z})|$ is usually very small even when there are many scenarios. Therefore, Problem (5.24) can be solved efficiently regardless of the number of scenarios.

From Theorem 5.4, we can redefine $f_{\mathcal{K}}(\boldsymbol{z})$ to be the optimal objective value of Problem (5.24) for $\boldsymbol{z} \in [0,1]^N$ as we did for $f(\boldsymbol{z})$. Let $\boldsymbol{\omega}^\star_\delta(\boldsymbol{z})$ be the optimal

solution of $\boldsymbol{\omega}$ to Problem (5.24) with $\mathcal{K} = \mathcal{K}_\delta(\boldsymbol{z})$. We then define

$$\boldsymbol{g}_\delta(\boldsymbol{z}) := -\frac{\gamma}{2}\boldsymbol{\omega}_\delta^\star(\boldsymbol{z}) \circ \boldsymbol{\omega}_\delta^\star(\boldsymbol{z}). \tag{5.25}$$

Note that when $\delta > 0$, $f_\delta(\boldsymbol{z})$ and $\boldsymbol{g}_\delta(\boldsymbol{z})$ are not exactly the same as $f(\boldsymbol{z})$ and $\boldsymbol{g}(\boldsymbol{z})$, respectively. However, the following theorem verifies that $f_\delta(\boldsymbol{z})$ and $\boldsymbol{g}_\delta(\boldsymbol{z})$ still provide a linear underestimator of $f(\boldsymbol{z})$.

Theorem 5.5. *Suppose that $\hat{\boldsymbol{z}} \in \{0,1\}^N$ and Problem (5.23) is feasible with $(\boldsymbol{z}, \mathcal{K}) = (\hat{\boldsymbol{z}}, \mathcal{K}_\delta(\hat{\boldsymbol{z}}))$. Then, it holds that for all $\boldsymbol{z} \in [0,1]^N$,*

$$f(\boldsymbol{z}) \geq f_\delta(\hat{\boldsymbol{z}}) + \boldsymbol{g}_\delta(\hat{\boldsymbol{z}})^\top(\boldsymbol{z} - \hat{\boldsymbol{z}}). \tag{5.26}$$

Proof. Because $\mathcal{K} = \mathcal{K}_\delta(\hat{\boldsymbol{z}})$, it follows from Lemma 5.3 that $\boldsymbol{g}_\delta(\hat{\boldsymbol{z}})$ is a subgradient of $f_\mathcal{K}(\boldsymbol{z})$ at $\boldsymbol{z} = \hat{\boldsymbol{z}}$. It then holds that for all $\boldsymbol{z} \in [0,1]^N$,

$$f(\boldsymbol{z}) \geq f_\mathcal{K}(\boldsymbol{z}) \geq f_\mathcal{K}(\hat{\boldsymbol{z}}) + \boldsymbol{g}_\delta(\hat{\boldsymbol{z}})^\top(\boldsymbol{z} - \hat{\boldsymbol{z}}). \tag{5.27}$$

This proof is completed because $f_\delta(\hat{\boldsymbol{z}}) = f_\mathcal{K}(\hat{\boldsymbol{z}})$ with $\mathcal{K} = \mathcal{K}_\delta(\hat{\boldsymbol{z}})$. □

5.3.5 Bilevel cutting-plane algorithm

We are now ready to describe our bilevel cutting-plane algorithm for solving the upper-level problem (5.5). We use the lower-level cutting-plane algorithm (Algorithm 5.2) to accelerate the cut generation at Step 2 of Algorithm 5.1.

Suppose that $(\boldsymbol{z}_t, \theta_t)$ is an optimal solution to the relaxed problem (5.12) at the tth iteration ($t \geq 1$). We then solve the primal lower-level problem (5.16) with $\boldsymbol{z} = \boldsymbol{z}_t$ by means of the lower-level cutting-plane algorithm (Algorithm 5.2), which provides $f_\delta(\boldsymbol{z}_t)$, $\hat{f}_\delta(\boldsymbol{z}_t)$, and $\mathcal{K}_\delta(\boldsymbol{z}_t)$. We next solve the reduced version (5.24) of the dual lower-level problem with $(\boldsymbol{z}, \mathcal{K}) = (\boldsymbol{z}_t, \mathcal{K}_\delta(\boldsymbol{z}_t))$, thereby yielding the optimal solution $\boldsymbol{\omega}_\delta^\star(\boldsymbol{z}_t)$. After that, we calculate $\boldsymbol{g}_\delta(\boldsymbol{z}_t)$ as in Eq. (5.25) and update the feasible region as follows:

$$\mathcal{F}_{t+1} \leftarrow \mathcal{F}_t \cap \{(\boldsymbol{z}, \theta) \in \mathcal{Z}_N^k \times \mathbb{R} \mid \theta \geq f_\delta(\boldsymbol{z}_t) + \boldsymbol{g}_\delta(\boldsymbol{z}_t)^\top(\boldsymbol{z} - \boldsymbol{z}_t)\}. \tag{5.28}$$

Our bilevel cutting-plane algorithm is summarized by Algorithm 5.3. Note that at Step 2 (Algorithm 5.3), the portfolio $\hat{\boldsymbol{x}}$ is generated by Algorithm 5.2 at each iteration.

Remark 5.3. Several algorithms (e.g., primal-dual interior-point methods [166] and alternating direction methods of multipliers [35]) solve primal and dual problems simultaneously. By applying such algorithms in Step 1 of Algorithm 5.2, we can skip solving the dual lower-level problem (5.24) at Step 2 of Algorithm 5.3.

Remark 5.4. Note that Step 1 of Algorithm 5.3 solves the MIO problem (5.12) at each iteration; this amounts to a "multi-tree" implementation, where a branch-and-bound algorithm is repeatedly executed from scratch. To improve computational efficiency, we can use the function of `lazy constraint callback`, which is offered by modern optimization software (e.g., CPLEX or Gurobi). This function enables a "single-tree" implementation [132]; that is, cutting planes (5.28) are dynamically generated during the process of the branch-and-bound algorithm.

To prove the convergence properties of Algorithm 5.3, we show the following lemma.

Algorithm 5.3 Bilevel cutting-plane algorithm for solving Problem (5.5)

Step 0 (Initialization) Let $\varepsilon \geq 0$ and $\delta \geq 0$ be tolerances for optimality. Define the feasible region $\mathcal{F}_1$ as in Eq. (5.11). Set $t \leftarrow 1$ and $\mathrm{UB}_0 \leftarrow \infty$.

Step 1 (Relaxed Problem) Solve Problem (5.12). Let $(\boldsymbol{z}_t, \theta_t)$ be an optimal solution, and set $\mathrm{LB}_t \leftarrow \theta_t$.

Step 2 (Cut Generation) Solve Problem (5.16) with $\boldsymbol{z} = \boldsymbol{z}_t$ by means of Algorithm 5.2 to calculate $f_\delta(\boldsymbol{z}_t)$, $\hat{f}_\delta(\boldsymbol{z}_t)$, and $\mathcal{K}_\delta(\boldsymbol{z}_t)$.

(a) If Problem (5.16) is infeasible, set $\mathrm{UB}_t \leftarrow \mathrm{UB}_{t-1}$ and update the feasible region as in Eq. (5.13).

(b) If Problem (5.16) is feasible, perform the following procedures:

i. Solve Problem (5.24) with $(\boldsymbol{z}, \mathcal{K}) = (\boldsymbol{z}_t, \mathcal{K}_\delta(\boldsymbol{z}_t))$ to calculate $\boldsymbol{\omega}^\star_\delta(\boldsymbol{z}_t)$.

ii. Calculate $\boldsymbol{g}_\delta(\boldsymbol{z}_t)$ as in Eq. (5.25).

iii. Update the feasible region as in Eq. (5.28).

iv. If $\hat{f}_\delta(\boldsymbol{z}_t) < \mathrm{UB}_{t-1}$, set $\mathrm{UB}_t \leftarrow \hat{f}_\delta(\boldsymbol{z}_t)$ and $(\hat{\boldsymbol{z}}, \hat{\theta}) \leftarrow (\boldsymbol{z}_t, \theta_t)$; otherwise, set $\mathrm{UB}_t \leftarrow \mathrm{UB}_{t-1}$.

Step 3 (Termination Condition) If $\boldsymbol{z}_u = \boldsymbol{z}_t$ for some $u < t$ or $\mathrm{UB}_t - \mathrm{LB}_t \leq \varepsilon$, then terminate the algorithm with the $\max\{\delta, \varepsilon\}$-optimal solution $\hat{\boldsymbol{z}}$.

Step 4 Set $t \leftarrow t + 1$ and return to Step 1.

Lemma 5.6. *Suppose that* $\{(\boldsymbol{z}_t, \theta_t) \mid t = 1, 2, \ldots, T\}$ *is a sequence of solutions generated by Algorithm 5.3. If there exists* $u < T$ *such that* $\boldsymbol{z}_u = \boldsymbol{z}_T$*, then* $\boldsymbol{z}_T$ *is a* δ*-optimal solution to Problem (5.5), meaning that*

$$f^\star \leq f(\boldsymbol{z}_T) \leq f^\star + \delta.$$

Proof. Note that $(\boldsymbol{z}_T, \theta_T)$ is contained in the feasible region (5.28) at $t = u$. Because $\boldsymbol{z}_u = \boldsymbol{z}_T$, it follows that

$$\theta_T \geq f_\delta(\boldsymbol{z}_u) + \boldsymbol{g}_\delta(\boldsymbol{z}_u)^\top (\boldsymbol{z}_T - \boldsymbol{z}_u) = f_\delta(\boldsymbol{z}_T).$$

From Eqs. (5.14) and (5.22), we have

$$f^\star \leq f(\boldsymbol{z}_T), \quad f(\boldsymbol{z}_T) - \delta \leq f_\delta(\boldsymbol{z}_T).$$

Because $\theta_T \leq f^\star$, it follows that

$$f^\star \leq f(\boldsymbol{z}_T) \leq f_\delta(\boldsymbol{z}_T) + \delta \leq \theta_T + \delta \leq f^\star + \delta,$$

which completes the proof. □

From Lemma 5.6, we can establish the convergence properties of Algorithm 5.3.

Theorem 5.7. *Algorithm 5.3 terminates in a finite number of iterations and outputs a* $\max\{\delta, \varepsilon\}$*-optimal solution.*

Proof. Because there are at most a finite number of solutions $\boldsymbol{z} \in \mathcal{Z}_N^k$, Algorithm 5.3 terminates in a finite number of iterations with $\boldsymbol{z}_u = \boldsymbol{z}_t$ for some $u < t$. In this case, a δ-optimal solution is found according to Lemma 5.6. If $\mathrm{UB}_t - \mathrm{LB}_t \leq \varepsilon$ is fulfilled first, Algorithm 5.3 terminates with an ε-optimal solution. □

5.4 Numerical experiments

In this section, we report numerical results to evaluate the efficiency of our algorithms for solving cardinality-constrained mean-CVaR portfolio optimization problems.

5.4.1 Problem instances

Table 5.1 lists the datasets used in our experiments, where N is the number of assets. From the data library on the website of Kenneth R. French [59], we downloaded two historical datasets (i.e., `ind49` and `sbm100`) of US stock returns. We used monthly data from January 2010 to December 2019 to compute the mean vector $\hat{\boldsymbol{\mu}}$ and covariance matrix $\hat{\boldsymbol{\Sigma}}$ of asset returns. From Kaggle datasets [90], we downloaded a dataset (i.e., `sp200`) of historical stock returns, where top 200 companies were selected from the S&P 500 index according to market capitalization as of December 2020. To compute $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$, we used daily data from February 8th, 2013 to February 7th, 2018. From the OR-Library [16, 41], we downloaded three datasets (i.e., `port1`, `port2`, and `port5`) of $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$, which were multiplied by 100 to be consistent with the other datasets.

Table 5.1: Dataset description

Abbr.	N	Original dataset
`ind49`	49	49 Industry Portfolios [59]
`sbm100`	100	100 Portfolios Formed on Size and Book-to-Market [59]
`sp200`	200	Top 200 companies in the S&P 500 index [90]
`port1`	31	port1 (Portfolio optimization: Single period) [16, 41]
`port2`	89	port2 (Portfolio Foptimization: Single period) [16, 41]
`port5`	225	port5 (Portfolio optimization: Single period) [16, 41]

For each dataset, we randomly generated S scenarios of asset returns $\{\boldsymbol{r}^{(s)} \mid s \in [S]\}$ from $\mathrm{N}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$, a normal distribution with the parameters $(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$. We set the occurrence probability as $p_s = 1/S$ for all $s \in [S]$. The required return level in Eq. (5.2) was set as $\bar{\mu} = 0.3\mu_{\min} + 0.7\mu_{\max}$, where $\mu_{\min}$ and $\mu_{\max}$ were the average returns of the bottom- and top-k assets, respectively; this setting ensures feasibility of Problem (5.4). The probability level of CVaR was set as $\beta = 0.9$.

5.4.2 Methods for comparison

A standard MIO formulation of Problem (5.4) uses the big-M method, which replaces the logical implication (5.4c) with

$$0 \leq x_n \leq z_n \quad (\forall n \in [N]), \tag{5.29}$$

which is valid because of the constraint $\boldsymbol{x} \in \mathcal{X}$ in Problem (5.4).

Another state-of-the-art MIO formulation uses the perspective reformulation [74, 75]:

$$\underset{a,v,\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}}{\text{minimize}} \quad \frac{1}{2\gamma}\sum_{n\in[N]} y_n + a + v \tag{5.30a}$$

$$\text{subject to} \quad v \geq \frac{1}{1-\beta}\sum_{s\in[S]} p_s \left[-(\boldsymbol{r}^{(s)})^\top \boldsymbol{x} - a\right]_+, \tag{5.30b}$$

$$x_n^2 \leq y_n z_n, \quad y_n \geq 0 \quad (\forall n \in [N]), \tag{5.30c}$$

$$\boldsymbol{x} \in \mathcal{X}, \quad \boldsymbol{z} \in \mathcal{Z}_N^k, \tag{5.30d}$$

where $\boldsymbol{y} := (y_1, y_2, \ldots, y_N)^\top$ is a vector of auxiliary decision variables. In recent years, the perspective reformulation has been actively studied [56, 57, 79, 173].

We compare the computational performances of the following methods:

BigM solves Problem (5.4) after replacing Eq. (5.4c) with Eq. (5.29);

Persp solves Problem (5.30) as a mixed-integer second-order cone optimization problem;

- **Lift** replaces Eqs. (5.4b) and (5.30b) with the lifting representation (5.7b)–(5.7d);
- **Cut** applies the cutting-plane algorithm [4, 80, 103, 149] to Problems (5.4) and (5.30);

CP solves Problem (5.5) by means of the upper-level cutting-plane algorithm (Algorithm 5.1);

BCP solves Problem (5.5) by means of the bilevel cutting-plane algorithm (Algorithm 5.3);

BCPc solves Problem (5.5) by means of the bilevel cutting-plane algorithm (Algorithm 5.3), where the callback function is employed in implementation.

All experiments were performed on a Windows 10 PC with an Intel Core i7-4790 CPU (3.6.0GHz) and 16 GB of memory. All methods were implemented in Python 3.7 with Gurobi Optimizer 8.1.1[1]. We employed the primal-dual interior-point method offered by Gurobi at Step 1 of Algorithm 5.2. We also tested the performance of an alternating direction method of multipliers, but it made Algorithm 5.2 slower because of numerical inaccuracies.

We set $\varepsilon = \delta = 10^{-5}$ for tolerances for optimality. Except for BCP, we used `lazy constraint callback` for adding cutting planes during the branch-and-bound procedure.

The computation of each method was terminated if it did not finish by itself within 3600 s. In these cases, the results obtained within 3600 s were taken as the final outcome.

