2018

Exploiting Structures in Mixed-Integer Second-Order Cone Optimization Problems for Branch-and-Conic-Cut Algorithms

Sertalp Bilal Cay
Lehigh University

Exploiting Structures in Mixed-Integer Second-Order Cone Optimization Problems for Branch-and-Conic-Cut Algorithms

by

Sertalp B. Çay

A Dissertation
Presented to the Graduate Committee
of Lehigh University
in Candidacy for the Degree of
Doctor of Philosophy
in
Industrial and Systems Engineering

Lehigh University
May, 2018

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Sertalp B. Çay

Exploiting Structures in Mixed-Integer Second-Order Cone Optimization Problems for Branch-and-Conic-Cut Algorithms

---

Date

Dr. Tamás Terlaky, Dissertation Director, Chair

Accepted Date

Committee Members

Dr. Julio C. Góez

Dr. Imre Pólik

Dr. Ted K. Ralphs

Dr. Martin Takáč

Dr. Natasha Vermaak

Acknowledgment

First and foremost I would like to thank my advisor Tamás Terlaky for guiding me through this entire process. I wish to express my sincere gratitude for his encouragement, inspiration, and support throughout my research.

I would like to thank each of my committee members, Julio C. Góez, Imre Pólik, Ted K. Ralphs, Martin Takáč, and Natasha Vermaak, for their enthusiasm and insightful comments to my dissertation. I am sincerely grateful to Imre Pólik, who provided invaluable insights for my work on warm-start and heuristics for mixed-integer second-order cone optimization problems.

This work would not have been possible without the support and opportunity I received from SAS Institute. I would like to express my gratitude to Joshua Griffin, Imre Pólik, Yan Xu, and Manoj Chari for providing me summer and year-round internships during my Ph.D. studies. The experience I gained at SAS has greatly benefited the development of this research.

I am also grateful to many people around me. I would like to thank Frank E. Curtis for his support and encouragement during though times. I owe a great deal of gratitude to my officemates, fellow graduate students and good friends for their emotional support. I thank the Graduate Programs Manager, Brianne Lisk, for her help during my Ph.D.

I owe a lot of gratitude to my parents, Mustafa Çay and Fatma Çay, my in-laws H. Barış Diren and Ceyla Diren, and my sister Mehtap Hilal Çabuk. I am grateful for their love and support. I am also grateful to my dog, Latte, for letting off my

steam with countless hours of fetch during last two years of my Ph.D.

Most importantly, my heartfelt and deepest gratitude and thanks go to my dear wife, Pelin Çay, for her endless love, motivation, support, encouragement and patience. It is fair to say that without her priceless encouragement and support, this dissertation would not have been possible.

Contents

List of Tables

List of Figures

Abbreviations

B&B	branch-and-bound
B&C	branch-and-cut
BCC	branch-and-conic-cut
DCC	disjunctive conic cut
DCyC	disjunctive cylindrical cut
DR	dual rounding problem
FR	fix-and-relax problem
IPM	interior-point method
LO	linear optimization
MILO	mixed-integer linear optimization
MINLO	mixed-integer nonlinear optimization
MISOCO	mixed-integer second-order cone optimization
NLO	nonlinear optimization
PDIPM	primal and dual interior-point method
PR	primal rounding problem
SOC	second-order cone
SOCO	second-order cone optimization

Notation and symbols

$A, B, \ldots$	matrices
$a, b, \ldots$	column vectors
$x, y, \ldots$	column vectors, variables
$\mathrm{d}, \mathrm{r}, \mathrm{z}, \ldots$	scalar values
i, j	indices
x_i	the i^{th} component of x
$x^{(i)}$	the i^{th} iteration of an algorithm
x^*	an optimal solution
$\alpha, \beta, \theta, \ldots$	algorithm parameters
$\kappa, \lambda, \mu, \nu$	auxiliary variables
$\mathrm{P}, \mathrm{D}, \ldots$	optimization problems
$\mathcal{F}, \mathcal{S}, \ldots$	sets
$\mathfrak{A}, \mathfrak{B}, \ldots$	measures
$\mathcal{B}, \mathcal{N}, \mathcal{R}, \mathcal{T}$	optimal partitions of a SOCO problem
$\mathbb{R}^n$	the real n-dimensional Euclidean vector space
$\mathbb{Z}^n$	the set of integers in $\mathbb{R}^n$
$\mathbb{L}^n$	the n-dimensional Lorentz cone, $\mathbb{L} = \{x \in \mathbb{R}^n : x_1 \geq \|x_{2:n}\|_2\}$
$\mathcal{K}$	the Cartesian product of Lorentz cones
$\mathcal{K}^*$	the dual cone of $\mathcal{K}$
$\mathrm{diag}(x)$	the matrix that has x as its diagonal
$\mathrm{conv}(\mathcal{C})$	the convex hull of a set $\mathcal{C}$
$x \circ y$	the Jordan product of vectors x and y
$\lfloor x \rfloor$	the largest integer less than or equal to x
$\lceil x \rceil$	the smallest integer greater than or equal to x

Abstract

This thesis studies computational approaches for mixed-integer second-order cone optimization (MISOCO) problems. MISOCO models appear in many real-world applications, so MISOCO has gained significant interest in recent years. However, despite recent advancements, there is a gap between the theoretical developments and computational practice. Three chapters of this thesis address three areas of computational methodology for an efficient branch-and-conic-cut (BCC) algorithm to solve MISOCO problems faster in practice. These chapters include a detailed discussion on practical work on adding cuts in a BCC algorithm, novel methodologies for warm-starting second-order cone optimization (SOCO) subproblems, and heuristics for MISOCO problems.

The first part of this thesis concerns the development of a novel warm-starting method of interior-point methods (IPM) for SOCO problems. The method exploits the Jordan frames of an original instance and solves two auxiliary linear optimization problems. The solutions obtained from these problems are used to identify an ideal initial point of the IPM. Numerical results on public test sets indicate that the warm-start method works well in practice and reduces the number of iterations required to solve related SOCO problems by around 30–40%.

The second part of this thesis presents novel heuristics for MISOCO problems. These heuristics use the Jordan frames from both continuous relaxations and penalty problems and present a way of finding feasible solutions for MISOCO problems. Numerical results on conic and quadratic test sets show significant

performance in terms of finding a solution that has a small gap to optimality.

The last part of this thesis presents application of disjunctive conic cuts (DCC) and disjunctive cylindrical cuts (DCyC) to asset allocation problems (AAP). To maximize the benefit from these powerful cuts, several decisions regarding the addition of these cuts are inspected in a practical setting. The analysis in this chapter gives insight about how these cuts can be added in case-specific settings.

Chapter 1

Introduction

MISOCO problems consist of linear and second-order cone constraints with mixed-integer variables. The aim of this dissertation is to study MISOCO problems from a BCC algorithm perspective. We are interested in developing the computational methodology for an efficient BCC algorithm for MISOCO, developing novel methodologies, and combining the existing literature and establishing connections among them.

There is a growing interest to solve MISOCO problems, not only because there are efficient methods to solve the underlying SOCO problems, but also because they appear in many applications from finance to healthcare and facility location assignment [5, 13, 32, 62, 88].

MISOCO is a generalization of mixed-integer linear optimization (MILO) and is a specific area of mixed-integer nonlinear optimization (MINLO). Despite the large number of studies for MILO and MINLO, only a small portion of the available literature specifically focuses on MISOCO problems. Recent developments in the MISOCO area are centered around generating valid inequalities for these problems.