5.4.3 Evaluation metrics

The row labels used in the tables of experimental results are defined as follows:

Solved number of problem instances solved to optimality within 3600 s;

[1]The source code is available at https://github.com/KenKoba2119/cardinality-constrained_cvar_optimization.

Time computation time in seconds;

Obj objective value of the obtained best feasible solution;

Gap(%) absolute difference between lower and upper bounds on the optimal objective value divided by the upper bound;

#Nodes number of nodes explored in the branch-and-bound algorithm;

#Cuts number of upper-level cutting planes generated.

Note that the best values of Time are indicated in bold for each instance, and those of Obj and Gap(%) are also indicated in bold for only the `sp200`, `port2`, and `port5` datasets.

To aggregate results of "Time," "#Nodes," and "#Cuts" in Tables 5.2 and 5.5 and Figure 5.1, we use the shifted geometric mean, which is used in Chapter 2. Recall the shifted geometric mean of values $x_1, x_2, \ldots, x_N$ is defined as

$$\left(\prod_{n \in [N]} (x_n + s) \right)^{1/N} - s,$$

where s is the shift parameter. Following Achterberg [1] and Mittlemann [77], we set $s = 10$ for "Time." We respectively set s to 100 and 10 for "#Nodes" and "#Cuts" in proportion to their scale. We use the arithmetic mean for "Gap(%)."

5.4.4 Results for various numbers of scenarios

We evaluate the computational performance of each method with the number of scenarios $S \in \{10^3, 10^4, 10^5\}$. Here, we set the parameters $k = 10$ for the cardinality constraint and $\gamma = 10/\sqrt{N}$ for the ℓ_2-regularization term. Table 5.2 summarizes numerical results for the six datasets (Table 5.1), and Tables 5.3 and 5.4 give the numerical results for each dataset.

First, we focus on the results of our cutting-plane algorithms (i.e., CP, BCP, and BCPc). BCP and BCPc were faster than CP for $S \geq 10^4$ (Table 5.2). In the case of the `ind49` dataset (Table 5.3), for example, although CP was the fastest among the three methods for $S = 10^3$, BCP and BCPc were much faster than CP for $S \geq 10^4$. These results suggest the effectiveness of the lower-level cutting-plane algorithm, which is used by BCP and BCPc to solve the lower-level problem efficiently regardless of the number of scenarios.

The differences in computation time between BCP and BCPs were relatively small (Table 5.2). For the `port2` dataset (Table 5.4), BCP and BCPc failed to complete the computations within 3600 s even when $S = 10^3$; however, BCPc found solutions of better quality than did BCP. BCP provides at most one feasible solution at every iteration t, whereas BCPc explores feasible solutions more frequently through the callback procedure. For this reason, BCPc is capable of yielding solutions of good quality even if the computation is terminated due to the time limit.

Next, we compare our bilevel cutting-plane algorithms (i.e., BCP and BCPc) with the MIO formulations (i.e., BigM and Persp). Persp+Lift was the fastest among the six methods for $S \leq 10^4$, whereas BCP and BCPc were faster than the other methods for $S = 10^5$ (Table 5.2). Indeed, for the `sbm100` dataset with $S = 10^5$ (Table 5.3), BCP was much faster than the four methods related to BigM and Persp. Moreover, for the `port5` dataset with $S = 10^5$ (Table 5.4), only

BCP and BCPc finished solving the problem within 3600 s, which is a remarkable result.

When $S = 10^5$, Persp+Lift finished solving only one problem instance, and its computation time was much longer than those of BCP and BCPc (Table 5.2). A main reason for this is that Persp+Lift solves at each node a second-order cone optimization problem whose problem size depends on S. In the case of the `port5` dataset with $S = 10^3$ (Table 5.4), Persp+Cut returned an incorrect optimal objective value because of numerical instability.

According to the convergence properties of the sample average approximation, a huge number of scenarios are required for calculating CVaR accurately [150]. Even with only 15 investable assets, it was necessary to have at least 5000 scenarios to ensure the stability of the CVaR optimization model [91]. These facts support the practicality of our bilevel cutting-plane algorithm, which has the potential to deal with many scenarios as shown in Tables 5.2–5.4.

5.4.5 Sensitivity to hyperparameter values

We examine the sensitivity of the computational performance to the hyperparameters k for the cardinality constraint and γ for the ℓ_2-regularization term. Here, we focus on the four datasets, namely, `ind49`, `sbm100`, `port1`, and `port5`.

Sensitivity to the cardinality parameter k

Table 5.5 summarizes numerical results for the four datasets with the cardinality parameter $k \in \{5, 10, 15\}$. Here, we set $\gamma = 10/\sqrt{N}$ for the ℓ_2-regularization term and $S = 10^5$ as the number of scenarios. The numerical results for each dataset are given in Appendix B (Table B.1).

Table 5.5 shows that when $k = 5$, although only Persp+Cut finished solving all the problem instances, the relative gaps of BCP and BCPc were very small (i.e., 0.36 and 1.68), which means that these methods found solutions of good quality. When $k \in \{10, 15\}$, both BCP and BCPc were faster than the other methods and solved all the problem instances to optimality within 3600 s.

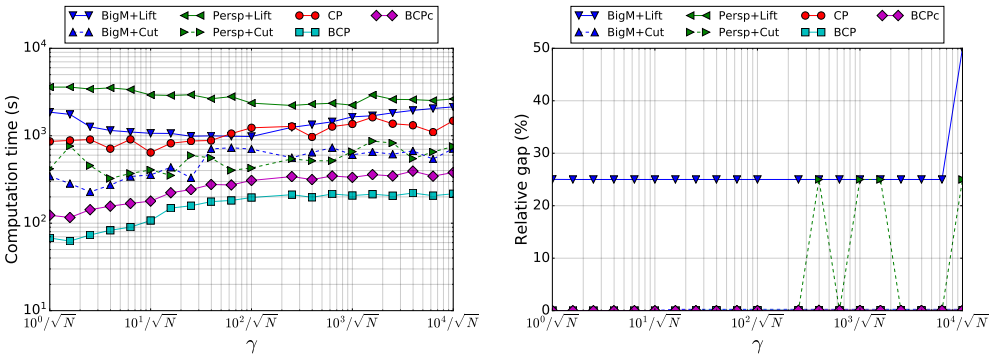
BCP and BCPc tended to be faster with larger k. Here, #Cuts was also much smaller for $k = 15$ than for $k = 5$, which is part of the reason why BCP and BCPc performed better with larger k.

Sensitivity to the regularization parameter γ

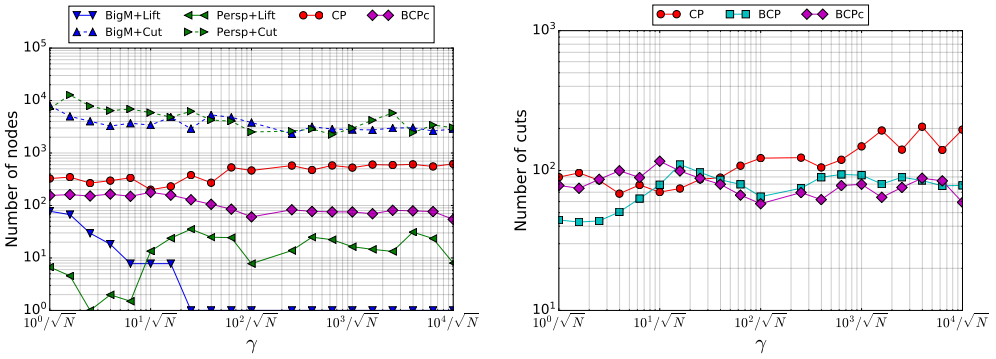
Figure 5.1 summarizes results of Time, GAP(%), #Nodes, and #Cuts for the four datasets with the regularization parameter $\gamma \in \{10^{0.2i}/\sqrt{N} \mid i = 0, 1, \ldots, 20\}$. Here, we set $k = 10$ for the cardinality constraint and $S = 10^5$ as the number of scenarios. The numerical results for each dataset are shown in Appendix B (Figures B.1–B.4).

We can see from Figure 5.1 that the performances of BCP and BCPc were not greatly affected by γ, and their computation times were consistently shorter than the other methods for every γ. Additionally, BCP and BCPc tended to be faster with smaller γ.

We next analyze the impact of the ℓ_2-regularization term on the objective value. Suppose that $(a^\star(\gamma), v^\star(\gamma), \boldsymbol{x}^\star(\gamma), \boldsymbol{z}^\star(\gamma))$ is an optimal solution computed by the bilelevel cutting-plane algorithm to Problem (5.4). Figure 5.2 shows



(a) Time (b) Gap(%)



(c) #Nodes (d) #Cuts

Figure 5.1: Summary of numerical results for the three datasets (`port1`, `ind49`, and `sbm100`) with $(S, k) = (10^5, 10)$ for $\gamma \in \{10^{0.2i}/\sqrt{N} \mid i = 0, 1, \ldots, 20\}$

"Reg + CVaR" and "CVaR" defined as

$$\underbrace{\frac{1}{2\gamma}\boldsymbol{x}^\star(\gamma)^\top \boldsymbol{x}^\star(\gamma)}_{\text{Reg}} + \underbrace{a^\star(\gamma) + v^\star(\gamma)}_{\text{CVaR}} \tag{5.31}$$

with the regularization parameter $\gamma \in \{10^{0.2i}/\sqrt{N} \mid i = 0, 1, \ldots, 20\}$. These values were averaged over the three datasets (`port1`, `ind49`, `sbm100`), where `port5` was omitted because most of the methods failed to find an optimal solution due to the time limit. The numerical results for each dataset are shown in B (Figure B.5).

We find from Figure 5.2 that the objective value (i.e., Reg + CVaR) was close to CVaR when $\gamma \geq 10^2/\sqrt{N}$. In addition, CVaR almost converged to its minimal value when $\gamma \geq 10/\sqrt{N}$. This result suggests that we can obtain a sufficiently good solution minimizing CVaR with $\gamma \geq 10/\sqrt{N}$.

5.5 Conclusion

In this chapter, we dealt with mean-CVaR portfolio optimization problems based on the sample average approximation with the cardinality constraint. It is very hard to exactly solve the problem when it involves a large number of investable assets. In addition, accurate approximation of CVaR based on the sample average approximation requires sufficiently many scenarios, which decreases computational efficiency. To overcome these challenges, we proposed a specialized

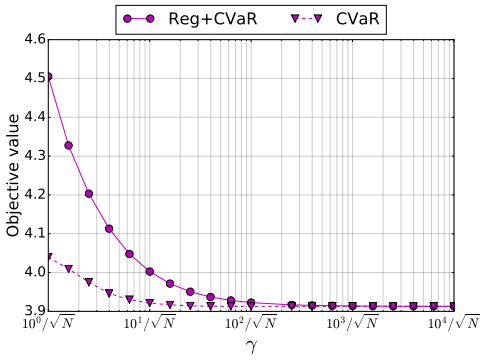


Figure 5.2: Summary of the objective value (5.31) for the three datasets (`port1`, `ind49`, and `sbm100`) with $(S, k) = (10^5, 10)$ for $\gamma \in \{10^{0.2i}/\sqrt{N} \mid i = 0, 1, \ldots, 20\}$

cutting-plane algorithm named the *bilevel cutting-plane algorithm* for solving the cardinality-constrained mean-CVaR portfolio optimization problem. We extended the cutting-plane algorithm discussed in Chapter 4 so that it incorporates another cutting-plane algorithm that efficiently minimizes CVaR. We also proved that our algorithms give a solution with guaranteed global optimality in a finite number of iterations.

The computational results indicate that our cutting-plane algorithms were very effective especially when the number of scenarios was large. Remarkably, our bilevel cutting-plane algorithm attained an optimal solution within 3600 s to a problem involving 225 assets and 100,000 scenarios. Furthermore, our algorithms performed well for most of the hyperparameter values.

Table 5.2: Summary of numerical results for the six datasets (Table 5.1) with $(k, \gamma) = (10, 10/\sqrt{N})$ for $S \in \{10^3, 10^4, 10^5\}$

S		BigM		Persp		CP	BCP	BCPc
		Lift	Cut	Lift	Cut			
10^3	Solved	**6**	5	**6**	4	5	5	5
	Time	6.8	107.6	**5.5**	109.6	79.2	116.7	71.0
	Gap(%)	0.00	3.19	0.00	3.58	0.73	1.80	1.15
	#Nodes	334.1	16351.5	106.1	12296.4	2234.6	—	1305.6
	#Cuts	—	—	—	—	322.3	147.4	166.1
10^4	Solved	**6**	4	**6**	4	4	4	4
	Time	106.6	207.7	**89.1**	185.4	338.1	154.6	172.0
	Gap(%)	0.00	2.61	0.00	2.02	2.49	1.97	1.34
	#Nodes	546.3	20097.8	228.3	13973.2	1507.4	—	1378.5
	#Cuts	—	—	—	—	216.8	96.8	165.2
10^5	Solved	3	3	1	3	3	**4**	**4**
	Time	1610.4	650.0	3245.9	535.7	1265.0	**370.8**	497.3
	Gap(%)	21.09	5.30	2.95	5.09	4.77	1.88	3.05
	#Nodes	53.4	9265.2	9.2	6763.1	493.9	—	476.9
	#Cuts	—	—	—	—	91.4	57.5	76.3

Table 5.3: Numerical results for the datasets [59, 90] with $(k, \gamma) = (10, 10/\sqrt{N})$ for $S \in \{10^3, 10^4, 10^5\}$

Data	N	S		BigM		Persp		CP	BCP	BCPc
				Lift	Cut	Lift	Cut			
ind49	49	10^3	Time	**1.8**	4.8	2.3	11.8	6.1	7.3	8.6
			Obj	3.095	3.095	3.095	3.095	3.095	3.095	3.095
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	16	3792	23	4506	211	—	33
			#Cuts	—	—	—	—	76	40	45
		10^4	Time	18.0	21.2	26.0	26.2	47.1	**5.2**	8.3
			Obj	3.343	3.343	3.343	3.343	3.343	3.343	3.343
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	1	3357	1	3815	89	—	0
			#Cuts	—	—	—	—	47	1	7
		10^5	Time	709.8	125.0	>3600	220.0	662.2	**43.1**	103.6
			Obj	3.379	3.379	3.380	3.379	3.379	3.379	3.379
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	31	4101	>1	3489	250	—	0
			#Cuts	—	—	—	—	68	1	12
sbm100	100	10^3	Time	3.3	1.1	3.7	4.2	4.9	**0.6**	0.8
			Obj	4.337	4.337	4.337	4.337	4.337	4.337	4.337
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	1	396	1	4626	14	—	0
			#Cuts	—	—	—	—	21	1	7
		10^4	Time	39.1	8.1	45.0	9.3	51.1	**3.3**	4.0
			Obj	4.397	4.397	4.397	4.397	4.397	4.397	4.397
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	1	1820	1	3274	58	—	0
			#Cuts	—	—	—	—	20	1	7
		10^5	Time	1107.1	65.1	>3600	84.6	482.7	**27.6**	36.6
			Obj	4.364	4.364	4.364	4.364	4.364	4.364	4.364
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	1	412	>1	4676	10	—	0
			#Cuts	—	—	—	—	17	1	7
sp200	200	10^3	Time	30.9	1379.0	**12.5**	>3600	277.1	3441.1	226.6
			Obj	**1.190**	**1.190**	**1.190**	1.274	**1.190**	**1.190**	**1.190**
			Gap(%)	**0.00**	**0.00**	**0.00**	6.88	**0.00**	**0.00**	**0.00**
			#Nodes	7790	323767	645	>87982	20663	—	20670
			#Cuts	—	—	—	—	899	757	953
		10^4	Time	1010.7	>3600	**250.8**	>3600	>3600	>3600	>3600
			Obj	**1.267**	1.275	**1.267**	1.321	1.285	1.273	**1.267**
			Gap(%)	**0.00**	2.51	**0.00**	5.20	5.10	1.20	1.53
			#Nodes	22932	>968714	1052	>100939	>21183	>—	>77070
			#Cuts	—	—	—	—	>1122	>771	>2941
		10^5	Time	>3600	>3600	>3600	>3600	>3600	>3600	>3600
			Obj	1.499	1.322	1.326	1.302	1.331	**1.282**	**1.282**
			Gap(%)	17.48	9.20	8.12	8.47	18.21	**1.97**	4.73
			#Nodes	>68	>59214	>1	>24933	>1873	—	>6095
			#Cuts	—	—	—	—	>145	>533	>406

Table 5.4: Numerical results for the datasets [16, 41] with $(k, \gamma) = (10, 10/\sqrt{N})$ for $S \in \{10^3, 10^4, 10^5\}$

Data	N	S		BigM		Persp		CP	BCP	BCPc
				Lift	Cut	Lift	Cut			
port1	31	10^3	Time	1.1	2.3	1.5	4.3	3.0	**0.8**	1.7
			Obj	4.404	4.404	4.404	4.404	4.404	4.404	4.404
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	1	2301	1	2494	58	—	0
			#Cuts	—	—	—	—	36	1	7
		10^4	Time	11.7	11.4	24.0	17.7	18.0	**4.8**	8.2
			Obj	4.374	4.374	4.374	4.374	4.374	4.374	4.374
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	1	850	1	3005	46	—	0
			#Cuts	—	—	—	—	17	1	8
		10^5	Time	468.6	163.3	1933.2	**15.8**	265.8	39.5	64.6
			Obj	4.264	4.264	4.264	4.264	4.264	4.264	4.264
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	1	2433	1	253	39	—	0
			#Cuts	—	—	—	—	25	1	7
port2	89	10^3	Time	**7.4**	>3600	7.5	>3600	>3600	>3600	>3600
			Obj	**1.773**	2.119	**1.773**	2.002	1.833	2.022	**1.773**
			Gap(%)	**0.00**	19.13	**0.00**	14.61	8.52	10.80	6.90
			#Nodes	1432	>287,645	577	>152,730	>216,174	—	>36,644
			#Cuts	—	—	—	—	>15,007	>7190	>3899
		10^4	Time	**311.2**	>3600	429.0	>3600	>3600	>3600	>3600
			Obj	**1.938**	2.054	**1.938**	1.981	2.009	2.181	1.943
			Gap(%)	**0.00**	13.17	**0.00**	6.90	9.83	10.61	6.53
			#Nodes	5715	>261,628	2429	>204,649	>22,111	—	>18,077
			#Cuts	—	—	—	—	>3654	>2695	>2548
		10^5	Time	>3600	>3600	>3600	>3600	>3600	>3600	>3600
			Obj	2.030	2.140	2.039	2.074	1.988	2.136	**1.950**
			Gap(%)	9.03	18.61	9.17	17.76	**8.35**	9.30	13.55
			#Nodes	>481	>99,640	>1	>40,749	>1756	—	>927
			#Cuts	—	—	—	—	>406	>350	>285
port5	225	10^3	Time	8.3	250.0	**7.9**	40.6	144.8	157.8	130.3
			Obj	**2.933**	**2.933**	**2.933**	2.650†	**2.933**	**2.933**	**2.933**
			Gap(%)	**0.00**	**0.00**	**0.00**	0.00	**0.00**	**0.00**	**0.00**
			#Nodes	368	45,800	21	4674	7982	—	7499
			#Cuts	—	—	—	—	775	446	489
		10^4	Time	246.1	668.1	**113.0**	211.4	1387.1	500.1	587.5
			Obj	**3.147**	**3.147**	**3.147**	**3.147**	**3.147**	**3.147**	**3.147**
			Gap(%)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
			#Nodes	428	42,363	317	9056	8269	—	7347
			#Cuts	—	—	—	—	701	517	727
		10^5	Time	>3600	>3600	>3600	>3600	>3600	**2351.6**	3300.8
			Obj	∞	3.246	3.143	3.276	3.173	**3.138**	**3.138**
			Gap(%)	100.00	3.98	0.44	4.29	2.05	**0.00**	**0.00**
			#Nodes	>0	>20,833	>61	>16,790	>2140	—	5697
			#Cuts	—	—	—	—	>219	352	519

† Gurobi returned an incorrect optimal objective value because of numerical instability.

Table 5.5: Summary of numerical results for the four datasets (`ind49`, `sbm100`, `port1`, and `port5`) with $(S, \gamma) = (10^5, 10/\sqrt{N})$ for $k \in \{5, 10, 15\}$

k		BigM		Persp		CP	BCP	BCPc
		Lift	Cut	Lift	Cut			
5	Solved	2	2	1	**4**	2	2	2
	Time	1863.3	795.3	3422.7	**766.0**	1902.8	789.4	905.2
	Gap(%)	50.00	4.36	7.12	0.00	4.26	0.36	1.68
	#Nodes	36.9	8430.2	35.2	7914.4	968.1	—	1817.6
	#Cuts	—	—	—	—	167.7	166.8	224.8
10	Solved	2	2	0	2	2	**4**	**4**
	Time	1789.2	592.9	3600.0	719.7	1431.3	**315.0**	480.8
	Gap(%)	50.00	1.99	0.22	2.14	1.02	0.00	0.00
	#Nodes	7.3	5440.8	27.5	8262.3	563.0	—	661.4
	#Cuts	—	—	—	—	92.5	53.1	91.1
15	Solved	2	**4**	1	**4**	2	**4**	**4**
	Time	1662.2	305.9	3558.8	268.0	1268.8	**53.5**	88.1
	Gap(%)	50.00	0.00	0.00	0.00	0.98	0.00	0.00
	#Nodes	0.5	2539.6	1.0	3769.4	494.1	—	0.0
	#Cuts	—	—	—	—	76.5	1.0	8.9

Chapter 6

Conclusion and Prospects

Now let us conclude this thesis. Specifically, we will summarize the contributions described herein and offer some possible future directions for research.

6.1 Summary

This thesis focused on solving mixed-integer semidefinite optimization problems. The mixed-integer semidefinite optimization problem involves minimizing or maximizing a linear objective function subject to constraints whereby a given matrix formed from the decision variables is positive semidefinite, and some of the variables are integer-valued. Since this problem includes nonlinearity and discreteness, various practical optimization problems can be formulated as a mixed-integer semidefinite optimization problem.

In this thesis, we devised efficient cutting-plane algorithms for solving mixed-integer semidefinite optimization problems. First, we proposed a general-purpose cutting-plane algorithm for solving the standard form of mixed-integer semidefinite optimization problems. This method allows us to handle the general mixed-integer semidefinite optimization problem with existing state-of-the-art mixed-integer optimization solvers such as Gurobi and CPLEX. Second, we devised specialized cutting-plane algorithms for some important applications of mixed-integer semidefinite optimization. In statistics and financial engineering, we sometimes encounter large-sized mixed-integer semidefinite problem instances, and such problems are difficult to solve even with general-purpose algorithms. We focused on a variable selection and portfolio selection problem formulated as mixed-integer semidefinite problems and proposed efficient cutting-plane algorithms that exploit the structures of these problems.

In Chapter 2, we described a general-purpose framework for solving the standard form of mixed-integer semidefinite optimization problems. First, we formulated a cutting-plane algorithm to solve mixed-integer semidefinite optimization problems and proved its convergence properties. Then, we developed a branch-and-cut algorithm where cutting planes are added dynamically to the relaxed mixed-integer linear optimization problem during a branch-and-bound procedure. The experimental results confirmed that our branch-and-cut algorithm could solve three different mixed-integer semidefinite optimization problems: random instances, computing restricted isometry constants, and robust truss topology design. Our algorithm is based on the branch-and-bound procedure for solving mixed-integer semidefinite optimization problems, so it can use a warm-starting strategy to solve a series of continuous relaxation problems efficiently. Moreover, our algorithm does not solve any semidefinite optimization problems, which makes the mixed-integer semidefinite optimization computation

very stable and practical.

In Chapter 3, we addressed the problem of selecting the best subset of explanatory variables subject to an upper bound on the condition number for eliminating multicollinearity from linear regression models. The first contribution of this study is a novel computational framework for eliminating multicollinearity based on the mixed-integer semidefinite optimization formulation. This framework reformulates the subset selection problem as a single mixed-integer semidefinite optimization problem. The second contribution is the establishment of a high-performance cutting-plane algorithm. In this algorithm, we generate strong cutting planes with a heuristic search and reduce the number of relaxation problems to be solved. While numerical experiments show that our mixed-integer semidefinite optimization formulation can only be applied to small-sized instances at present, this chapter provides a new statistical application of mixed-integer semidefinite optimization formulation. Moreover, we found that our cutting-plane algorithm frequently provided a better subset of variables than did the common local search algorithms.

In Chapter 4, we studied moment-based distributionally robust portfolio optimization problems with a cardinality constraint. Because of the discreteness of the cardinality constraint, this problem is formulated as a mixed-integer semidefinite optimization problem, which is hard to solve exactly when the number of investable assets is large. We reformulated the problem as a bilevel optimization problem and devised a cutting-plane algorithm for solving the upper-level problem. In addition, we applied the technique of positive semidefinite matrix completion to the lower-level problem in order to efficiently generate cutting planes. The computational results indicate that our cutting-plane algorithm is more effective than the existing general-purpose mixed-integer semidefinite optimization solver, especially when the number of investable assets is large. In addition, the out-of-sample investment performances given by the cardinality-constrained distributionally robust model were better than those of the cardinality-constrained mean-variance model.

Chapter 5 dealt with cardinality-constrained mean-CVaR portfolio optimization problems, and we extended the cutting-plane algorithm for the distributionally-robust portfolio optimization problem discussed in Chapter 4. This model is formulated as a mixed-integer linear optimization problem, which is a special case of mixed-integer semidefinite optimization problems. Since its problem size depends not only on the number of investable assets but also on the number of asset return scenarios, the computational efficiency decreases when the number of scenarios is large. To overcome this challenge, we propose a specialized cutting-plane algorithm named the *bilevel cutting-plane algorithm* for solving the cardinality-constrained mean-CVaR portfolio optimization problem. We extended the cutting-plane algorithm discussed in Chapter 4 so that it incorporates another cutting-plane algorithm that efficiently minimizes CVaR. The computational results indicate that our cutting-plane algorithms are very effective, especially when the number of scenarios is large.

6.2 Future directions

The results in this thesis suggest that the cutting-plane algorithms could be a powerful tool for solving mixed-integer semidefinite optimization problems. However, there is still much room for research on mixed-integer semidefinite optimization.

While our general-purpose algorithms delivered better computational performances than the existing methods for several problem instances in Chapter 2, our algorithms are not always faster for general problem instances. Thus, we have to continue improving the efficiency of our algorithms to make mixed-integer semidefinite optimization more practical. In Chapters 3 to 5, we devised specialized cutting-plane algorithms that exploit the individual problems' structure to improve the computational efficiency. In particular, we proposed to generate strong cutting planes with a heuristic search for subset selection problems in Chapter 3. Also, we used the technique of positive semidefinite matrix completion to calculate cutting planes efficiently in Chapter 4. To develop more efficient and practical general-purpose algorithms, these techniques should be incorporated into our general-purpose algorithms for mixed-integer semidefinite optimization. Thus, we would be interested in characterizing whether these specialized techniques can be applied to a given mixed-integer semidefinite optimization problem.

Of course, it also makes sense to extend the specialized cutting-plane algorithms not only to mixed-integer semidefinite optimization problems but also to other problems appearing in the corresponding fields. In the case of subset selection, it is interesting to extend the cutting-plane algorithm so that it can deal with other variable selection problems for statistics and machine learning fields. For portfolio optimization, various practical and distributionally robust risk measures have been proposed. We expect that our cutting-plane algorithm can be extended to handle such risk measures.

While mixed-integer semidefinite optimization has the potential to be applied to a wide range of practical decision-making problems, vigorous research has just started. We will continue refining both general-purpose and specialized algorithms for mixed-integer semidefinite optimization, and expand the horizons of its applications. As our modern society is becoming more and more complex, we hope that this work will be a new avenue for solving some of its fundamental problems.

References

[1] T. Achterberg. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[2] A. Agarwal, S. N. Negahban, and M. J. Wainwright. Stochastic optimization and sparse statistical recovery: Optimal algorithms for high dimensions. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, pages 1538–1546, 2012.

[3] K. C. Ágoston. CVaR minimization by the SRA algorithm. *Central European Journal of Operations Research*, 20(4):623–632, 2012.

[4] S. Ahmed. Convexity and decomposition of mean-risk stochastic programs. *Mathematical Programming*, 106(3):433–446, 2006.

[5] F. A. Al-Khayyal and J. E. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286, 1983.

[6] S. Alexander, T. F. Coleman, and Y. Li. Minimizing CVaR and VaR for a portfolio of derivatives. *Journal of Banking and Finance*, 30(2):583–605, 2006.

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

[8] D. Aloise and P. Hansen. A branch-and-cut SDP-based algorithm for minimum sum-of-squares clustering. *Pesquisa Operacional*, 29(3):503–516, 2009.

[9] E. Angelelli, R. Mansini, and M. G. Speranza. A comparison of MAD and CVaR models with real features. *Journal of Banking and Finance*, 32(7):1188–1197, 2008.

[10] M. F. Anjos, B. Ghaddar, L. Hupp, F. Liers, and A. Wiegele. Solving k-way graph partitioning problems to optimality: The impact of semidefinite relaxations and the bundle method. In M. Jünger and G. Reinelt, editors, *Facets of Combinatorial Optimization*, pages 355–386. Springer, 2013.

[11] M. Armbruster, M. Fügenschuh, C. Helmberg, and A. Martin. LP and SDP branch-and-cut algorithms for the minimum graph bisection problem: a computational comparison. *Mathematical Programming Computation*, 4(3):275–306, 2012.

[12] P. Artzner, F. Delbaen, J. M. Eber, and H. David. Coherent measures of risk. *Mathematical Finance*, 9(3):203–228, 1999.

[13] R. Baraniuk. Compressive sensing [lecture notes]. *IEEE Signal Processing Magazine*, 24(4):118–121, 2007.

[14] E. M. L. Beale. Two transportation problems. In *Proceedings of the Third International Conference on Operational Research*, pages 780–788, 1963.

[15] E. M. L. Beale and J. Tomlin. Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. In *Proceedings of the Fifth International Conference on Operational Research*, pages 447–454, 1970.

[16] J. E. Beasley. OR-library: Distributing test problems by electronic mail. *The Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

[17] G. Beliakov and A. Bagirov. Non-smooth optimization methods for computation of the conditional value-at-risk and portfolio optimization. *Optimization*, 55(5-6):459–479, 2006.

[18] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press, 2009.

[19] A. Ben-Tal and A. Nemirovski. Robust truss topology design via semidefinite programming. *SIAM Journal on Optimization*, 7(4):991–1016, 1997.

[20] A. Ben-Tal and A. Nemirovski. Robust optimization-methodology and applications. *Mathematical Programming*, 92(3):453–480, 2002.

[21] S. J. Benson, Y. Ye, and X. Zhang. Solving large-scale sparse semidefinite programs for combinatorial optimization. *SIAM Journal on Optimization*, 10(2):443–461, 2000.

[22] L. Berk and D. Bertsimas. Certifiably optimal sparse principal component analysis. *Mathematical Programming Computation*, 11(3):381–420, 2019.

[23] D. Bertsekas, A. Nedić, and A. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, 2003.

[24] D. Bertsimas and R. Cory-Wright. A scalable algorithm for sparse portfolio selection. *INFORMS Journal on Computing*, 2022. https://doi.org/10.1287/ijoc.2021.1127.