Despite the growing interest and the recent literature, there are only a limited number of research studies that focus on solution algorithms for MISOCO as a whole. In her thesis, Drewes [41] discusses many aspects of a branch-and-cut algorithm for

MISOCO, and combines some of the existing methodologies in the literature. In their respective dissertations, Narayanan [84] and Góez [54] provide methodologies to be used in a branch-and-cut framework, but to the best of our knowledge there is no work in the literature on practical implementations of these approaches. The purpose of this dissertation is to experiment with existing approaches and propose novel methodologies to contribute to building a full BCC framework.

1.1 Second-order cone optimization (SOCO)

We give a brief review on SOCO in this section. One needs to solve the SOCO relaxations of MISOCO many times in a branch-and-bound type algorithm to solve MISOCO problems. The aim of a SOCO model is to minimize a linear objective function over a set of linear and second-order cone constraints. Second-order cone constraints are also known as *Lorentz* cones. A second-order cone (SOC) represented as $(x, y) \in \mathbb{L} \subset \mathbb{R}^n$ corresponds to

$$x \geq \sqrt{\sum_{i=1}^{n-1} y_i^2},$$

where $x \in \mathbb{R}, y \in \mathbb{R}^{n-1}$.

SOCO problems are an important subclass of convex nonlinear optimization problems. SOCO has numerous applications [19, 72], and powerful solvers [7, 46, 61, 64] have been developed in the last 20 years.

Consider a primal SOCO problem in the standard form

$$\begin{array}{rlcl} \text{minimize:} & c^\top x & & \\ \text{subject to:} & Ax & = & b, \\ & x & \in & \mathcal{K}, \\ & x & \in & \mathbb{R}^n, \end{array} \tag{P-SOCO}$$

where $A \in \mathbb{R}^{m \times n}$ is a matrix with full row rank, and $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ are column vectors. Here, $\mathcal{K}$ is a Cartesian product of SOCs of various dimensions. For

$x = \left((x^1)^\top, (x^2)^\top, \ldots, (x^k)^\top\right)^\top$, $x^i \in \mathbb{L}^{n_i}$, it is shown as $\mathcal{K} = \mathbb{L}^{n_1} \times \mathbb{L}^{n_2} \times \cdots \times \mathbb{L}^{n_k}$, $\mathbb{L}^{n_i} = \{x^i \in \mathbb{R}^{n_i} \mid x_1^i \geq \|x_{2:n_i}^i\|\}$, for $i = 1, \ldots, k$, with $\sum_{i=1}^k n_i = n$. The cone $\mathcal{K}$ is self dual [85]; that is, $K^* = \left\{u | u^\top x \geq 0 \; \forall x \in \mathcal{K}\right\} = \mathcal{K}$.

The associated dual problem of (P-SOCO) is written as

$$\begin{array}{lrcl} \text{maximize:} & b^\top y & & \\ \text{subject to:} & A^\top y + z & = & c, \\ & z & \in & \mathcal{K}, \end{array} \qquad \text{(D-SOCO)}$$

where $z = ((z^1)^\top, (z^2)^\top, \ldots, (z^k)^\top)^\top$, $z \in \mathbb{R}^n$, $z^i \in \mathbb{R}^{n_i}$ for all i, and $y \in \mathbb{R}^m$.

Benson and Sağlam [20] gives a review of solution algorithms for SOCO. They identify three methodologies to solve SOCO problems in the literature. The first method is to use IPMs and solve SOCO as a conic problem. The second method is to solve SOCO problems by using IPMs for NLO. The third approach is to use a polyhedral relaxation of SOCO problems and solve them approximately as LO problems. In this thesis, our focus is on solving SOCO by IPMs as a conic problem. See Section 2.1 for details on this solution algorithm.

1.1.1 Jordan algebra

Relevant concepts of Jordan algebra are reviewed in this subsection [45].

A nonzero vector $x \in \mathbb{L}^n$ can be decomposed into two components, such as

$$x = f^+\lambda^+ + f^-\lambda^-$$

where

$$\lambda^+ = x_1 + \|x_{2:n}\|, \qquad \lambda^- = x_1 - \|x_{2:n}\|,$$
$$f^+ = \frac{1}{2}\begin{pmatrix} 1 \\ \frac{x_{2:n}}{\|x_{2:n}\|} \end{pmatrix}, \qquad f^- = \frac{1}{2}\begin{pmatrix} 1 \\ -\frac{x_{2:n}}{\|x_{2:n}\|} \end{pmatrix}.$$

In this system, λ^+ and λ^- are called the Jordan values, and f^+ and f^- are called the Jordan frames of the vector x. They correspond to eigenvalues and eigenvectors

of x. Notice that the norm of both of the Jordan frame vectors are $1/2$. Moreover, these Jordan frames are orthogonal, and they are on the rays on the boundary of the standard SOC which are the intersection of the SOC with the hyperplanes generated by x and the unit vector $[1, 0, \ldots, 0]^\top$ of the SOC. For the special case of $x = [1, 0, \ldots, 0]^\top$, any Jordan frame can be chosen. See Figure 1.1 for a representation of Jordan frames for the vector x.

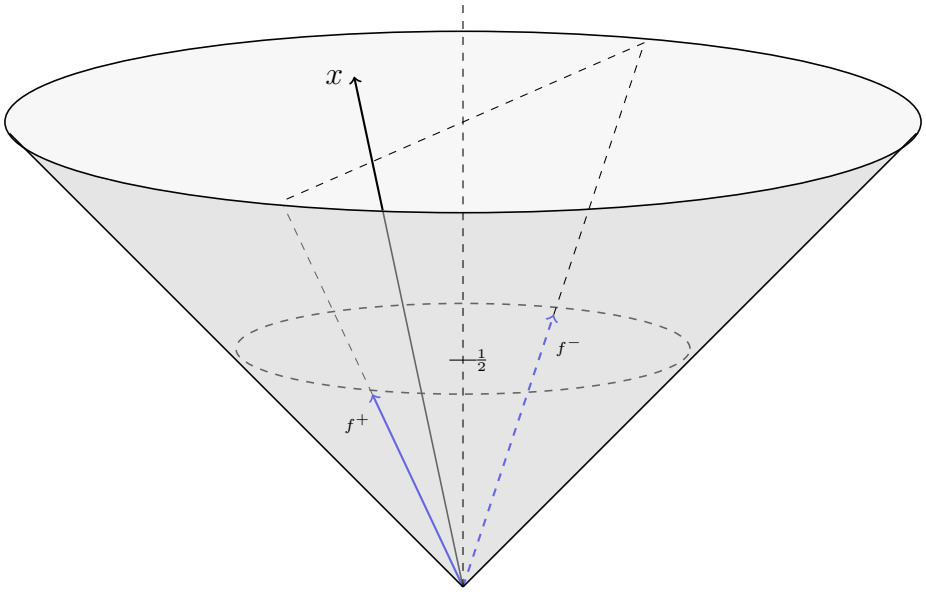


Figure 1.1: Jordan frames in a cone.

1.1.2 Primal and dual rounding problems

Terlaky and Pólik [101] and Pólik and Góez [89] presented rounding solutions for SOCO problems. Their method takes a near-optimal solution of an IPM iteration, fixes its Jordan frames, and solves the resulting LO problems. These LO problems are called "rounding problems".