[25] D. Bertsimas, R. Cory-Wright, and J. Pauphilet. Solving large-scale sparse PCA to certifiable (near) optimality. *arXiv preprint arXiv:2005.05195*, 2020.

[26] D. Bertsimas, R. Cory-Wright, and J. Pauphilet. A unified approach to mixed-integer optimization problems with logical constraints. *SIAM Journal on Optimization*, 31(3):2340–2367, 2021.

[27] D. Bertsimas, C. Darnell, and R. Soucy. Portfolio construction through mixed-integer programming at Grantham, Mayo, Van Otterloo and Company. *Interfaces*, 29(1):49–66, 1999.

[28] D. Bertsimas, I. Dunning, and M. Lubin. Reformulation versus cutting-planes for robust optimization. *Computational Management Science*, 13(2):195–217, 2016.

[29] D. Bertsimas, V. Gupta, and N. Kallus. Data-driven robust optimization. *Mathematical Programming*, 167(2):235–292, 2017.

[30] D. Bertsimas and A. King. OR forum—an algorithmic approach to linear regression. *Operations Research*, 64(1):2–16, 2016.

[31] D. Bertsimas and A. King. Logistic regression: From art to science. *Statistical Science*, 32(3):367–384, 2017.

[32] D. Bertsimas, A. King, and R. Mazumder. Best subset selection via a modern optimization lens. *The Annals of Statistics*, 44(2):813–852, 2016.

[33] D. Bertsimas, J. Lamperski, and J. Pauphilet. Certifiably optimal sparse inverse covariance estimation. *Mathematical Programming*, 184(1-2):491–530, 2019.

[34] D. Bienstock. Computational study of a family of mixed-integer quadratic programming problems. *Mathematical Programming*, 74(2):121–140, 1996.

[35] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

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

[37] M. Branda, M. Bucher, M. Červinka, and A. Schwartz. Convergence of a scholtes-type regularization method for cardinality-constrained optimization problems with an application in sparse robust portfolio optimization. *Computational Optimization and Applications*, 70(2):503–530, 2018.

[38] G. Braun, S. Fiorini, S. Pokutta, and D. Steurer. Approximation limits of linear programs (beyond hierarchies). *Mathematics of Operations Research*, 40(3):756–772, 2015.

[39] S. B. Çay, I. Pólik, and T. Terlaky. Warm-start of interior point methods for second order cone optimization via rounding over optimal jordan frames. Technical Report 17T-00, Lehigh University, 2017.

[40] A. Cerveira, A. Agra, F. Bastos, and J. Gromicho. A new branch and bound method for a discrete truss topology design problem. *Computational Optimization and Applications*, 54(1):163–187, 2013.

[41] T. J. Chang, N. Meade, E. B. John, and Y. M. Sharaiha. Heuristics for cardinality constrained portfolio optimisation. *Computers and Operations Research*, 27(13):1271–1302, 2000.

[42] S. Chatterjee and A. S. Hadi. *Regression Analysis by Example, Fifth Edition*. John Wiley & Sons, 2012.

[43] R. Cheng and J. Gao. On cardinality constrained mean-cvar portfolio optimization. In *Proceedings of the 27th Chinese Control and Decision Conference*, pages 1074–1079, 2015.

[44] I. G. Chong and C. H. Jun. Performance of some variable selection methods when multicollinearity is present. *Chemometrics and Intelligent Laboratory Systems*, 78(1-2):103–112, 2005.

[45] C. Coey, M. Lubin, and J. P. Vielma. Outer approximation with conic certificates for mixed-integer convex problems. *Mathematical Programming Computation*, 12(2):249–293, 2020.

[46] J. Czyzyk, M. Mesnier, and J. More. The NEOS server. *IEEE Computational Science and Engineering*, 5(3):68–75, 1998.

[47] E. Delage and Y. Ye. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations Research*, 58(3):595–612, 2010.

[48] V. DeMiguel, L. Garlappi, F. J. Nogales, and R. Uppal. A generalized approach to portfolio optimization: Improving performance by constraining portfolio norms. *Management Science*, 55(5):798–812, 2009.

[49] A. V. Dorugade and D. N. Kashid. Variable selection in linear regression based on ridge estimator. *Journal of Statistical Computation and Simulation*, 80(11):1211–1224, 2010.

[50] M. Efroymson. Multiple regression analysis. In A. Ralston and H. Wilf, editors, *Mathematical Methods for Digital Computers*, pages 191–203. John Wiley & Sons, 1960.

[51] L. El Ghaoui, M. Oks, and F. Oustry. Worst-case value-at-risk and robust portfolio optimization: A conic programming approach. *Operations Research*, 51(4):543–556, 2003.

[52] C. I. Fábián. Handling CVaR objectives and constraints in two-stage stochastic models. *European Journal of Operational Research*, 191(3):888–911, 2008.

[53] F. J. Fabozzi, D. Huang, and G. Zhou. Robust portfolios: contributions from operations research and finance. *Annals of Operations Research*, 176(1):191–220, 2010.

[54] D. E. Farrar and R. R. Glauber. Multicollinearity in regression analysis: The problem revisited. *The Review of Economics and Statistics*, 49(1):92, 1967.

[55] S. Foucart and M. J. Lai. Sparsest solutions of underdetermined linear systems via ℓ_q-minimization for $0 < q \leq 1$. *Applied and Computational Harmonic Analysis*, 26(3):395–407, 2009.

[56] A. Frangioni, F. Furini, and C. Gentile. Approximated perspective relaxations: a project and lift approach. *Computational Optimization and Applications*, 63(3):705–735, 2016.

[57] A. Frangioni and C. Gentile. SDP diagonalizations and perspective cuts for a class of nonseparable MIQP. *Operations Research Letters*, 35(2):181–185, 2007.

[58] l. E. Frank and J. H. Friedman. A statistical view of some chemometrics regression tools. *Technometrics*, 35(2):109–135, 1993.

[59] K. R. French. Kenneth R. French - data library. https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html, Accessed 17 July 2020.

[60] M. Fukuda, M. Kojima, K. Murota, and K. Nakata. Exploiting sparsity in semidefinite programming via matrix completion I: General framework. *SIAM Journal on Optimization*, 11(3):647–674, 2001.

[61] T. Gally and M. E. Pfetsch. Computing restricted isometry constants via mixed-integer semidefinite programming. *Oprimization Online*, 2016.

[62] T. Gally, M. E. Pfetsch, and S. Ulbrich. A framework for solving mixed-integer semidefinite programs. *Optimization Methods and Software*, 33(3):594–632, 2018.

[63] A. Gleixner, L. Eifler, T. Gally, G. Gamrath, P. Gemander, R. L. Gottwald, G. Hendel, C. Hojny, T. Koch, M. Miltenberger, B. Müller, M. Pfetsch, C. Puchert, D. Rehfeldt, F. Schlösser, F. Serrano, Y. Shinano, J. M. Viernickel, S. Vigerske, D. Weninger, J. T. Witt, and J. Witzig. The SCIP optimization suite 5.0. Technical Report 17–61, Zuse Institute Berlin, 2017.

[64] J.-L. Goffin and J.-P. Vial. Convex nondifferentiable optimization: A survey focused on the analytic center cutting plane method. *Optimization Methods and Software*, 17(5):805–867, 2002.

[65] J. Goh and M. Sim. Distributionally robust optimization and its tractable approximations. *Operations Research*, 58(4-part-1):902–917, 2010.

[66] D. Goldfarb and G. Iyengar. Robust portfolio selection problems. *Mathematics of Operations Research*, 28(1):1–38, 2003.

[67] R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–279, 1958.

[68] J. Gondzio. Warm start of the primal-dual method applied in the cutting-plane scheme. *Mathematical Programming*, 83(1-3):125–143, 1998.

[69] J. Gotoh, M. J. Kim, and A. E. B. Lim. Calibration of distributionally robust empirical optimization models. *Operations Research*, 69(5):1630–1650, 2021.

[70] J. Gotoh, K. Shinozaki, and A. Takeda. Robust portfolio techniques for mitigating the fragility of CVaR minimization and generalization to coherent risk measures. *Quantitative Finance*, 13(10):1621–1635, 2013.

[71] J. Gotoh and A. Takeda. On the role of norm constraints in portfolio selection. *Computational Management Science*, 8(4):323–353, 2011.

[72] C. Gregory, K. Darby-Dowman, and G. Mitra. Robust optimization and portfolio selection: The cost of robustness. *European Journal of Operational Research*, 212(2):417–428, 2011.

[73] I. E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, 3(3):227–252, 2002.

[74] O. Günlük and J. Linderoth. Perspective reformulations of mixed integer nonlinear programs with indicator variables. *Mathematical Programming*, 124(1-2):183–205, 2010.

[75] O. Günlük and J. Linderoth. Perspective reformulation and applications. In *Mixed Integer Nonlinear Programming*, pages 61–89. Springer, 2011.

[76] R. Gunst and J. Webster. Regression analysis and problems of multicollinearity. *Communications in Statistics*, 4(3):277–292, 1975.

[77] H. Mittelmann. Benchmarks for optimization software. http://plato.asu.edu/bench.html, Accessed 6 Junuary 2021.

[78] A. S. Hadi and R. F. Ling. Some cautionary notes on the use of principal components regression. *The American Statistician*, 52(1):15, 1998.

[79] S. Han, A. Gómez, and A. Atamtürk. 2×2 convexifications for convex quadratic optimization with indicator variables. *arXiv preprint arXiv:2004.07448*, 2020.

[80] W. K. Haneveld and M. H. van der Vlerk. Integrated chance constraints: Reduced forms and an algorithm. *Computational Management Science*, 3(4):245–269, 2006.

[81] C. Helmberg and F. Rendl. Solving quadratic (0, 1)-problems by semidefinite programs and cutting planes. *Mathematical Programming*, 82(3):291–315, 1998.

[82] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I*. Springer, 1993.

[83] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[84] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 2nd edition, 2012.

[85] G. Iyengar and A. K. C. Ma. Fast gradient descent method for Mean-CVaR optimization. *Annals of Operations Research*, 205(1):203–212, 2013.

[86] J. J. E. Kelley. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

[87] N. H. Jadhav, D. N. Kashid, and S. R. Kulkarni. Subset selection in multiple linear regression in the presence of outlier and multicollinearity. *Statistical Methodology*, 19:44–59, 2014.

[88] I. T. Jolliffe. A note on the use of principal components in regression. *Applied Statistics*, 31(3):300, 1982.

[89] S. Joshi and S. Boyd. Sensor selection via convex optimization. *IEEE Transactions on Signal Processing*, 57(2):451–462, 2009.

[90] Kaggle. S&P 500 stock data. https://www.kaggle.com/camnugent/sandp500, Accessed 23 December 2020.

[91] M. Kaut, H. Vladimirou, S. W. Wallace, and S. A. Zenios. Stability analysis of portfolio management with conditional value-at-risk. *Quantitative Finance*, 7(4):397–409, 2007.

[92] K. Kimura and H. Waki. Minimization of akaike's information criterion in linear regression analysis via mixed integer nonlinear program. *Optimization Methods and Software*, 33(3):633–649, 2017.

[93] A. J. Kleywegt, A. Shapiro, and T. H. de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

[94] K. Kobayashi and Y. Takano. A branch-and-cut algorithm for solving mixed-integer semidefinite optimization problems. *Computational Optimization and Applications*, 75(2):493–513, 2019.

[95] K. Kobayashi, Y. Takano, and K. Nakata. Bilevel cutting-plane algorithm for cardinality-constrained mean-CVaR portfolio optimization. *Journal of Global Optimization*, 81(2):493–528, 2021.

[96] K. Kobayashi, Y. Takano, and K. Nakata. Cardinality-constrained distributionally robust portfolio optimization. *arXiv preprint arXiv:2112.12454*, 2021.

[97] H. Konno, J. Gotoh, T. Uno, and A. Yuki. A cutting plane algorithm for semi-definite programming problems with applications to failure discriminant analysis. *Journal of Computational and Applied Mathematics*, 146(1):141–154, 2002.

[98] H. Konno, N. Kawadai, and H. Tuy. Cutting plane algorithms for nonlinear semi-definite programming problems with applications. *Journal of Global Optimization*, 25(2):141–155, 2003.

[99] H. Konno and Y. Takaya. Multi-step methods for choosing the best set of variables in regression analysis. *Computational Optimization and Applications*, 46(3):417–426, 2010.

[100] H. Konno, H. Waki, and A. Yuuki. Portfolio optimization under lower partial risk measures. *Asia-Pacific Financial Markets*, 9(2):127–140, 2002.

[101] H. Konno and R. Yamamoto. Choosing the best set of variables in regression analysis using integer programming. *Journal of Global Optimization*, 44(2):273–282, 2009.

[102] K. Krishnan and J. E. Mitchell. A unifying framework for several cutting plane methods for semidefinite programming. *Optimization Methods and Software*, 21(1):57–74, 2006.

[103] A. Künzi-Bay and J. Mayer. Computational aspects of minimizing conditional value-at-risk. *Computational Management Science*, 3(1):3–27, 2006.

[104] C. Lim, H. D. Sherali, and S. Uryasev. Portfolio optimization by minimizing Conditional Value-at-Risk via nondifferentiable optimization. *Computational Optimization and Applications*, 46(3):391–415, 2010.

[105] H. Liu, X. Wang, T. Yao, R. Li, and Y. Ye. Sample average approximation with sparsity-inducing penalty for high-dimensional stochastic programming. *Mathematical Programming*, 178(1-2):69–108, 2018.

[106] J. Lofberg. YALMIP : a toolbox for modeling and optimization in MATLAB. In *2004 IEEE International Conference on Robotics and Automation*, pages 284–289.

[107] S. Lotfi and S. A. Zenios. Robust VaR and CVaR optimization under joint ambiguity in distributions, means, and covariances. *European Journal of Operational Research*, 269(2):556–576, 2018.

[108] M. Lubin, E. Yamangil, R. Bent, and J. P. Vielma. Polyhedral approximation in mixed-integer convex optimization. *Mathematical Programming*, 172(1-2):139–168, 2017.

[109] M. Lichman. UCI machine learning repository. http://archive.ics.uci.edu/ml, Accessed 5 September 2021.

[110] M. Mäkelä. Survey of bundle methods for nonsmooth optimization. *Optimization Methods and Software*, 17(1):1–29, 2002.

[111] N. M. Manousakis and G. N. Korres. Semidefinite programming for optimal placement of PMUs with channel limits considering pre-existing SCADA and PMU measurements. In *2016 Power Systems Computation Conference*, pages 1–7, 2016.

[112] E. R. Mansfield and B. P. Helms. Detecting multicollinearity. *The American Statistician*, 36(3):158, 1982.

[113] R. Mansini, W. Ogryczak, and M. G. Speranza. Twenty years of linear programming based portfolio optimization. *European Journal of Operational Research*, 234(2):518–535, 2014.

[114] H. Markowitz. Portfolio Selection. *The Journal of Finance*, 7(1):77–91, 1952.

[115] W. F. Massy. Principal components regression in exploratory statistical research. *Journal of the American Statistical Association*, 60(309):234–256, 1965.

[116] G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I — convex underestimating problems. *Mathematical Programming*, 10(1):147–175, 1976.

[117] H. Mittelmann. An independent benchmarking of SDP and SOCP solvers. *Mathematical Programming*, 95(2):407–430, 2003.

[118] R. Miyashiro and Y. Takano. Mixed integer second-order cone programming formulations for variable selection in linear regression. *European Journal of Operational Research*, 247(3):721–731, 2015.

[119] R. Miyashiro and Y. Takano. Subset selection by mallows' c_p: A mixed integer programming approach. *Expert Systems with Applications*, 42(1):325–331, 2015.

[120] MOSEK ApS. MOSEK. https://www.mosek.com/, Accessed 20 October 2021.

[121] K. Nakata, K. Fujisawa, M. Fukuda, M. Kojima, and K. Murota. Exploiting sparsity in semidefinite programming via matrix completion II: implementation and numerical results. *Mathematical Programming*, 95(2):303–327, 2003.

[122] K. Natarajan, M. Sim, and J. Uichanco. Tractable robust expected utility and risk models for portfolio optimization. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, 20(4):695–731, 2010.

[123] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994.

[124] N. Noyan, B. Balcik, and S. Atakan. A stochastic optimization model for designing last mile relief networks. *Transportation Science*, 50(3):1092–1113, 2015.

[125] W. Ogryczak and T. Śliwiński. On solving the dual for portfolio selection by optimizing Conditional Value at Risk. *Computational Optimization and Applications*, 50(3):591–595, 2011.

[126] B. P. G. V. Parys, P. M. Esfahani, and D. Kuhn. From data to decisions: Distributionally robust optimization is optimal. *Management Science*, 67(6):3387–3402, 2020.

[127] J. Peng and Y. Xia. A new theoretical framework for k-means-type clustering. In W. Chu and L. T. Young, editors, *Foundations and Advances in Data Mining*, pages 79–96. Springer, 2005.

[128] A. F. Perold. Large-scale portfolio optimization. *Management Science*, 30(10):1143–1160, 1984.

[129] G. C. Pflug. Some remarks on the value-at-risk and the conditional value-at-risk. In S. Uryasev, editor, *Nonconvex Optimization and Its Applications*, pages 272–281. Springer, 2000.

[130] A. Philipp, S. Ulbrich, Y. Cheng, and M. Pesavento. Multiuser downlink beamforming with interference cancellation using a SDP-based branch-and-bound algorithm. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7724–7728, 2014.

[131] I. Popescu. Robust mean-covariance solutions for stochastic optimization. *Operations Research*, 55(1):98–112, 2007.

[132] I. Quesada and I. Grossmann. An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. *Computers and Chemical Engineering*, 16(10-11):937–947, 1992.

[133] H. Rahimian and S. Mehrotra. Distributionally robust optimization: A review. *arXiv preprint arXiv:1908.05659*, 2019.

[134] F. Rendl. Semidefinite relaxations for integer programming. In M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, editors, *50 Years of Integer Programming 1958-2008*, pages 687–726. Springer, 2010.

[135] F. Rendl, G. Rinaldi, and A. Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121(2):307–335, 2010.

[136] R. T. Rockafellar. *Convex analysis*. Princeton Mathematical Series. Princeton University Press, 1970.

[137] R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *The Journal of Risk*, 2(3):21–41, 2000.

[138] R. T. Rockafellar and S. Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking and Finance*, 26(7):1443–1471, 2002.

[139] C. Rowe and J. Maciejowski. An efficient algorithm for mixed integer semidefinite optimisation. In *Proceedings of the 2003 American Control Conference*, pages 4730–4735, 2003.

[140] T. Sato, Y. Takano, and R. Miyashiro. Piecewise-linear approximation for feature subset selection in a sequential logit model. *Journal of the Operations Research Society of Japan*, 60(1):1–14, 2017.

[141] T. Sato, Y. Takano, R. Miyashiro, and A. Yoshise. Feature subset selection for logistic regression via mixed integer optimization. *Computational Optimization and Applications*, 64(3):865–880, 2016.

[142] H. Scarf. A min-max solution of an inventory problem. *Studies in The Mathematical Theory of Inventory and Production*, pages 201–209, 1958.

[143] N. Schwertman and D. Allen. Smoothing an indefinite variance-covariance matrix. *Journal of Statistical Computation and Simulation*, 9(3):183–194, 1979.

[144] A. Shapiro. Monte carlo sampling methods. In *Handbooks in Operations Research and Management Science*, pages 353–425. Elsevier, 2003.

[145] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on Stochastic Programming: Modeling and Theory*. Society for Industrial and Applied Mathematics, 2009.

[146] A. Skajaa, E. D. Andersen, and Y. Ye. Warmstarting the homogeneous and self-dual interior point method for linear and conic quadratic problems. *Mathematical Programming Computation*, 5(1):1–25, 2012.

[147] R. Sotirov. SDP relaxations for some combinatorial optimization problems. In M. F. Anjos and J. B. Lasserre, editors, *Handbook on Semidefinite, Conic and Polynomial Optimization*, pages 795–819. Springer, 2012.

[148] J. F. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1-4):625–653, 1999.

[149] Y. Takano, K. Nanjo, N. Sukegawa, and S. Mizuno. Cutting plane algorithms for mean-CVaR portfolio optimization with nonconvex transaction costs. *Computational Management Science*, 12(2):319–340, 2014.

[150] A. Takeda and T. Kanamori. A robust approach based on conditional value-at-risk measure to statistical learning problems. *European Journal of Operational Research*, 198(1):287–296, 2009.

[151] R. Tamura, K. Kobayashi, Y. Takano, R. Miyashiro, K. Nakata, and T. Matsui. Best subset selection for eliminating multicollinearity. *Journal of the Operations Research Society of Japan*, 60(3):321–336, 2017.

[152] K. Tanaka and R. Yamamoto. Identification of best discrimination surface by mixed-integer semi-definite programming for support vector machine. *International Journal of Financial Engineering*, 2021. https://doi.org/10.1142/S2424786321500420.

[153] J. A. Taylor, N. Luangsomboon, and D. Fooladivanda. Allocating sensors and actuators via optimal estimation and control. *IEEE Transactions on Control Systems Technology*, 25(3):1060–1067, 2017.

[154] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[155] M. J. Todd. Semidefinite optimization. *Acta Numerica*, 10:515–560, 2001.

[156] K. C. Toh, M. J. Todd, and R. H. Tütüncü. SDPT3 — A Matlab software package for semidefinite programming, version 1.3. *Optimization Methods and Software*, 11(1-4):545–581, 1999.

[157] X. Tong, L. Qi, F. Wu, and H. Zhou. A smoothing method for solving portfolio optimization with CVaR and applications in allocation of generation asset. *Applied Mathematics and Computation*, 216(6):1723–1740, 2010.

[158] M. Torchio, L. Magni, and D. M. Raimondo. A mixed integer SDP approach for the optimal placement of energy storage devices in power grids with renewable penetration. In *Proceedings of the 2015 American Control Conference*, pages 3892–3897, 2015.

[159] S. F. Tóth, M. E. McDill, N. Könnyü, and S. George. Testing the use of lazy constraints in solving area-based adjacency formulations of harvest scheduling models. *Forest Science*, 59(2):157–176, 2013.

[160] R. Tütüncü and M. Koenig. Robust asset allocation. *Annals of Operations Research*, 132(1-4):157–187, 2004.

[161] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[162] T. Westerlund and F. Pettersson. An extended cutting plane method for solving convex MINLP problems. *Computers and Chemical Engineering*, 19:131–136, 1995.

[163] H. P. Williams. *Model Building in Mathematical Programming*. John Wiley & Sons, 2013.

[164] H. Wold. Estimation of principal components and related models by iterative least squares. In P. Krishnaiaah, editor, *Multivariate Analysis*, pages 391–420. Academic Press, 1966.

[165] S. Wold, A. Ruhe, H. Wold, and I. W. J. Dunn. The collinearity problem in linear regression. the partial least squares (PLS) approach to generalized inverses. *SIAM Journal on Scientific and Statistical Computing*, 5(3):735–743, 1984.

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

[167] Yahoo Japan Corporation. Yahoo! Finance. https://finance.yahoo.co.jp, Accessed 7 October 2021.

[168] 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 M. Anjos and J. Lasserre, editors, *Handbook on Semidefinite, Conic and Polynomial Optimization*, pages 687–713. Springer, 2012.

[169] M. Yamashita, K. Fujisawa, and M. Kojima. Implementation and evaluation of SDPA 6.0 (semidefinite programming algorithm 6.0). *Optimization Methods and Software*, 18(4):491–505, 2003.

[170] R. Yokoyama, Y. Shinano, S. Taniguchi, M. Ohkura, and T. Wakui. Optimization of energy supply systems by MILP branch and bound method in consideration of hierarchical relationship between design and operation. *Energy Conversion and Management*, 92:92–104, 2015.

[171] K. Yonekura and Y. Kanno. Global optimization of robust truss topology via mixed integer semidefinite programming. *Optimization and Engineering*, 11(3):355–379, 2010.

[172] Y. Zhang, S. Shen, and S. A. Erdogan. Solving 0–1 semidefinite programs for distributionally robust allocation of surgery blocks. *Optimization Letters*, 12(7):1503–1521, 2018.

[173] X. Zheng, X. Sun, and D. Li. Improving the performance of MIQP solvers for quadratic programs with cardinality and minimum threshold constraints: A semidefinite program approach. *INFORMS Journal on Computing*, 26(4):690–703, 2014.

[174] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[175] S. Zymler, D. Kuhn, and B. Rustem. Distributionally robust joint chance constraints with second-order moment information. *Mathematical Programming*, 137(1-2):167–198, 2011.

Appendix A

Proofs

In this chapter, we give the complete proofs of Theorem 4.2, Lemma 4.6, Theorem 5.1, and Theorem 5.4.

A.1 Proof of Theorem 4.2

The Lagrange function of Problem (4.12) is expressed as

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{p}, \boldsymbol{q}, r, s; \boldsymbol{\alpha}, \boldsymbol{B}, \boldsymbol{\beta}, \boldsymbol{\eta}, \boldsymbol{\Lambda}, \boldsymbol{\lambda}, \nu, \pi, \boldsymbol{\rho}) \\
&:= \frac{1}{2\gamma}\boldsymbol{x}^\top \boldsymbol{x} + (\kappa_2 \hat{\boldsymbol{\Sigma}} - \hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^\top) \bullet \boldsymbol{Q} + r + \hat{\boldsymbol{\Sigma}} \bullet \boldsymbol{P} - 2\hat{\boldsymbol{\mu}}^\top \boldsymbol{p} + \kappa_1 s \\
&\quad - \boldsymbol{\alpha}^\top (\boldsymbol{p} + \boldsymbol{q}/2 + \boldsymbol{Q}\hat{\boldsymbol{\mu}}) \\
&\quad - \sum_{\ell \in [L]} \begin{pmatrix} \boldsymbol{B}^{(\ell)} & \boldsymbol{\beta}^{(\ell)} \\ (\boldsymbol{\beta}^{(\ell)})^\top & \eta^{(\ell)} \end{pmatrix} \bullet \begin{pmatrix} \boldsymbol{Q} & \boldsymbol{q}/2 + a^{(\ell)}\boldsymbol{Z}\boldsymbol{x}/2 \\ (\boldsymbol{q}/2 + a^{(\ell)}\boldsymbol{Z}\boldsymbol{x}/2)^\top & r + b^{(\ell)} \end{pmatrix} \\
&\quad - \begin{pmatrix} \boldsymbol{\Lambda} & \boldsymbol{\lambda} \\ \boldsymbol{\lambda}^\top & \nu \end{pmatrix} \bullet \begin{pmatrix} \boldsymbol{P} & \boldsymbol{p} \\ \boldsymbol{p}^\top & s \end{pmatrix} - \pi(\mathbf{1}^\top \boldsymbol{Z}\boldsymbol{x} - 1) - \boldsymbol{\rho}^\top \boldsymbol{Z}\boldsymbol{x},
\end{aligned}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^N$, $\begin{pmatrix} \boldsymbol{B}^{(\ell)} & \boldsymbol{\beta}^{(\ell)} \\ (\boldsymbol{\beta}^{(\ell)})^\top & \eta^{(\ell)} \end{pmatrix} \succeq \boldsymbol{O}$ $(\ell \in [L])$, $\begin{pmatrix} \boldsymbol{\Lambda} & \boldsymbol{\lambda} \\ \boldsymbol{\lambda}^\top & \nu \end{pmatrix} \succeq \boldsymbol{O}$, $\pi \in \mathbb{R}$, and $\boldsymbol{\rho} \geq \mathbf{0}$ are Lagrange multipliers. The Lagrange dual of Problem (4.12) is then posed as

$$\max_{\boldsymbol{\alpha}, \boldsymbol{B}, \boldsymbol{\beta}, \boldsymbol{\eta}, \boldsymbol{\Lambda}, \boldsymbol{\lambda}, \nu, \pi, \boldsymbol{\rho}} \min_{\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{p}, \boldsymbol{q}, r, s} \mathcal{L}(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{p}, \boldsymbol{q}, r, s; \boldsymbol{\alpha}, \boldsymbol{B}, \boldsymbol{\beta}, \boldsymbol{\eta}, \boldsymbol{\Lambda}, \boldsymbol{\lambda}, \nu, \pi, \boldsymbol{\rho}). \quad \text{(A.1)}$$

Now, let us focus on the inner minimization problem:

$$\min_{\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{p}, \boldsymbol{q}, r, s} \mathcal{L}(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{p}, \boldsymbol{q}, r, s; \boldsymbol{\alpha}, \boldsymbol{B}, \boldsymbol{\beta}, \boldsymbol{\eta}, \boldsymbol{\Lambda}, \boldsymbol{\lambda}, \nu, \pi, \boldsymbol{\rho}). \quad \text{(A.2)}$$

Note that Problem (A.2) is an unconstrained convex quadratic optimization problem and its objective function is linear in $(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{p}, \boldsymbol{q}, r, s)$. Since Problem (A.2) must be bounded, the Lagrange multipliers are required to satisfy the following

conditions:

$$\nabla_{\boldsymbol{P}}\mathcal{L} = \hat{\boldsymbol{\Sigma}} - \boldsymbol{\Lambda} = \boldsymbol{O}, \tag{A.3}$$

$$\nabla_{\boldsymbol{Q}}\mathcal{L} = \kappa_2\hat{\boldsymbol{\Sigma}} - \hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^\top - \frac{1}{2}(\hat{\boldsymbol{\mu}}\boldsymbol{\alpha}^\top + \boldsymbol{\alpha}\hat{\boldsymbol{\mu}}^\top) - \sum_{\ell\in[L]}\boldsymbol{B}^{(\ell)} = \boldsymbol{O}, \tag{A.4}$$

$$\nabla_{\boldsymbol{p}}\mathcal{L} = -2\hat{\boldsymbol{\mu}} - \boldsymbol{\alpha} - 2\boldsymbol{\lambda} = \boldsymbol{0}, \tag{A.5}$$

$$\nabla_{\boldsymbol{q}}\mathcal{L} = -\frac{1}{2}\boldsymbol{\alpha} - \sum_{\ell\in[L]}\boldsymbol{\beta}^{(\ell)} = \boldsymbol{0}, \tag{A.6}$$

$$\nabla_r\mathcal{L} = 1 - \sum_{\ell\in[L]}\eta^{(\ell)} = 0, \tag{A.7}$$

$$\nabla_s\mathcal{L} = \kappa_1 - \nu = 0. \tag{A.8}$$

Also, the following optimality condition should be satisfied:

$$\nabla_{\boldsymbol{x}}\mathcal{L} = \frac{1}{\gamma}\boldsymbol{x} - \boldsymbol{Z}\left(\sum_{\ell\in[L]}a^{(\ell)}\boldsymbol{\beta}^{(\ell)} + \pi\boldsymbol{1} + \boldsymbol{\rho}\right) = \boldsymbol{0}. \tag{A.9}$$

According to the conditions (A.3)–(A.9), the optimal objective value of Problem (A.2) is calculated as

$$-\frac{\gamma}{2}\boldsymbol{\omega}^\top\boldsymbol{Z}^2\boldsymbol{\omega} - \sum_{\ell\in[L]}\eta^{(\ell)}b^{(\ell)} + \pi, \tag{A.10}$$

where

$$\boldsymbol{\omega} = \sum_{\ell\in[L]}a^{(\ell)}\boldsymbol{\beta}^{(\ell)} + \pi\boldsymbol{1} + \boldsymbol{\rho}.$$