Suppose we have a conic feasible primal and dual solution $x \in \mathcal{K}, z \in \mathcal{K}$. We derive primal and dual rounding problems for the given solutions. Denote f_P and f_D the

Jordan frames which correspond to the primal variable x and the dual slack variable z, respectively. For notational convenience, we merge Jordan frames of all cones into matrices $F_P \in \mathbb{R}^{n\times 2k}$ and $F_D \in \mathbb{R}^{n\times 2k}$ for the primal and dual side, respectively, which consist of two block-diagonal parts. Denote

$$F_P = \begin{bmatrix} f_{P^1}^+ & 0 & \dots & 0 & f_{P^1}^- & 0 & \dots & 0 \\ 0 & f_{P^2}^+ & \dots & 0 & 0 & f_{P^2}^- & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & f_{P^k}^+ & 0 & 0 & \dots & f_{P^k}^- \end{bmatrix},$$

$$F_D = \begin{bmatrix} f_{D^1}^+ & 0 & \dots & 0 & f_{D^1}^- & 0 & \dots & 0 \\ 0 & f_{D^2}^+ & \dots & 0 & 0 & f_{D^2}^- & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & f_{D^k}^+ & 0 & 0 & \dots & f_{D^k}^- \end{bmatrix}.$$

Similarly, denote $\lambda \in \mathbb{R}^{2k}$ and $\kappa \in \mathbb{R}^{2k}$ as the vectors of the primal and dual Jordan values, respectively:

$$\lambda = \begin{bmatrix} \lambda_1^+ \\ \lambda_2^+ \\ \vdots \\ \lambda_k^+ \\ \lambda_1^- \\ \lambda_2^- \\ \vdots \\ \lambda_k^- \end{bmatrix} \qquad \kappa = \begin{bmatrix} \kappa_1^+ \\ \kappa_2^+ \\ \vdots \\ \kappa_k^+ \\ \kappa_1^- \\ \kappa_2^- \\ \vdots \\ \kappa_k^- \end{bmatrix}.$$

Now we can write $x = F_P\lambda$ and $z = F_D\kappa$. To find a lower rank solution, we can fix the Jordan frames F_P and F_D at a solution (x, y, z) and approximate the primal and dual SOCOs as LO problems. The dual variable y and the Jordan values λ and κ are our decision variables in these rounding problems.

Using the introduced notation, the primal rounding (PR) problem can be written

as follows:

$$\begin{aligned} \text{minimize:} \quad & (c^\top F_P)\lambda & & \\ \text{subject to:} \quad & (AF_P)\lambda & = & \ b, \\ & \lambda & \geq & \ 0. \end{aligned} \tag{PR}$$

The dual rounding problem is written as follows:

$$\begin{aligned} \text{maximize:} \quad & b^\top y & & \\ \text{subject to:} \quad & A^\top y + F_D\kappa & = & \ c, \\ & \kappa & \geq & \ 0. \end{aligned} \tag{DR}$$

Notice that (PR) and (DR) are not duals of each other. We can write the duals of both the primal and dual rounding problems. The dual of (PR) is written as follows:

$$\begin{aligned} \text{maximize:} \quad & b^\top y & & \\ \text{subject to:} \quad & F_P^\top A^\top y + u & = & \ F_P^\top c, \\ & u & \geq & \ 0. \end{aligned} \tag{D-PR}$$

Finally, the dual of the (DR) is written as follows:

$$\begin{aligned} \text{minimize:} \quad & c^\top x & & \\ \text{subject to:} \quad & Ax & = & \ b, \\ & F_D^\top x & \geq & \ 0. \end{aligned} \tag{D-DR}$$

1.2 Mixed-integer second-order cone optimization (MISOCO)

MISOCO is a special branch of convex MINLO. A MISOCO problem minimizes a linear objective function over a set of linear and second-order cone constraints. By definition, the positive half line $\mathbb{R}_+$ is also a Lorentz cone and so $\mathbb{R}_+^n$ is the product of n one-dimensional Lorentz cones. Thus, MISOCO can be also seen as a generalization of MILO.

MISOCO problems include integer variables on top of SOCO problems (P-SOCO). A MISOCO in the standard form can be written as follows:

$$\begin{aligned} \text{minimize:} \quad & c^\top x \\ \text{subject to:} \quad & Ax = b, \\ & x \in \mathcal{K}, \\ & x \in \mathbb{Z}^d \times \mathbb{R}^{n-d}. \end{aligned} \tag{MISOCO}$$

MISOCO is an active research area, and there have been significant advances in both theoretical and practical aspects of it. An increasing number of studies and the development of both commercial and open-source SOCO solvers have increased the interest to the field. Commercial solvers like CPLEX and MOSEK provide functionality to model and solve MISOCO problems, and they can be seen as a proof of interest in MISOCO.

Two major reasons drive this rising interest in MISOCO. First, many real-life applications from various sectors like finance, energy, and healthcare can be formulated as a MISOCO problem. Certain types of portfolio optimization problems [20] and option-pricing problems [88] from finance, the turbine balancing problem [41] from energy, and the problem of sterotactic surgery treatment planning with isocentre selection [53] from healthcare are some of the applications in which MISOCO formulations appear. An extensive list of problems are given by Ben-Tal and Nemirovski [19] and Lobo et al. [72]. The second reason is that there are efficient IPM-based solution methodologies to the underlying problems that can solve SOCO problems in polynomial time as mentioned in the previous section.

The following subsection briefly reviews the solution methodologies for MISOCO problems.

1.2.1 Solution algorithms for MISOCO

In their survey, Benson and Sağlam [20] present solution strategies for MISOCO. They argue that a very intuitive solution strategy is to use a branch-and-bound

algorithm where each node is solved as a SOCO subproblem. These subproblems can be solved by an IPM, which is described in Section 2.1. An effective implementation, especially for large-scale problems, must benefit from warm-start, heuristics, and cut management strategies, which are the main foci of this dissertation.

MISOCO problems can be solved by any MINLO solution algorithms. There are three main strategies for solving MINLO problems. Duran and Grossmann [42] present an outer-approximation algorithm to solve MINLO problems. This ad-hoc adaptive linearization technique can be used to solve MISOCO, since SOC constraints are convex and thus satisfy assumptions of the given method. Recently, Bulut [31] covered this solution approach for MISOCO in his dissertation. His implementation uses adaptive linearization to solve resulting SOCO subproblems in a BCC framework. Westerlund and Pettersson [105] extend Kelley's cutting-plane method for MINLO, while the original method is applicable only to convex MINLO problems. The proposed method can solve a large convex MINLO problem with a moderate degree of nonlinearity. A generalized Benders decomposition is presented by Geoffrion [52].

MISOCO problems can be also solved by approximating the underlying SOCO problems with LO [19, 104]. Thus, the original MISOCO problem can be solved with a branch-and-bound algorithm that uses a simplex-based method after a structrual linearization. Such an approach may also benefit from the warm-starting capabilities of the simplex-based methods.

In her thesis, Drewes [41] proposes two methods to solve MISOCO problems. The first is a naive branch-and-cut algorithm that benefits from Chvátal-Gomory cuts for MISOCO problems. The second is a hybrid approach that combines a branch-and-bound algorithm with outer-approximation.