Since $\boldsymbol{z} \in \{0,1\}^N$, it holds that $\boldsymbol{\omega}^\top\boldsymbol{Z}^2\boldsymbol{\omega} = \boldsymbol{z}^\top(\boldsymbol{\omega}\circ\boldsymbol{\omega})$. Thus, the Lagrange dual (A.1) of Problem (4.12) is formulated as follows:

$$\underset{\boldsymbol{\omega},\boldsymbol{\alpha},\boldsymbol{B},\boldsymbol{\beta},\eta,\boldsymbol{\Lambda},\boldsymbol{\lambda},\nu,\pi,\boldsymbol{\rho}}{\text{maximize}} \quad -\frac{\gamma}{2}\boldsymbol{z}^\top(\boldsymbol{\omega}\circ\boldsymbol{\omega}) - \sum_{\ell\in[L]}\eta^{(\ell)}b^{(\ell)} + \pi \tag{A.11a}$$

$$\text{subject to} \quad \boldsymbol{\omega} = \sum_{\ell\in[L]}a^{(\ell)}\boldsymbol{\beta}^{(\ell)} + \pi\boldsymbol{1} + \boldsymbol{\rho}, \tag{A.11b}$$

$$\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Lambda} = \boldsymbol{O}, \tag{A.11c}$$

$$\kappa_2\hat{\boldsymbol{\Sigma}} - \hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^\top - \frac{1}{2}(\hat{\boldsymbol{\mu}}\boldsymbol{\alpha}^\top + \boldsymbol{\alpha}\hat{\boldsymbol{\mu}}^\top) - \sum_{\ell\in[L]}\boldsymbol{B}^{(\ell)} = \boldsymbol{O}, \tag{A.11d}$$

$$-2\hat{\boldsymbol{\mu}} - \boldsymbol{\alpha} - 2\boldsymbol{\lambda} = \boldsymbol{0}, \tag{A.11e}$$

$$-\frac{1}{2}\boldsymbol{\alpha} - \sum_{\ell\in[L]}\boldsymbol{\beta}^{(\ell)} = \boldsymbol{0}, \tag{A.11f}$$

$$1 - \sum_{\ell\in[L]}\eta^{(\ell)} = 0, \tag{A.11g}$$

$$\kappa_1 - \nu = 0, \tag{A.11h}$$

$$\begin{pmatrix}\boldsymbol{B}^{(\ell)} & \boldsymbol{\beta}^{(\ell)}\\ (\boldsymbol{\beta}^{(\ell)})^\top & \eta^{(\ell)}\end{pmatrix} \succeq \boldsymbol{O} \quad (\ell\in[L]), \tag{A.11i}$$

$$\begin{pmatrix}\boldsymbol{\Lambda} & \boldsymbol{\lambda}\\ \boldsymbol{\lambda}^\top & \nu\end{pmatrix} \succeq \boldsymbol{O}, \tag{A.11j}$$

$$\boldsymbol{\rho} \geq \boldsymbol{0}. \tag{A.11k}$$

We then delete $\boldsymbol{\rho}$, $\boldsymbol{\Lambda}\,\boldsymbol{\alpha}$, and ν from Problem (A.11) by substituting Eqs. (A.11b), (A.11c), (A.11f), and (A.11h) into other constraints. We now obtain the desired formulation (4.13).

To prove the strong duality, we next show that both the primal problem (4.12) and the dual problem (4.13) are strictly feasible [7]. Let us set $\boldsymbol{x} = \boldsymbol{z}/(\mathbf{1}^\top \boldsymbol{z})$, $\boldsymbol{q} = \mathbf{0}$, $r > -\min_{\ell\in[L]}\{b^{(\ell)}\}$, and $s > 0$ in the primal problem (4.12). Then, there exists $\boldsymbol{Q}$ satisfying

$$\begin{pmatrix} \boldsymbol{Q} & \boldsymbol{q}/2 + a^{(\ell)}\boldsymbol{Z}\boldsymbol{x}/2 \\ (\boldsymbol{q}/2 + a^{(\ell)}\boldsymbol{Z}\boldsymbol{x}/2)^\top & r + b^{(\ell)} \end{pmatrix} \succ \boldsymbol{O} \quad (\forall \ell \in [L]).$$

We next set $\boldsymbol{p} = -\boldsymbol{Q}\hat{\boldsymbol{\mu}}$, and then there exists $\boldsymbol{P}$ satisfying

$$\begin{pmatrix} \boldsymbol{P} & \boldsymbol{p} \\ \boldsymbol{p}^\top & s \end{pmatrix} \succ \boldsymbol{O},$$

which implies that the primal problem (4.12) is strictly feasible.

For the dual problem (4.13), we set $\boldsymbol{\lambda} = \mathbf{0}$ and $(\boldsymbol{B}, \boldsymbol{\beta}, \boldsymbol{\eta})$ as

$$\boldsymbol{B}^{(\ell)} = \frac{1}{L}(\kappa_2\hat{\boldsymbol{\Sigma}} + \hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^\top), \quad \boldsymbol{\beta}^{(\ell)} = \frac{1}{L}\hat{\boldsymbol{\mu}}, \quad \eta^{(\ell)} = \frac{1}{L} \quad (\forall \ell \in [L])$$

such that the constraints (4.13c)–(4.13e) are satisfied. We next set $\boldsymbol{\omega} = \sum_{\ell\in[L]} a^{(\ell)}\boldsymbol{\beta}^{(\ell)} + \pi\mathbf{1}$. As for the constraint (4.13f), we have

$$\begin{pmatrix} \boldsymbol{B}^{(\ell)} & \boldsymbol{\beta}^{(\ell)} \\ (\boldsymbol{\beta}^{(\ell)})^\top & \eta^{(\ell)} \end{pmatrix} = \frac{1}{L}\begin{pmatrix} \kappa_2\hat{\boldsymbol{\Sigma}} + \hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^\top & \hat{\boldsymbol{\mu}} \\ \hat{\boldsymbol{\mu}}^\top & 1 \end{pmatrix} \succ \boldsymbol{O} \quad (\forall \ell \in [L]), \tag{A.12}$$

because the Schur complement is given from Assumption 4.1 by

$$\frac{1}{L}\left(\kappa_2\hat{\boldsymbol{\Sigma}} + \hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^\top - \hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^\top\right) = \frac{\kappa_2}{L}\hat{\boldsymbol{\Sigma}} \succ \boldsymbol{O}. \tag{A.13}$$

Also for the constraint (4.13g), Assumption 4.1 ensures that

$$\begin{pmatrix} \hat{\boldsymbol{\Sigma}} & \boldsymbol{\lambda} \\ \boldsymbol{\lambda}^\top & \kappa_1 \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{\Sigma}} & \mathbf{0} \\ \mathbf{0} & \kappa_1 \end{pmatrix} \succ \boldsymbol{O}, \tag{A.14}$$

which shows that the dual problem (4.13) is strictly feasible.

A.2 Proof of Lemma 4.6

For notational simplicity, we assume $\eta^{(\ell)} > 0$ for all $\ell \in [L]$ without loss of generality. As in the proof of Theorem 4.8, we use the notations (4.29) and (4.30) under the assumption (4.28). We define $\boldsymbol{\phi}^{(\ell)} := \bar{\boldsymbol{\beta}}^{(\ell)} - \eta^{(\ell)}\hat{\boldsymbol{\mu}}$ for $\ell \in [L]$, and due to the definition (4.22), we have

$$\boldsymbol{\phi}_1^{(\ell)} = \boldsymbol{\beta}_1^{(\ell)} - \eta^{(\ell)}\hat{\boldsymbol{\mu}}_1, \tag{A.15}$$

$$\boldsymbol{\phi}_2^{(\ell)} = \left(\hat{\boldsymbol{\Sigma}}_{21}(\hat{\boldsymbol{\Sigma}}_{11})^{-1}(\boldsymbol{\beta}_1^{(\ell)} - \eta^{(\ell)}\hat{\boldsymbol{\mu}}_1) + \eta^{(\ell)}\hat{\boldsymbol{\mu}}_2\right) - \eta^{(\ell)}\hat{\boldsymbol{\mu}}_2 = \boldsymbol{\Psi}\boldsymbol{\phi}_1^{(\ell)}, \tag{A.16}$$

where

$$\boldsymbol{\Psi} := \hat{\boldsymbol{\Sigma}}_{21}(\hat{\boldsymbol{\Sigma}}_{11})^{-1}. \tag{A.17}$$

Substituting them into the right-hand side of Eq. (4.23), we obtain

$$\begin{aligned}
&\sum_{\ell\in[L]} \frac{1}{\eta^{(\ell)}} (\bar{\boldsymbol{\beta}}^{(\ell)} - \eta^{(\ell)}\hat{\boldsymbol{\mu}})(\bar{\boldsymbol{\beta}}^{(\ell)} - \eta^{(\ell)}\hat{\boldsymbol{\mu}})^\top \\
&= \sum_{\ell\in[L]} \frac{1}{\eta^{(\ell)}} \begin{pmatrix} \boldsymbol{\phi}_1^{(\ell)}(\boldsymbol{\phi}_1^{(\ell)})^\top & \boldsymbol{\phi}_1^{(\ell)}(\boldsymbol{\phi}_2^{(\ell)})^\top \\ \boldsymbol{\phi}_2^{(\ell)}(\boldsymbol{\phi}_1^{(\ell)})^\top & \boldsymbol{\phi}_2^{(\ell)}(\boldsymbol{\phi}_2^{(\ell)})^\top \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Phi} & \boldsymbol{\Phi}\boldsymbol{\Psi}^\top \\ \boldsymbol{\Psi}\boldsymbol{\Phi} & \boldsymbol{\Psi}\boldsymbol{\Phi}\boldsymbol{\Psi}^\top \end{pmatrix},
\end{aligned} \tag{A.18}$$

where

$$\boldsymbol{\Phi} := \sum_{\ell\in[L]} \frac{1}{\eta^{(\ell)}} \boldsymbol{\phi}_1^{(\ell)}(\boldsymbol{\phi}_1^{(\ell)})^\top. \tag{A.19}$$

From the constraint (4.21f), we can use the following Schur complement property (see, e.g., Section A.5.5 [36]):

$$\begin{pmatrix} \boldsymbol{B}_{11}^{(\ell)} & \boldsymbol{\beta}_1^{(\ell)} \\ \left(\boldsymbol{\beta}_1^{(\ell)}\right)^\top & \eta^{(\ell)} \end{pmatrix} \succeq \boldsymbol{O} \;\Rightarrow\; \boldsymbol{B}_{11}^{(\ell)} \succeq \frac{1}{\eta^{(\ell)}} \boldsymbol{\beta}_1^{(\ell)} \left(\boldsymbol{\beta}_1^{(\ell)}\right)^\top \quad (\forall \ell \in [L]). \tag{A.20}$$

It then follows that

$$\begin{aligned}
\boldsymbol{\Phi} &= \sum_{\ell\in[L]} \frac{1}{\eta^{(\ell)}} \boldsymbol{\beta}_1^{(\ell)}(\boldsymbol{\beta}_1^{(\ell)})^\top - \left(\sum_{\ell\in[L]} \boldsymbol{\beta}_1^{(\ell)}\right) \hat{\boldsymbol{\mu}}_1^\top - \hat{\boldsymbol{\mu}}_1 \left(\sum_{\ell\in[L]} \boldsymbol{\beta}_1^{(\ell)}\right)^\top + \hat{\boldsymbol{\mu}}_1\hat{\boldsymbol{\mu}}_1^\top \\
&\qquad \because \text{Eqs. (4.21e), (A.15), (A.19)} \\
&\preceq \sum_{\ell\in[L]} \boldsymbol{B}_{11}^{(\ell)} + \hat{\boldsymbol{\mu}}_1\hat{\boldsymbol{\mu}}_1^\top - \hat{\boldsymbol{\mu}}_1 \left(\sum_{\ell\in[L]} \boldsymbol{\beta}_1^{(\ell)}\right)^\top - \left(\sum_{\ell\in[L]} \boldsymbol{\beta}_1^{(\ell)}\right) \hat{\boldsymbol{\mu}}_1^\top \quad \because \text{Eq. (A.20)} \\
&= \kappa_2 \hat{\boldsymbol{\Sigma}}_{11}. \quad \because \text{Eq. (4.21c)}
\end{aligned} \tag{A.21}$$

From Assumption 4.1, we can also use the following Schur complement property (see, e.g., Section A.5.5 [36]):

$$\begin{pmatrix} \hat{\boldsymbol{\Sigma}}_{11} & \hat{\boldsymbol{\Sigma}}_{21}^\top \\ \hat{\boldsymbol{\Sigma}}_{21} & \hat{\boldsymbol{\Sigma}}_{22} \end{pmatrix} \succ \boldsymbol{O} \;\Rightarrow\; \hat{\boldsymbol{\Sigma}}_{22} - \hat{\boldsymbol{\Sigma}}_{21}(\hat{\boldsymbol{\Sigma}}_{11})^{-1}\hat{\boldsymbol{\Sigma}}_{21}^\top \succ \boldsymbol{O}. \tag{A.22}$$

We now obtain the desired result:

$$\begin{aligned}
\sum_{\ell\in[L]} \frac{1}{\eta^{(\ell)}} (\bar{\boldsymbol{\beta}}^{(\ell)} - \eta^{(\ell)}\hat{\boldsymbol{\mu}})(\bar{\boldsymbol{\beta}}^{(\ell)} - \eta^{(\ell)}\hat{\boldsymbol{\mu}})^\top &= \begin{pmatrix} \boldsymbol{\Phi} & \boldsymbol{\Phi}\boldsymbol{\Psi}^\top \\ \boldsymbol{\Psi}\boldsymbol{\Phi} & \boldsymbol{\Psi}\boldsymbol{\Phi}\boldsymbol{\Psi}^\top \end{pmatrix} \quad \because \text{Eq. (A.18)} \\
&\preceq \kappa_2 \begin{pmatrix} \hat{\boldsymbol{\Sigma}}_{11} & \hat{\boldsymbol{\Sigma}}_{21}^\top \\ \hat{\boldsymbol{\Sigma}}_{21} & \hat{\boldsymbol{\Sigma}}_{21}(\hat{\boldsymbol{\Sigma}}_{11})^{-1}\hat{\boldsymbol{\Sigma}}_{21}^\top \end{pmatrix} \quad \because \text{Eqs. (A.17), (A.21)} \\
&\preceq \kappa_2 \begin{pmatrix} \hat{\boldsymbol{\Sigma}}_{11} & \hat{\boldsymbol{\Sigma}}_{21}^\top \\ \hat{\boldsymbol{\Sigma}}_{21} & \hat{\boldsymbol{\Sigma}}_{22} \end{pmatrix}. \quad \because \text{Eq. (A.22)}
\end{aligned}$$

A.3 Proof of Theorem 5.1

Problem (5.7) is formulated as follows:

$$
\begin{aligned}
f(\boldsymbol{z}) = \underset{a,\boldsymbol{q},v,\boldsymbol{x}}{\text{minimize}} \quad & \frac{1}{2\gamma}\boldsymbol{x}^\top \boldsymbol{x} + a + v && \text{(A.23a)}\\
\text{subject to} \quad & v \geq \frac{1}{1-\beta}\sum_{s\in\mathcal{S}} p_s q_s, && \text{(A.23b)}\\
& q_s \geq -(\boldsymbol{r}^{(s)})^\top \boldsymbol{Z}\boldsymbol{x} - a \quad (\forall s \in \mathcal{S}), && \text{(A.23c)}\\
& \boldsymbol{C}\boldsymbol{Z}\boldsymbol{x} \leq \boldsymbol{d}, && \text{(A.23d)}\\
& \boldsymbol{1}^\top \boldsymbol{Z}\boldsymbol{x} = 1, && \text{(A.23e)}\\
& \boldsymbol{Z}\boldsymbol{x} \geq \boldsymbol{0}, && \text{(A.23f)}\\
& \boldsymbol{q} \geq \boldsymbol{0}. && \text{(A.23g)}
\end{aligned}
$$