1.2.2 Branch-and-bound algorithms

B&B is a popular way to solve integer optimization problems. The basic idea behind the algorithm is to divide a problem into subproblems and then solve the continuous relaxations at every node. These subproblems can be solved exactly or partially, depending on the type of the B&B algorithm. Each subproblem provides a lower bound for its branch, and every integer feasible node provides a global upper bound. The aim of the algorithm is to close the gap between the lower and upper bounds to zero when optimality is reached. The structure of a naive branch-and-bound algorithm is given in Algorithm 1 [70, 71]. Note that B&B is an exact algorithm.

Algorithm 1 Branch-and-bound algorithm framework

Require: A mixed-integer problem P
Ensure: Optimal solution x^*, infeasible flag if $x^* = \emptyset$, or unboundedness flag if $x^* = -\infty$.

1 $\mathcal{L} = \{\mathrm{P}\}$. Set $\mathrm{z_U} = \infty, \mathrm{z_L^P} = -\infty, x^* = \emptyset$. ▷ Initialize
2 **if** $\mathcal{L} = \emptyset$ **then** ▷ Terminate
3 **return** x^*
4 Choose and delete a problem N from $\mathcal{L}$. ▷ Select
5 Solve linear relaxation of N. ▷ Evaluate
6 **if** linear relaxation of N is infeasible **then**
7 Go to Step 2.
8 **else**
9 Let z be the optimal objective function value and x be the solution of linear relaxation of N.
10 **if** $\mathrm{z} \geq \mathrm{z_U}$ **then** ▷ Prune
11 Go to Step 2.
12 **else if** x is integer feasible **then**
13 Set $\mathrm{z_U} = \mathrm{z}, x^* = x$. Delete all problems with $\mathrm{z_L^j} \geq \mathrm{z_U}$ from $\mathcal{L}$. Go to Step 2.
14 Divide feasible region of N into smaller regions $\mathrm{N}^1, \mathrm{N}^2, \ldots, \mathrm{N}^k$ such as ▷ Branch

$$\cup_{j=1}^{k} \mathrm{N}^j = \mathrm{N}, \quad \mathrm{z_L^j} = \mathrm{z}$$

and add them to $\mathcal{L}$. Go to step 2.

In Algorithm 1, $\mathcal{L}$ denotes the set of active subproblems. Branch-and-cut (B&C) algorithms combine a B&B algorithm with cutting-plane methods. The basic idea is to add valid cuts at each subproblem, which improves the bound obtained from the subproblems. Compared to B&B, B&C algorithms usually generate smaller search

trees, if the same rules are applied. On top of the challenges coming from B&B, there are also other questions to be answered for B&C algorithms. Which types of cuts to add, when to add, and when to stop cutting and continue with branching are some of these basic questions, which are also in the interest of this thesis for MISOCO problems.

BCC algorithm

We are interested in applying branch-and-bound algorithms to solve MISOCO problems, where the algorithm is enhanced by addition of linear and conic cuts. This method is called BCC. A general representation of BCC is given in Algorithm 2.

Algorithm 2 Branch-and-conic-cut algorithm framework for MISOCO

Require: A MISOCO problem P of the form (4.1)
Ensure: Optimal solution x^*, infeasible flag if $x^* = \emptyset$, or unboundedness flag if $x^* = -\infty$.

1 $\mathcal{L} = \{\mathrm{P}\}$. Set $\mathrm{z_U} = \infty, \mathrm{z_L^P} = -\infty, x^* = \emptyset$. ▷ Initialize
2 **if** $\mathcal{L} = \emptyset$ **then** ▷ Terminate
3 **return** x^*
4 Choose and delete a problem N from $\mathcal{L}$. ▷ Select
5 Apply solution heuristics on N.
6 Solve continuous relaxation of N with an IPM with warm-starting. ▷ Evaluate
7 **if** continuous relaxation of N is infeasible **then**
8 Go to Step 2.
9 **else**
10 Search for valid linear and conic cuts
11 **if** valid cuts exist **then**
12 Add linear and conic cuts to the subproblem N. ▷ Cut generation
13 Go to Step 6.
14 **else**
15 Let z be the optimal objective function value and x be the solution of continuous relaxation of N.
16 **if** $\mathrm{z} \geq \mathrm{z_U}$ **then** ▷ Prune
17 Go to Step 2.
18 **else if** x is integer feasible **then**
19 Set $\mathrm{z_U} = \mathrm{z}, x^* = x$. Delete all problems with $\mathrm{z_L^j} \geq \mathrm{z_U}$ from $\mathcal{L}$. Go to Step 2.
20 Divide feasible region of N into smaller regions $\mathrm{N}^1, \mathrm{N}^2, \ldots, \mathrm{N}^k$ such as ▷ Branch

$$\cup_{j=1}^{k} \mathrm{N}^j = \mathrm{N}, \quad \mathrm{z_L^j} = \mathrm{z}$$

and add them to $\mathcal{L}$. Go to step 2.

1.3 Structure

This dissertation consists of four parts. In Chapter 2, background and state-of-the-art information about related topics are presented. In Chapter 3, we present a novel warm-start methodology for SOCO problems by using rounding on IPMs. In Chapter 4, we present novel primal heuristics for MISOCO problems using the Jordan frames of SOCs. In Chapter 5, we present the effects of applying of disjunctive conic and cylindrical cuts (DCC) on asset allocation problems (AAPs). Finally, in Chapter 6, we give conclusions and future research directions.

Chapter 2

Background and state of the art

2.1 Interior-point methods (IPMs)

After Karmarkar's paper [67], the field of IPMs saw a rapid expansion. IPMs are considered to be efficient methods for solving large-scale linear and convex optimization problems. As discussed earlier, IPMs are also efficient methods for solving SOCO problems. In addition to their polynomial complexity, IPMs are often faster than simplex-type algorithms in practice. Survey papers, see Terlaky [100] for example, cover key aspects of IPMs for interested readers. Extensive review on IPMs can be found in the literature for LO [92, 106, 108] and semi-definite optimization [40].

In this section, we recall some well-known facts about the primal-dual interior-point methods (PDIPMs) for LO and SOCO, in that order. We also give a literature review on warm-starting for IPMs.

2.1.1 IPMs for LO

In this section, we present the key concepts of a PDIPM for LO. The contents of this section are taken from Roos et al. [92]. Consider an LO problem in the standard

form,

$$\begin{aligned} \text{minimize:} \quad & c^\top x \\ \text{subject to:} \quad & Ax = b, \\ & x \geq 0, \end{aligned} \tag{2.1}$$

where the vectors $c \in \mathbb{R}^n$ and $x \in \mathbb{R}^n$, the matrix $A \in \mathbb{R}^{m \times n}$, and vector $b \in \mathbb{R}^m$. Without loss of generality, we assume that A is a matrix with full row rank.

The associated dual problem can be written as

$$\begin{aligned} \text{maximize:} \quad & b^\top y \\ \text{subject to:} \quad & A^\top y + z = c, \\ & z \geq 0, \end{aligned} \tag{2.2}$$

where vector $y \in \mathbb{R}^m$ and vector $z \in \mathbb{R}^n$ are the dual variables and dual slacks, respectively. These two problems are called the *primal-dual pair*. The optimality conditions of the primal-dual pair are as follows.