The Lagrange function of Problem (A.23) is expressed as

$$
\begin{aligned}
\mathcal{L}(a,\boldsymbol{q},v,\boldsymbol{x};\eta,\boldsymbol{\alpha},\boldsymbol{\zeta},\lambda,\boldsymbol{\pi},\boldsymbol{\theta}) := \frac{1}{2\gamma}\boldsymbol{x}^\top\boldsymbol{x} + a + v - \eta\left(v - \frac{1}{1-\beta}\sum_{s\in\mathcal{S}} p_s q_s\right)\\
- \sum_{s\in\mathcal{S}} \alpha_s \left(q_s + (\boldsymbol{r}^{(s)})^\top \boldsymbol{Z}\boldsymbol{x} + a\right)\\
- \boldsymbol{\zeta}^\top(\boldsymbol{d} - \boldsymbol{C}\boldsymbol{Z}\boldsymbol{x}) - \lambda\left(\boldsymbol{1}^\top\boldsymbol{Z}\boldsymbol{x} - 1\right) - \boldsymbol{\pi}^\top\boldsymbol{Z}\boldsymbol{x} - \boldsymbol{\theta}^\top\boldsymbol{q},
\end{aligned}
$$

where $\eta \geq 0$, $\boldsymbol{\alpha} := (\alpha_s)_{s\in\mathcal{S}} \geq \boldsymbol{0}$, $\boldsymbol{\zeta} \geq \boldsymbol{0}$, $\lambda \in \mathbb{R}$, $\boldsymbol{\pi} \geq \boldsymbol{0}$ and $\boldsymbol{\theta} \geq \boldsymbol{0}$ are Lagrange multipliers. Then, the Lagrange dual problem of Problem (A.23) is posed as:

$$
\max_{\substack{\eta\geq 0,\ \boldsymbol{\alpha}\geq\boldsymbol{0},\ \boldsymbol{\zeta}\geq\boldsymbol{0},\\ \lambda\in\mathbb{R},\ \boldsymbol{\pi}\geq\boldsymbol{0},\ \boldsymbol{\theta}\geq\boldsymbol{0}}} \ \min_{\substack{a\in\mathbb{R},\ \boldsymbol{q}\in\mathbb{R}^S,\\ v\in\mathbb{R},\ \boldsymbol{x}\in\mathbb{R}^N}} \mathcal{L}(a,\boldsymbol{q},v,\boldsymbol{x};\eta,\boldsymbol{\alpha},\boldsymbol{\zeta},\lambda,\boldsymbol{\pi},\boldsymbol{\theta}). \tag{A.24}
$$

Recall that Problem (A.23) is feasible. Also, the objective function is proper convex, and all the constraints are linear in Problem (A.23). Then, the strong duality holds; see, for example, Section 5.2.3 in Boyd and Vandenberghe [36]. As a result, $f(\boldsymbol{z})$ is equal to the optimal objective value of Problem (A.24). Now, let us focus on the inner minimization problem:

$$
\min_{\substack{a\in\mathbb{R},\ \boldsymbol{q}\in\mathbb{R}^S,\\ v\in\mathbb{R},\ \boldsymbol{x}\in\mathbb{R}^N}} \mathcal{L}(a,\boldsymbol{q},v,\boldsymbol{x};\eta,\boldsymbol{\alpha},\boldsymbol{\zeta},\lambda,\boldsymbol{\pi},\boldsymbol{\theta}). \tag{A.25}
$$

Note that Problem (A.25) is an unconstrained convex quadratic optimization problem and its objective function is linear in $(a,\boldsymbol{q},v)$. Because Problem (A.25) must be bounded, the Lagrange multipliers are required to satisfy the following conditions:

$$
\begin{aligned}
\nabla_a \mathcal{L} &= 1 - \sum_{s\in\mathcal{S}} \alpha_s = 0, && \text{(A.26)}\\
\nabla_{\boldsymbol{q}} \mathcal{L} &= \frac{\eta}{1-\beta}\boldsymbol{p} - \boldsymbol{\alpha} - \boldsymbol{\theta} = 0, && \text{(A.27)}\\
\nabla_v \mathcal{L} &= 1 - \eta = 0. && \text{(A.28)}
\end{aligned}
$$

Also, the following optimality condition should be satisfied:

$$
\nabla_{\boldsymbol{x}} \mathcal{L} = \frac{1}{\gamma}\boldsymbol{x} - \boldsymbol{Z}\left(\sum_{s\in\mathcal{S}} \alpha_s \boldsymbol{r}^{(s)} - \boldsymbol{C}^\top\boldsymbol{\zeta} + \lambda\boldsymbol{1} + \boldsymbol{\pi}\right) = \boldsymbol{0}. \tag{A.29}
$$

According to Eqs. (A.26), (A.27), (A.28), and (A.29), the optimal objective value of Problem (A.25) is calculated as

$$-\frac{\gamma}{2}\boldsymbol{\omega}^\top \boldsymbol{Z}^2\boldsymbol{\omega} - \boldsymbol{d}^\top\boldsymbol{\zeta} + \lambda,$$

where $\boldsymbol{\omega} \in \mathbb{R}^N$ is a vector of auxiliary decision variables satisfying

$$\boldsymbol{\omega} = \sum_{s\in\mathcal{S}} \alpha_s \boldsymbol{r}^{(s)} - \boldsymbol{C}^\top\boldsymbol{\zeta} + \lambda\mathbf{1} + \boldsymbol{\pi}.$$

Because $\boldsymbol{z} \in \{0,1\}^N$, it holds that $\boldsymbol{\omega}^\top\boldsymbol{Z}^2\boldsymbol{\omega} = \boldsymbol{z}^\top(\boldsymbol{\omega}\circ\boldsymbol{\omega})$. Therefore, the Lagrange dual problem (A.24) is formulated as follows:

$$\begin{aligned} f(\boldsymbol{z}) = \underset{\boldsymbol{\alpha},\boldsymbol{\zeta},\lambda,\boldsymbol{\omega}}{\text{maximize}} \quad & -\frac{\gamma}{2}\boldsymbol{z}^\top(\boldsymbol{\omega}\circ\boldsymbol{\omega}) - \boldsymbol{d}^\top\boldsymbol{\zeta} + \lambda \\ \text{subject to} \quad & \boldsymbol{\omega} \geq \sum_{s\in\mathcal{S}} \alpha_s \boldsymbol{r}^{(s)} - \boldsymbol{C}^\top\boldsymbol{\zeta} + \lambda\mathbf{1}, \\ & \sum_{s\in\mathcal{S}} \alpha_s = 1, \\ & \alpha_s \leq \frac{p_s}{1-\beta} \quad (\forall s \in \mathcal{S}), \\ & \boldsymbol{\alpha} \geq \mathbf{0},\ \boldsymbol{\zeta} \geq \mathbf{0}, \end{aligned}$$

where we substitute $\eta = 1$, and eliminate the nonnegative variables $\boldsymbol{\pi}$ and $\boldsymbol{\theta}$. □

A.4 Proof of Theorem 5.4

Problem (5.23) is formulated as follows:

$$\begin{aligned} f_{\mathcal{K}}(\boldsymbol{z}) = \underset{a,v,\boldsymbol{x}}{\text{minimize}} \quad & \frac{1}{2\gamma}\boldsymbol{x}^\top\boldsymbol{x} + a + v && \text{(A.30a)} \\ \text{subject to} \quad & v \geq \frac{1}{1-\beta}\sum_{s\in\mathcal{J}} p_s(-(\boldsymbol{r}^{(s)})^\top\boldsymbol{Z}\boldsymbol{x} - a) \quad (\forall \mathcal{J} \in \mathcal{K}), && \text{(A.30b)} \\ & v \geq 0, && \text{(A.30c)} \\ & \boldsymbol{C}\boldsymbol{Z}\boldsymbol{x} \leq \boldsymbol{d}, && \text{(A.30d)} \\ & \mathbf{1}^\top\boldsymbol{Z}\boldsymbol{x} = 1, && \text{(A.30e)} \\ & \boldsymbol{Z}\boldsymbol{x} \geq \mathbf{0}. && \text{(A.30f)} \end{aligned}$$

The Lagrange function of Problem (A.30) is expressed as

$$\begin{aligned} \mathcal{L}(a,v,\boldsymbol{x};\boldsymbol{\alpha},\xi,\boldsymbol{\zeta},\lambda,\boldsymbol{\pi}) := \frac{1}{2\gamma}\boldsymbol{x}^\top\boldsymbol{x} + a + v - \sum_{\mathcal{J}\in\mathcal{K}} \alpha_{\mathcal{J}} \left(v + \frac{1}{1-\beta}\sum_{s\in\mathcal{J}} p_s \left((\boldsymbol{r}^{(s)})^\top\boldsymbol{Z}\boldsymbol{x} + a \right) \right) \\ - \xi v - \boldsymbol{\zeta}^\top(\boldsymbol{d} - \boldsymbol{C}\boldsymbol{Z}\boldsymbol{x}) - \lambda\left(\mathbf{1}^\top\boldsymbol{Z}\boldsymbol{x} - 1\right) - \boldsymbol{\pi}^\top\boldsymbol{Z}\boldsymbol{x}, \end{aligned}$$

where $\boldsymbol{\alpha} := (\alpha_{\mathcal{J}})_{\mathcal{J}\in\mathcal{K}} \geq \mathbf{0}$, $\xi \geq 0$, $\boldsymbol{\zeta} \geq \mathbf{0}$, $\lambda \in \mathbb{R}$ and $\boldsymbol{\pi} \geq \mathbf{0}$ are Lagrange multipliers. Recall that Problem (A.30) is feasible. Then, the strong duality holds as well as the proof of Theorem 5.1, and $f_{\mathcal{K}}(\boldsymbol{z})$ is equal to the optimal objective value of the following Lagrange dual problem:

$$\max_{\substack{\boldsymbol{\alpha}\geq\mathbf{0},\ \xi\geq 0,\ \boldsymbol{\zeta}\geq\mathbf{0},\\ \lambda\in\mathbb{R},\ \boldsymbol{\pi}\geq\mathbf{0}}} \quad \min_{a\in\mathbb{R},\ v\in\mathbb{R},\ \boldsymbol{x}\in\mathbb{R}^N} \quad \mathcal{L}(a,v,\boldsymbol{x};\boldsymbol{\alpha},\xi,\boldsymbol{\zeta},\lambda,\boldsymbol{\pi}). \qquad \text{(A.31)}$$

Now, let us consider the inner minimization problem:

$$\min_{a\in\mathbb{R},\ v\in\mathbb{R},\ \boldsymbol{x}\in\mathbb{R}^N} \quad \mathcal{L}(a, v, \boldsymbol{x}; \boldsymbol{\alpha}, \xi, \boldsymbol{\zeta}, \lambda, \boldsymbol{\pi}). \tag{A.32}$$

Similarly to the proof of Theorem 5.1, the following conditions must be satisfied:

$$\nabla_a \mathcal{L} = 1 - \frac{1}{1-\beta} \sum_{\mathcal{J}\in\mathcal{K}} \alpha_{\mathcal{J}} \sum_{s\in\mathcal{J}} p_s = 0, \tag{A.33}$$

$$\nabla_v \mathcal{L} = 1 - \sum_{\mathcal{J}\in\mathcal{K}} \alpha_{\mathcal{J}} - \xi = 0, \tag{A.34}$$

$$\nabla_{\boldsymbol{x}} \mathcal{L} = \frac{1}{\gamma}\boldsymbol{x} - \boldsymbol{Z}\left(\frac{1}{1-\beta} \sum_{\mathcal{J}\in\mathcal{K}} \alpha_{\mathcal{J}} \sum_{s\in\mathcal{J}} p_s \boldsymbol{r}^{(s)} - \boldsymbol{C}^\top \boldsymbol{\zeta} + \lambda \mathbf{1} + \boldsymbol{\pi}\right) = \mathbf{0}. \tag{A.35}$$

According to Eqs. (A.33), (A.34), and (A.35), the optimal objective value of Problem (A.32) is calculated as

$$-\frac{\gamma}{2}\boldsymbol{\omega}^\top \boldsymbol{Z}^2 \boldsymbol{\omega} - \boldsymbol{d}^\top \boldsymbol{\zeta} + \lambda,$$

where $\boldsymbol{\omega} \in \mathbb{R}^N$ is a vector of auxiliary decision variables satisfying

$$\boldsymbol{\omega} = \frac{1}{1-\beta} \sum_{\mathcal{J}\in\mathcal{K}} \alpha_{\mathcal{J}} \sum_{s\in\mathcal{J}} p_s \boldsymbol{r}^{(s)} - \boldsymbol{C}^\top \boldsymbol{\zeta} + \lambda \mathbf{1} + \boldsymbol{\pi}.$$

Because $\boldsymbol{z} \in \{0,1\}^N$, it holds that $\boldsymbol{\omega}^\top \boldsymbol{Z}^2 \boldsymbol{\omega} = \boldsymbol{z}^\top (\boldsymbol{\omega} \circ \boldsymbol{\omega})$. Therefore, the Lagrange dual problem (A.31) is formulated as follows:

$$\begin{aligned} f_{\mathcal{K}}(\boldsymbol{z}) = \underset{\boldsymbol{\alpha},\boldsymbol{\zeta},\lambda,\boldsymbol{\omega}}{\text{maximize}} \quad & -\frac{\gamma}{2}\boldsymbol{z}^\top(\boldsymbol{\omega}\circ\boldsymbol{\omega}) - \boldsymbol{d}^\top\boldsymbol{\zeta} + \lambda \\ \text{subject to} \quad & \boldsymbol{\omega} \geq \frac{1}{1-\beta} \sum_{\mathcal{J}\in\mathcal{K}} \alpha_{\mathcal{J}} \sum_{s\in\mathcal{J}} p_s \boldsymbol{r}^{(s)} - \boldsymbol{C}^\top\boldsymbol{\zeta} + \lambda\mathbf{1}, \\ & \sum_{\mathcal{J}\in\mathcal{K}} \alpha_{\mathcal{J}} \leq 1, \\ & \sum_{\mathcal{J}\in\mathcal{K}} \alpha_{\mathcal{J}} \sum_{s\in\mathcal{J}} p_s = 1-\beta, \\ & \boldsymbol{\alpha} \geq \mathbf{0},\ \boldsymbol{\zeta} \geq \mathbf{0}, \end{aligned}$$

where we eliminate the nonnegative variables ξ and $\boldsymbol{\pi}$. □

Appendix B

Detailed Experimental Results

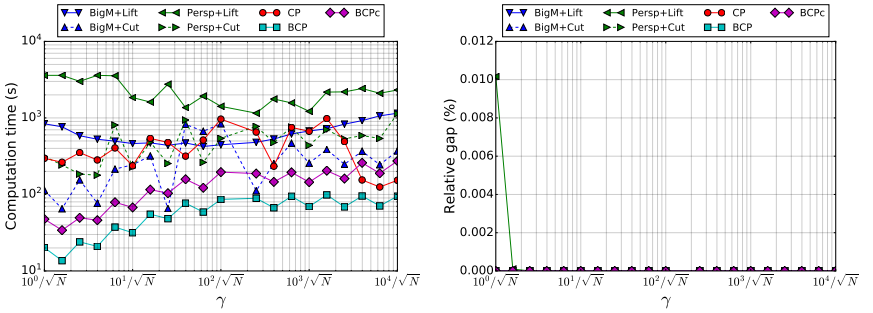
In this chapter, we show detailed experimental results for sensitivity analysis reported in Section 5.4.5. Table B.1 gives the numerical results for the four datasets (`ind49`, `sbm100`, `port1`, and `port5`) with the cardinality parameter $k \in \{5, 10, 15\}$. Here, we set $\gamma = 10/\sqrt{N}$ for the ℓ_2-regularization term and $S = 10^5$ as the number of scenarios. Table B.1 shows that BCP and BCPc were almost always faster than the other methods when $k \in \{10, 15\}$. Also, BCP and BCPc tended to be faster with larger k. In the case of the `port5` dataset, for example, BCP solved the problem with $k = 15$ in 73.7 s, whereas it failed to complete the computation within 3600 s with $k = 5$.