Theorem 1. *The vector $x \in \mathbb{R}^n$ is an optimal solution of (2.1) if and only if there exist vectors $y \in \mathbb{R}^m$ and $z \in \mathbb{R}^n$ such that*

$$\begin{aligned} Ax = b, \quad & x \geq 0, \\ A^\top y + z = c, \quad & z \geq 0, \\ x_i z_i = 0, \quad & i = 1, \ldots, n. \end{aligned} \tag{2.3}$$

The first and second conditions in (2.3) are called primal feasibility condition and dual feasibility condition, respectively. The third set of conditions are called *complementarity conditions*. A triplet (x, y, z) is called a *primal-dual optimal solution* if it satisfies the optimality conditions (2.3). Let $\mathcal{F}$ and $\mathcal{F}^0$ denote the primal-dual feasible set and strictly feasible set, respectively, such as

$$\begin{aligned} \mathcal{F} &= \{(x, y, z) \mid Ax = b, A^\top y + z = c, x \geq 0, z \geq 0\}, \\ \mathcal{F}^0 &= \{(x, y, z) \mid Ax = b, A^\top y + z = c, x > 0, z > 0\}. \end{aligned} \tag{2.4}$$

A triplet $(x, y, z) \in \mathcal{F}^0$ satisfying (2.3) is called a *strictly complementary optimal solution* if it satisfies $x + z > 0$. The existence of a point in the strictly feasible set

$\mathcal{F}^0$ is referred as the interior-point condition (IPC). Roos et al. [92] present self-dual embedding, which transforms a given LO problem into another formulation that a feasible interior point is known for. Here, we assume that a strictly feasible solution for the primal dual pair exists and is known.

PDIPMs generate a sequence of strictly feasible points of the primal-dual problems following the so-called *central path*. The central path is parametrized by a positive scalar μ. The perturbed system is written as

$$\begin{aligned} Ax &= b, \quad x \geq 0, \\ A^\top y + s &= c, \quad s \geq 0, \\ x_i z_i &= \mu, \quad i = 1, \ldots, n. \end{aligned} \tag{2.5}$$

Each point (x, y, z) is on the central path $\mathcal{C}$ if it solves the perturbed system. Note that the perturbed system (2.5) consists of the optimality conditions of the so-called *primal and dual logarithmic barrier problems*. The primal and dual logarithmic barrier problems are written as follows:

$$\begin{aligned} &\text{minimize} \left\{ c^\top x - \mu \sum_{i=1}^{n} \log x_i : Ax = b, x > 0 \right\}, \\ &\text{maximize} \left\{ b^\top y - \mu \sum_{i=1}^{n} \log z_i : A^\top y + z = c, z > 0 \right\}. \end{aligned}$$

System (2.5) has a unique solution for each $\mu > 0$. For a given $\mu > 0$, denote $(x(\mu), y(\mu), z(\mu))$ as the unique solution of system (2.5). The primal-dual central path approaches the set of optimal solutions as $\mu \to 0$, and their limit point at $\mu = 0$ is the analytic center of the optimal set.

PDIPMs use Newton steps to follow the central path. For a given feasible (x, y, s) with $x > 0$ and $z > 0$, the Newton step $(\Delta x, \Delta y, \Delta z)$ is the unique solution of the following system

$$\begin{aligned} A\Delta x &= 0, \\ A^\top \Delta y + \Delta z &= 0, \\ Z\Delta x + X\Delta z &= \mu e - XZe, \end{aligned} \tag{2.6}$$

where $X = \text{diag}(x_1, \ldots, x_n)$, $Z = \text{diag}(z_1, \ldots, z_n)$, and $e = (1, \ldots, 1)^\top \in \mathbb{R}^n$. PDIPMs often take a damped Newton step that is parameterized by the step length α, such as the new point becoming $(x + \alpha\Delta x, y + \alpha\Delta y, z + \alpha\Delta z)$ with $x + \alpha\Delta x > 0$ and $z + \Delta z > 0$. In the implementation of PDIPMs, we need to solve the Newton system (2.6) for each interior-point iteration.

We can rewrite (2.6) as follows

$$\begin{bmatrix} 0 & A & 0 \\ A^\top & 0 & I \\ 0 & Z & X \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta x \\ \Delta z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \mu e - XZe \end{bmatrix}. \tag{2.7}$$

Now, we can eliminate the change in the dual slack variable Δz from the formulation by using the third block of equations as

$$\Delta z = X^{-1}(\mu e - XZe - Z\Delta x).$$

Let $D = Z^{-1/2}X^{1/2}$. We can simplify (2.7) to the so-called *augmented system*; that is

$$\begin{bmatrix} 0 & A \\ A^\top & -D^{-2} \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta x \end{bmatrix} = \begin{bmatrix} 0 \\ -X^{-1}(\mu e - XZe) \end{bmatrix}.$$

We can further eliminate Δx from the system as

$$\Delta x = D^2 A^\top \Delta y + D^2 X^{-1}(\mu e - XZe),$$

and reach the most compact form of the system, which is called *normal equations*

$$AD^2 A^\top \Delta y = -AZ^{-1}(\mu e - XZe).$$

Inside an IPM, either the augmented system or the normal equation is solved at every iteration. Normal equations are a popular choice since the sparsity structure of $AD^2A^\top$ is fixed, and so symbolic sparse Cholesky factorization combined with numerical factorization can be applied to the matrix $AD^2A^\top$.

During IPM iterations, we need to measure the distance of the iterates to the central path. Several measures are available in the literature. One of the

neighborhoods defined by the Euclidean norm is the so-called 2-norm neighborhood, $\mathcal{N}_2(\theta)$, which is defined as

$$\mathcal{N}_2(\theta) = \left\{(x, y, z) \in \mathcal{F}^0 : \|XZe - \mu_q e\|_2 \leq \theta\mu_q\right\}, \tag{2.8}$$

where $\mathcal{F}^0$ is defined in (2.4), $\theta < 1$ is a given parameter, and the central path parameter μ_q related to the duality gap $\mu_q = x^\top z/n$. The neighborhood $\mathcal{N}_2(\theta)$ defines a narrow neighborhood around the central path. The one-sided ∞-norm neighborhood, $\mathcal{N}_{-\infty}(\gamma)$, is defined by

$$\mathcal{N}_{-\infty}(\gamma) = \{(x, y, z) \in \mathcal{F} : x_i z_i \geq \gamma\mu_q \quad \forall i\}, \tag{2.9}$$

for some $0 < \gamma < 1$. This is a rather large neighborhood definition which prevents iterates from getting too close to the boundary of $\mathbb{R}^n_+$.

Finally we can write the general scheme of PDIPMs as shown in Algorithm 3.

Algorithm 3 PDIPMs for LO

Require: An LO problem in the form (2.1),
a neighborhood definition $\mathcal{N}$ around the central path,
an accuracy parameter $\epsilon > 0$,
a variable dumping factor α,
a barrier update parameter $0 < \theta < 1$,
a strictly feasible starting point $(x^0, y^0, z^0) \in \mathcal{F}^0$ in the neighborhood $\mathcal{N}$ of the central path.

$x = x^0, y = y^0, z = z^0$
while $n\mu \geq \epsilon$ **do**
  $\mu = (1 - \theta)\mu$ ▷ Duality parameter
  while $(x, y, z) \notin \mathcal{N}(x, z, \mu)$ **do**
    Solve the Newton system (2.6) for $(\Delta x, \Delta y, \Delta z)$ ▷ Search direction
    $x = x + \alpha\Delta x$ ▷ Newton step
    $y = y + \alpha\Delta y$
    $z = z + \alpha\Delta z$
return (x, y, z)

2.1.2 IPMs for SOCO

IPMs for SOCO differ slightly from IPMs for LO [8]. Consider a SOCO problem in the form of (P-SOCO) and its associated dual problem (D-SOCO). Assume that

we have feasible interior solutions for both the primal and the dual problem. The primal-dual optimality conditions are written as

$$\begin{aligned} Ax = b, \quad & x \in \mathcal{K}, \\ A^\top y + z = c, \quad & z \in \mathcal{K}, \\ x^i \circ z^i = 0, \quad & i = 1, \dots, k, \end{aligned}$$

where $\circ$ is a the Jordan product of two vectors defined as

$$u \circ v = (u^\top v; u_1 v_{2:n} + u_1 v_{2:n}).$$

Note that this product can be also represented as

$$u \circ v = \begin{bmatrix} u^\top v \\ u_1 v_2 + u_1 v_2 \\ \vdots \\ u_1 v_n + u_1 v_n \end{bmatrix} = \mathrm{Arr}(u)v = \mathrm{Arr}(u)\mathrm{Arr}(v)\iota,$$

where $\iota = (1, 0, \dots, 0) \in \mathbb{R}^n$ and $\mathrm{Arr}(u), \mathrm{Arr}(v)$ are the arrowhead matrices of vectors u and v, respectively. An arrowhead matrix associated with a vector $u \in \mathbb{R}^n$ is defined as

$$\mathrm{Arr}(u) = \begin{bmatrix} u_1 & u_2 & \dots & u_n \\ u_2 & u_1 & & \\ \vdots & & \ddots & \\ u_n & & & u_1 \end{bmatrix}.$$

We perturb the complementarity conditions as we did for LO, and get the central path as

$$\begin{aligned} Ax = b, \quad & x \in \mathcal{K}, \\ A^\top y + z = c, \quad & z \in \mathcal{K}, \\ x^i \circ z^i = 2\mu\iota^i, \quad & i = 1, \dots, k, \end{aligned}$$

where $\iota^i = (1, 0, \ldots, 0) \in \mathbb{R}^{n_i}$ [6, 90]. The Newton system to find the search direction is written as

$$\begin{aligned} A\Delta x &= 0, \\ A^\top \Delta y + \Delta z &= 0, \\ x^i \circ \Delta z^i + \Delta x^i \circ z^i &= 2\mu\iota^i - x^i \circ z^i, \quad i = 1, \ldots, k, \end{aligned} \tag{2.10}$$

where $\Delta x = ((\Delta x^1)^\top, \ldots, (\Delta x^k)^\top)^\top$ and $\Delta z = ((\Delta z^1)^\top, \ldots, (\Delta z^k)^\top)^\top$. Let $\mathrm{Arr}(x)$ and $\mathrm{Arr}(z)$ be the block-diagonal matrices built by $\mathrm{Arr}(x^i)$ and $\mathrm{Arr}(z^i)$, respectively. We can rewrite system (2.10) by using the arrowhead matrix notation, such as

$$\begin{bmatrix} 0 & A & 0 \\ A^\top & 0 & I \\ 0 & \mathrm{Arr}(z) & \mathrm{Arr}(x) \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta x \\ \Delta z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2\mu\iota - x \circ z \end{bmatrix},$$

where $\iota = ((\iota^1)^\top, \ldots, (\iota^k)^\top)^\top$. If we eliminate Δx and Δz from the system, we get the normal equation for the SOCO case, which is

$$\left(A\mathrm{Arr}^{-1}(z)\mathrm{Arr}(x)A^\top\right)\Delta y = -A\mathrm{Arr}^{-1}(z)(2\mu\iota - x \circ z).$$

In this system, the coefficient matrix is a square matrix; however, it is not symmetric and in general it can be singular. To prevent such cases, block-diagonal scaling is usually applied in IPMs for SOCO. See Andersen et al. [8] for computational methods for implementing IPMs for SOCO in more detail.

Let p be a vector in the interior of $\mathcal{K}$. We define the scaling matrix $Q_p \succ 0$ as

$$Q_p = 2\mathrm{Arr}(p)^2 - \mathrm{Arr}(p \circ p).$$

Let p^{-1} be the inverse of p, such as $p \circ p^{-1} = \iota$. Now, consider the following scaled problem,

$$\begin{aligned} \text{minimize:} \quad & (Q_{p^{-1}}c)^\top (Q_p x) \\ \text{subject to:} \quad & (AQ_{p^{-1}})(Q_p x) = b, \\ & Q_p x \in \mathcal{K}, \\ & x \in \mathbb{R}^n, \end{aligned} \tag{2.11}$$

where $Q_p Q_{p^{-1}} = I$ and $Q_p(\mathcal{K}) = \mathcal{K}$. Pólik and Terlaky [90] show that (P-SOCO) and (2.11) are equivalent. Similarly, the scaled dual problem is written as

$$\begin{aligned} \text{maximize:} \quad & b^\top y \\ \text{subject to:} \quad & (AQ_{p^{-1}})^\top y + Q_{p^{-1}} z = Q_{p^{-1}} c, \\ & Q_{p^{-1}} z \in \mathcal{K}, \\ & y \in \mathbb{R}^m, \\ & z \in \mathbb{R}^n, \end{aligned}$$

which is equivalent to (D-SOCO). Now the Newton system takes the following form:

$$\begin{aligned} (AQ_{p^{-1}})(Q_p \Delta x) &= 0, \\ (AQ_{p^{-1}})^\top \Delta y + Q_{p^{-1}} \Delta z &= 0, \\ (Q_p x) \circ (Q_{p^{-1}} \Delta z) + (Q_p \Delta x) \circ (Q_{p^{-1}} z) &= 2\mu\iota - (Q_p x) \circ (Q_{p^{-1}} z). \end{aligned} \tag{2.12}$$

The scaling vector p can be specified in many ways. A very popular one is Nesterov-Todd (NT) scaling [85], where

$$p = \left(Q_{x^{1/2}} \left(Q_{x^{1/2}} z \right)^{-1/2} \right)^{-1/2} = \left(Q_{z^{-1/2}} \left(Q_{z^{1/2}} x \right)^{1/2} \right)^{-1/2}.$$

This choice simplifies the variables, such that

$$Q_p x = Q_{p^{-1}} z.$$

Define $u := Q_p x = Q_{p^{-1}} z$. Then with NT scaling, (2.12) can be rewritten as follows

$$\begin{bmatrix} 0 & AQ_{p^{-1}} & 0 \\ (AQ_{p^{-1}})^\top & 0 & I \\ 0 & \mathrm{Arr}(u) & \mathrm{Arr}(u) \end{bmatrix} \begin{bmatrix} \Delta y \\ Q_p \Delta x \\ Q_{p^{-1}} \Delta z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2\mu\iota - u \circ u \end{bmatrix}. \tag{2.13}$$

After eliminating Δz, system (2.13) reduces to the *augmented system* for SOCO. That is,

$$\begin{bmatrix} 0 & A \\ A^\top & -H^{-1} \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta x \end{bmatrix} = \begin{bmatrix} 0 \\ -Q_p \mathrm{Arr}^{-1}(u)(2\mu\iota - u \circ u) \end{bmatrix},$$

where $H = Q_{p^{-2}}$ [6]. After eliminating Δx, we reach the the *normal equation* corresponding to this system, which is

$$\left(AHA^{\top}\right)\Delta y = -AQ_{p^{-1}}\mathrm{Arr}^{-1}(u)\left(2\mu\iota - u \circ u\right).$$

2.1.3 Warm-starting of IPMs

"Warm-start" is the general name of exploiting information obtained from one problem instance to solve another closely related instance [43]. Warm-starting techniques often use the solution status and/or optimal solution of the original instance in the solution process of the new instance. Warm-starting plays an important role in reducing the solution time and/or number of iterations to reach an optimal solution. The need for such a mechanism is evident, considering that many real-life applications have changing parameters and that problems are solved repeatedly. Similarly, any iteration-based optimization algorithm can benefit from warm-starting to some extent, which could lead to important savings.