Figures B.1–B.4 show the numerical results for the four datasets (`ind49`, `sbm100`, `port1`, and `port5`) with the regularization parameter $\gamma \in \{10^{0.2i}/\sqrt{N} \mid i = 0, 1, \ldots, 20\}$. Here, we set $k = 10$ for the cardinality constraint and $S = 10^5$ as the number of scenarios. We can see from Figures B.1–B.4 that the performances of BCP and BCPc were not greatly affected by γ, and they were generally faster than the other methods when they completed the computations within 3600 s. In addition, BCP and BCPc tended to be faster with smaller γ for each dataset.

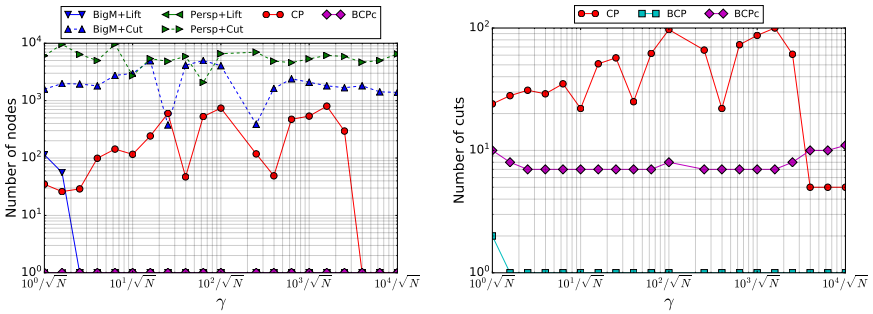
Figure B.5 shows "Reg+CVaR" and "CVaR" defined in Eq. (5.31) for the three datasets (`ind49`, `sbm100`, and `port1`) with the regularization parameter $\gamma \in \{10^{0.2i}/\sqrt{N} \mid i = 0, 1, \ldots, 20\}$. For each dataset, the objective value (i.e., "Reg+CVaR") was close to CVaR when $\gamma \geq 10^2/\sqrt{N}$, and CVaR almost converged to its minimum value when $\gamma \geq 10/\sqrt{N}$.

Table B.1: Numerical results for the four datasets (`port1`, `ind49`, `sbm100`, and `port5`) with $(S, \gamma) = (10^5, 10/\sqrt{N})$ for $k \in \{5, 10, 15\}$

Data	N	k		BigM		Persp		CP	BCP	BCPc
				Lift	Cut	Lift	Cut			
`port1`	30	5	Time	643.6	221.0	1496.1	**200.0**	644.0	279.6	319.8
			Obj	4.436	4.436	4.436	4.436	4.436	4.436	4.436
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	32	2636	13	3542	445	—	76
			#Cuts	—	—	—	—	105	59	62
		10	Time	468.6	163.3	1933.2	**15.8**	265.8	39.5	64.6
			Obj	4.264	4.264	4.264	4.264	4.264	4.264	4.264
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	1	2433	1	253	39	—	0
			#Cuts	—	—	—	—	25	1	7
		15	Time	469.4	120.1	3220.8	217.1	157.6	**39.1**	89.8
			Obj	4.222	4.222	4.222	4.222	4.222	4.222	4.222
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	1	879	1	4290	3	—	0
			#Cuts	—	—	—	—	8	1	7
`ind49`	49	5	Time	733.7	**160.0**	>3600	434.3	1455.9	638.5	907.5
			Obj	3.443	3.443	3.443	3.443	3.443	3.443	3.443
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	83	2475	>61	5172	2614	—	970
			#Cuts	—	—	—	—	226	132	172
		10	Time	709.8	125.0	>3600	220.0	662.2	**43.1**	103.6
			Obj	3.379	3.379	3.380	3.379	3.379	3.379	3.379
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	31	4101	>1	3489	250	—	0
			#Cuts	—	—	—	—	68	1	12
		15	Time	555.2	128.2	>3600	76.7	448.6	**48.7**	82.3
			Obj	3.366	3.366	3.366	3.366	3.366	3.366	3.366
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	1	3392	>1	1043	67	—	0
			#Cuts	—	—	—	—	39	1	7
`sbm100`	100	5	Time	1260.7	179.8	2941.5	147.8	690.8	**38.3**	48.7
			Obj	4.367	4.367	4.367	4.367	4.367	4.367	4.367
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	92	1092	3	1866	116	—	7
			#Cuts	—	—	—	—	53	5	12
		10	Time	1107.1	65.1	>3600	84.6	482.7	**27.6**	36.6
			Obj	4.364	4.364	4.364	4.364	4.364	4.364	4.364
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	1	412	>1	4676	10	—	0
			#Cuts	—	—	—	—	17	1	7
		15	Time	1051.6	43.2	3438.1	44.4	437.5	**29.5**	48.0
			Obj	4.364	4.364	4.364	4.364	4.364	4.364	4.364
			Gap(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			#Nodes	1	90	1	5738	9	—	0
			#Cuts	—	—	—	—	12	1	7
`port5`	225	5	Time	>3600	>3600	>3600	**2263.9**	>3600	>3600	>3600
			Obj	∞	3.491	3.755	**3.295**	3.397	**3.295**	**3.295**
			Gap(%)	100.00	8.72	14.25	**0.00**	8.52	0.72	3.37
			#Nodes	>0	>41,433	>42	19,851	>1390	—	>10,767
			#Cuts	—	—	—	—	>249	>667	>861
		10	Time	>3600	>3600	>3600	>3600	>3600	**2351.6**	3300.8
			Obj	∞	3.246	3.143	3.276	3.173	**3.138**	**3.138**
			Gap(%)	100.00	3.98	0.44	4.29	2.05	**0.00**	**0.00**
			#Nodes	>0	>20,833	>61	>16,790	>2140	—	5697
			#Cuts	—	—	—	—	>219	352	519
		15	Time	>3600	1154.4	>3600	1116.1	>3600	**73.7**	121.4
			Obj	∞	**3.109**	**3.109**	**3.109**	3.170	**3.109**	**3.109**
			Gap(%)	100.00	**0.00**	0.01	**0.00**	1.97	**0.00**	**0.00**
			#Nodes	>0	8454	>1	5696	>2516	—	0
			#Cuts	—	—	—	—	>218	1	11

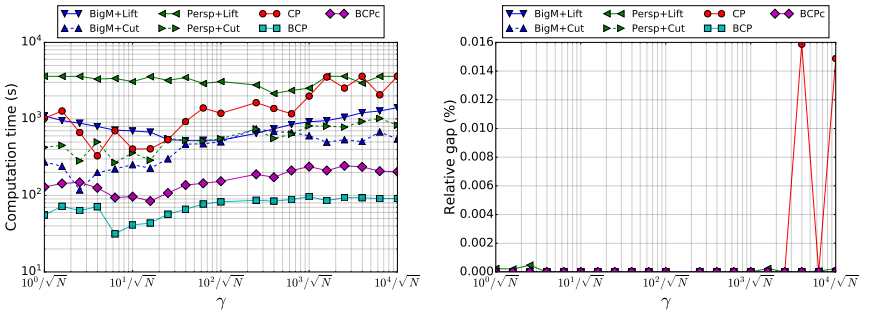


(a) Time (b) Gap(%)

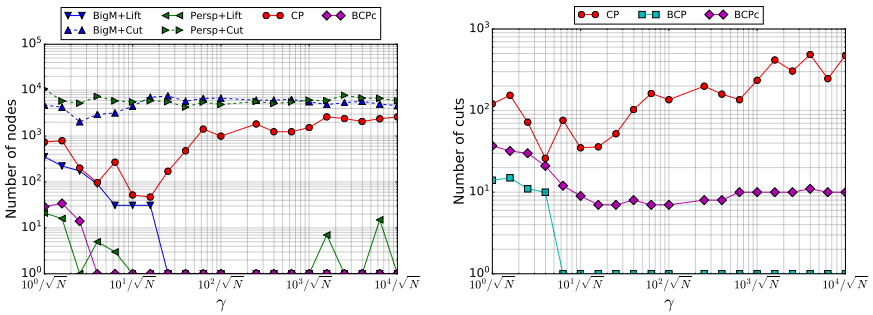


(c) #Nodes (d) #Cuts

Figure B.1: Results for `port1` with $(S, k) = (10^5, 10)$ for $\gamma \in \{10^{0.2i}/\sqrt{N} \mid i = 0, 1, \ldots, 20\}$

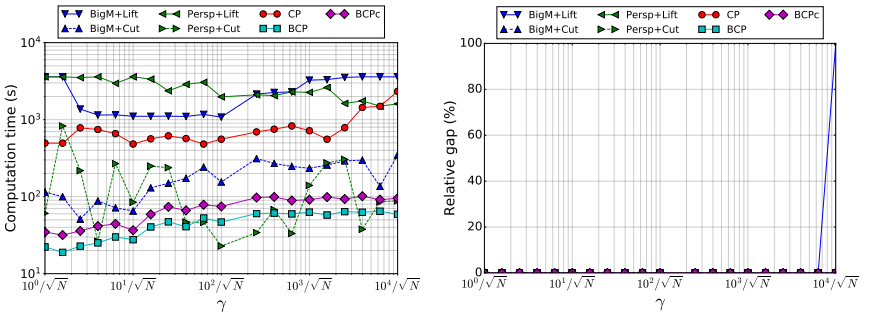


(a) Time (b) Gap(%)

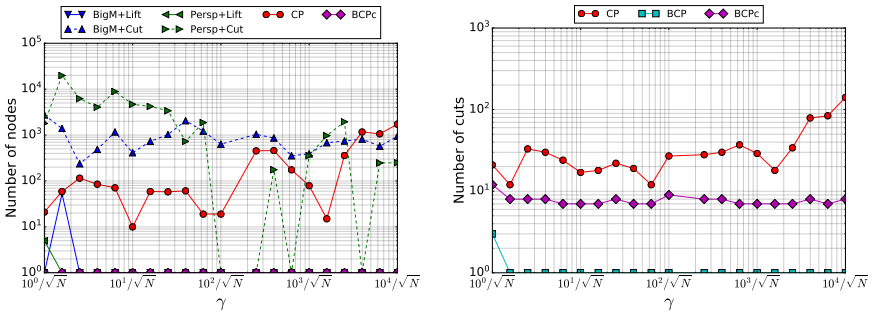


(c) #Nodes (d) #Cuts

Figure B.2: Results for `ind49` with $(S, k) = (10^5, 10)$ for $\gamma \in \{10^{0.2i}/\sqrt{N} \mid i = 0, 1, \ldots, 20\}$

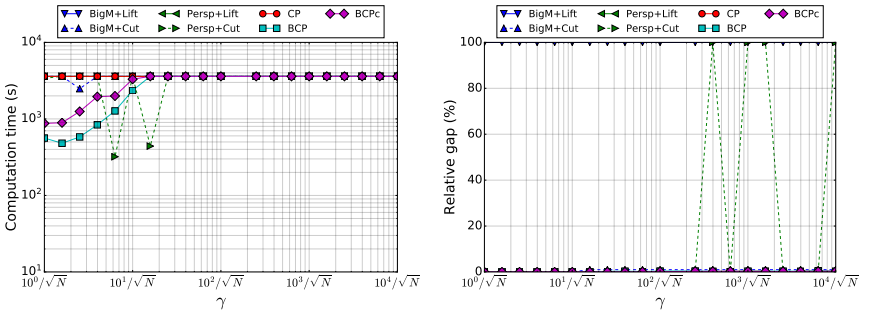


(a) Time (b) Gap(%)

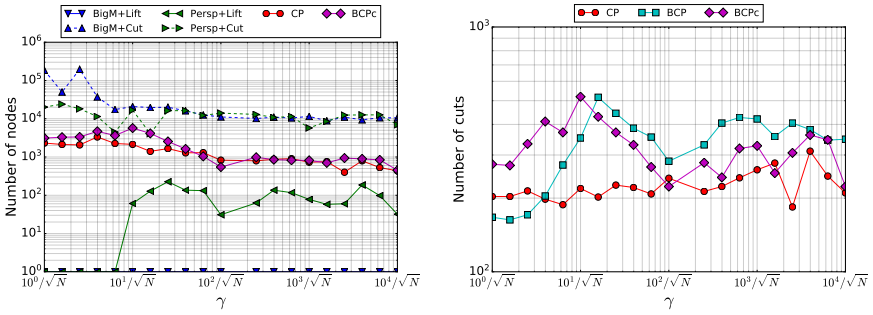


(c) #Nodes (d) #Cuts

Figure B.3: Results for `sbm100` with $(S, k) = (10^5, 10)$ for $\gamma \in \{10^{0.2i}/\sqrt{N} \mid i = 0, 1, \ldots, 20\}$

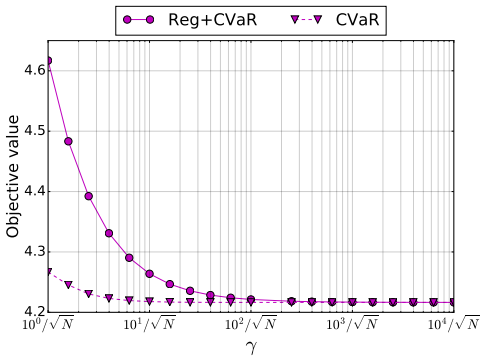


(a) Time (b) Gap(%)

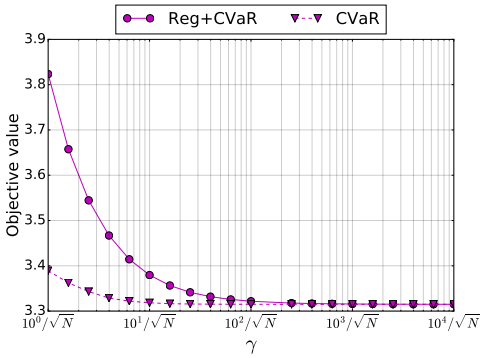


(c) #Nodes (d) #Cuts

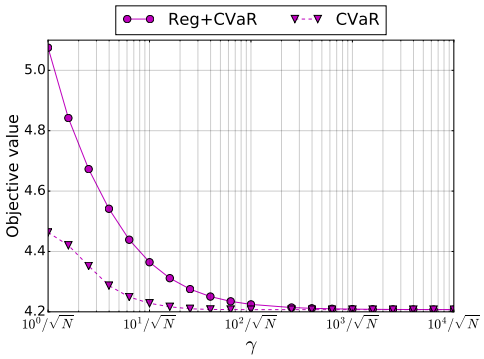
Figure B.4: Results for `port5` with $(S, k) = (10^5, 10)$ for $\gamma \in \{10^{0.2i}/\sqrt{N} \mid i = 0, 1, \ldots, 20\}$



(a) `port1`



(b) `ind49`



(c) `sbm100`

Figure B.5: Objective value (5.31) for the three datasets (`port1`, `ind49`, and `sbm100`) with $(S, k) = (10^5, 10)$ for $\gamma \in \{10^{0.2i}/\sqrt{N} \mid i = 0, 1, \ldots, 20\}$