The effect of warm-starting depends heavily on the algorithm itself. It is a known fact that warm-start approaches are effective for active-set methods when instances differ only slightly. Due to its efficiency, warm-start is available in many commercial primal and dual simplex method solvers [39, 61, 81]. Starting simplex iterations from the optimal solution of the original instance usually means that a new optimum can be obtained with significantly fewer iterations compared to starting from scratch [36]. If a new constraint is added to the problem, then the user may use the dual simplex method by starting from the optimal solution of the original instance. If a new variable is added to the problem and the original solution is feasible for the primal problem, then the user may use the previous solution to warm-start the primal simplex method. As mentioned earlier, warm-starting in active-set methods are efficient only if the problem is slightly changed. If many constraints or variables are added to the problem at the same time, then the efficiency of warm-start may be lost.

Warm-start of IPMs is a rather active research area [65]. It is one of the areas of IPMs that still needs to be studied further [59]. Note that the efficiency of warm-start strategies in IPMs is rather limited compared to active-set methods. In their extensive review of IPMs for LO, Roos et al. [92] point out two main issues with warm-start of IPMs. First, the optimal solution of the original instance often becomes infeasible for the new instance. Such cases often occur in B&C methods, where a cut or branch is generated to tighten the formulation. Second, even if the point stays feasible, it can be too close to the boundary, which leads a poor warm-start performance. If the starting point of an IPM is close to the boundary, then the IPM takes a series of small steps which leads to a slow progress. In such cases, a cold-start could be faster than warm-start. The research on warm-starting of IPMs focuses on handling these two issues.

This section presents a brief literature review of the topic. Since the topic is rather broad, studies are put in two groups. In the first group, results that focus on warm-starting of IPMs for LO are presented. Papers that also consider SOCO and semi-definite optimization (SDO) are discussed in the second group.

Warm-starting of IPMs for LO

Due to its importance in practice, warm-starting of IPMs has an extensive literature for the LO case. Considering the fact that IPMs are successful in terms of solving very large-scale LO problems, it is appealing to further improve the performance of the IPMs with warm-start, which can be used effectively in cutting-plane and B&C algorithms. Indeed, warm-starting is of great importance for solving integer optimization problems.

There are two lines of research in the literature of warm-starting of IPMs for LO [57]. The first line considers the perturbations in the problem parameters where the problem variables and constraints stay the same. The second line studies the addition of the new constraints or variables to the problem. These two lines are distinctively

different; thus, the reviews are presented separately, in that order.

Perturbation-based approaches in warm-starting of IPMs for LO

Perturbation-based approaches are useful in cases of reoptimization where a sequence of related problems with slightly changing data needs to be solved. Such problems arise in many engineering and business applications [43] — for example when solving sequential LO problems, in decomposition-based algorithms, or when doing a "what if" analysis.

Some early studies focus on warm-start after an LO problem is perturbed. Freund [49] uses a shifted barrier method, which allows non-negativity of the variables to satisfy the feasibility requirement of the starting point. Lustig et al. [73] perturb the problem to move the iterate away from the boundary, whereas Polyak [91] applies a modified barrier function. A review of more recent studies is given below.

Yildirim and Wright [109] consider a case where the problem parameters are slightly perturbed. They work on a LO problem in the standard form (2.1). They develop two warm-start strategies —a least-squares correction and a Newton-step correction— for the new instance where the problem parameters (A, b, c) are perturbed by a small amount $(\Delta A, \Delta b, \Delta c)$. They define the size of the perturbation as the maximum Euclidean norm of these three components, $\|\Delta d\| = \max(\|\Delta A\|_2, \|\Delta b\|_2, \|\Delta c\|_2)$, where Δd is used to represent triplet $(\Delta A, \Delta b, \Delta c)$. After storing interim primal-dual solutions (x^k, y^k, z^k) of the original instance, they aim to obtain an initial point for the perturbed instance that is inside a known neighborhood of the central path. They use the same neighborhood definitions given in (2.8) and (2.9). Among the recorded iterates, they use the one with the largest k value. Then they find an adjustment $(\Delta x, \Delta y, \Delta z)$ to this iterate that is based on the perturbation Δd. This step generates the starting point of the IPM for the new instance: $(x^0, y^0, z^0) = (x^k, y^k, z^k) + (\Delta x, \Delta y, \Delta z)$. Their methods differ in terms of how the adjustment is calculated. They emphasize that this

adjustment step costs only one iteration of an IPM. Since a large change in parameters may violate the feasibility of the initial point for the perturbed instance, they assume that the perturbation is small. Therefore, the performance of the strategies depends heavily on Δd. Storing the iterate information could be expensive, which makes this approach less useful for practical purposes.

Gondzio and Grothey [57] use a well centered solution of the original instance to generate a good starting point for the perturbed problem. They provide bounds on the size of perturbations of the problem parameters, where primal and dual feasibility can be absorbed in merely one Newton step with an infeasible IPM. Their measure for perturbations is different than Yildirim and Wright [109] and is argued to be more practical. They also use 2-norm and ∞-norm neighborhood definitions, given in (2.8) and (2.9), for their short-step and long-step methods, respectively. Their methods consist of two steps under the assumption that perturbations on the problem parameters are small. In the first step, they make a step in the Newton direction to recover primal and dual feasibility. In the second step, quality of proximity to the central path is restored. They emphasize that primal and dual feasibility is preserved after the second step only if the perturbation is small. They implemented these approaches in an object-oriented parallel solver, OOPS. They report that the number of iterations to reach an optimal solution decreases by 33% on average. Their parallel implementation is also shown to be efficient by their numerical experiments.

Benson and Shanno [21] consider a case where the objective vector c and the right hand side vector b are perturbed. They use an infeasible IPM; hence the optimal solution of the original instance could be used as a starting point. However, they emphasize that starting from this solution may increase the number of iterations that IPMs take to solve the perturbed instance. Therefore, they propose a primal-dual penalty approach to find a good starting point for the perturbed instance. A penalty term ℓ_1 is applied to both the primal and the dual problems. The new LO instances are solved by an IPM by starting from the optimal solution of the original instance. This

method prevents short step lengths and provides an improvement on the performance. They show that the suggested method provides a decrease around 40-50% in the number of IPM iterations in their experiment of solving the LO problems of the Netlib library.

Modification-based approaches in warm-starting of IPMs for LO

Modification-based approaches consider cases when a new constraint or a variable are added to the problem. Such transformations occur in cutting-plane, column generation [60], and B&B algorithms.

Mitchell and Todd [78] present one of the earliest discussions on warm-starting of IPMs after the original problem is modified. They propose an approach to solving combinatorial optimization problems by applying a cutting plane algorithm, where the relaxations are solved by using IPMs. They start solving each instance from an optimal solution of the previous instances. They show that the approach is not efficient in general by providing numerical experiments. They conclude that to make IPMs competitive with simplex methods, it is necessary to have warm-start strategies that limit the number of iterations and reduce time.

Gondzio [55] proposes a warm-start procedure for infeasible primal-dual IPMs within a cutting plane method. He assumes that the size of the master problem is very large, so that the use of IPMs is encouraged. Consider an LO problem in the standard form (2.1) and its dual (2.2). The original problem is solved by using a PDIPM to moderate accuracy. Let μ be the corresponding duality parameter of the approximate primal-dual solution (x_μ, y_μ, s_μ). Linear cuts in the form of $\bar{A}^\top y \leq \bar{c}$ are generated for the dual problem based on this approximate solution. Notice that an optimal solution of the original instance may be too close to the boundary, or infeasible for the new instance. To overcome the difficulty of having a solution that is close to the boundary, they classify cuts as *deep* or *shallow* based on their impact on the μ-center system (2.5). This definition of *deep* cuts is based on the computational

experience that the infeasibility of the cut cannot be absorbed easily by an infeasible PDIPM. On the other hand, infeasibility of a shallow cut can be absorbed usually in one Newton step. In the next step, these new cuts are added to the problem. They initialize dual slack variables of the new cuts based on the violations. The primal feasibility is restored by a change in primal variable x that depends on the scaling of the problem. Since the old solution x_μ was strictly positive, there always exists an $\alpha > 0$ such that $x_\mu + \alpha\Delta x > 0$, which satisfies primal feasibility for the new instance. A fixed value of α is chosen by applying a ratio test. On the other hand, restoring dual feasibility is a relatively harder problem. Centrality of the initial point is violated to restore duality. Gondzio [55] tells that the provided method has no guarantee, especially for large violations. As a numerical example, the method is applied to solve multi-commodity flow problems, where many deep cuts are needed to reach an optimal solution. The average number of IPM iterations stays between 6 and 15, whereas cold-start needs 40 to 60 iterations. It is a significant improvement, considering the fact that many cuts are being added to the problem at the same time.

As a follow-up of [55], a warm-starting approach for a primal-dual column generation method is presented by Gondzio and González-Brevis [56]. They propose a new warm-starting technique to be used on a particular class of combinatorial optimization problems. These problems have some special structures, such as that coefficients of matrices A and $\bar{A}$ are non-negative. Their approach consists of two steps. In the first step, they try to find an initial point among the earlier iterations instead of using an optimal solution of the original instance. They check iterates that are inside a known neighborhood of the central path. The second step is finding a direction based on this initial point. In contrast to Gondzio [55], they do not choose the value of $\bar{x}$ as a function of $\bar{s}$. They define and solve an LO problem to recover dual feasibility. The aim of this problem is minimizing the relative changes of variables. Similarly, primal feasibility is restored by solving another LO problem, where the decision variable is Δx. They keep $(\Delta x, \Delta s)$ small by solving

these LO problems to avoid large variations. In this way, the user can have control over the new duality gap that results from the adjustment of the variables. In their numerical experiments, warm-start saves 30% to 48% in the number of iterations and 19% to 45% in CPU time.

Roos et al. [92] discuss warm-start strategies for self-dual embedding IPMs. They propose using a well-centered and almost optimal IPM iteration of the original instance as an initial point for the new instance. Auxiliary values are redefined to keep the point well-centered for the new system. This warm-start approach allows simultaneous perturbations of problem parameters. This approach is argued to be applicable for cases where new constraints and variables are added to the problem.

Recently, Munari and Gondzio [82] reviewed the challenges of using IPMs in a branch, price, and cut (BPC) method. They provide strategies for using IPMs more efficiently in a BPC method and present an overview of the related literature. Their discussion on the topic ends with an application of the aforementioned methods to the vehicle routing problem (VRP). By using column generation, cut generation, and branching procedures, with IPMs in mind, they reach some interesting results. They apply warm-start procedures of Gondzio [55] and Gondzio and Grothey [57] to these problems. They store non-optimal and well-centered solutions, which are used after column and cut generation. Their work is a good example of how warm-starting after column and cut generation can be applied to IPMs within a BPC algorithm context.

Warm-starting of IPMs for SOCO and SDO

The literature on the warm-starting approaches that focus on SOCO and SDO is rather limited compared to the literature on warm-starting for LO. Proposed approaches are usually extensions of the LO methodology. For example, Engau [43] presents a primal-dual penalty method for SDO by extending the work of Benson and Shanno [21]. Although SOCO and SDO problems have SOC and semidefinite constraints on top of the linear ones, existing studies in the warm-starting literature

focus only on linear constraints.

Recently, Skajaa et al. [96] present two warm-start approaches for primal-dual IPM for LO and SOCO problems. Both of their approaches use the final iteration of the original instance. These approaches are not computationally expensive, which is argued to be useful in practice by the authors. The first approach uses the primal optimal solution of the original instance, say x^*, whereas the second approach uses the optimal solution of the primal-dual pair, (x^*, y^*, s^*). They assume that the conic constraints are exactly the same both in the original and in the new instance, if any exists. They consider a case where only linear parameters (A, b, c) are perturbed. Perturbation on the parameters are limited by the quantities $(\alpha, \alpha', \beta, \gamma)$, where

$$\begin{aligned}
\|\Delta A\| &\leq \alpha \|A\|, \\
\|\Delta A^\top\| &\leq \alpha' \|A^\top\|, \\
\|\Delta b\| &\leq \beta \|b\|, \\
\|\Delta c\| &\leq \gamma \|c\|.
\end{aligned}$$

If any conic constraint is present in the problem, then the proposed method changes slightly. Authors provide extensive experiment results to demonstrate the gain from warm-starting. For each warm-start they use the following measure to quantify the gain from warm-starting:

$$\mathfrak{R} = \frac{\text{\# Iterations to solve new instance by using warm-start}}{\text{\# Iterations to solve new instance by using cold-start}}.$$

They define a performance measure $\mathfrak{G}$ as the geometric mean of the measures $\mathfrak{R}$ such as

$$\mathfrak{G} = \sqrt[K]{\mathfrak{R}_1 \cdots \mathfrak{R}_K}.$$

They use LO problems in the Netlib test library to test their methods. They use Markowitz' portfolio selection problem as their SOCO formulation. The changing nature of the parameters of the portfolio selection problem makes it a good test bed. In all these experiments, they show that the performance measure $\mathcal{G}$ is in the range

of 30—75%. As one expects, the greater the perturbation parameters, the lesser the reduction in the iterates.

Yonekura and Kanno [111] present an application of warm-start for SOCO. They propose a warm-starting strategy, which is another extension of [21]. They aim to enhance the numerical performance of the primal-dual IPM for quasistatic elastoplastic analysis with the von Mises yield criterion. They solve a sequence of closely related SOCO problems, due to the change in parameters. They solve SOCO penalty problems to be able to deploy their warm-start approach for the original SOCO problems. They show that the average time reduction by using warm-start is around 66%.

To the best of our knowledge, there are two open research areas on this topic. First, modification-based approaches for SOCO problems are not yet studied. We introduce a novel warm-start approach for modifiation-based changes on SOCO problems in Chapter 3. Second, the development of conic cuts for MISOCO is rather recent; therefore, there are currently no studies that focus on the warm-starting approaches after addition of these conic cuts.

2.2 BCC algorithms for MISOCO

The B&C algorithm is a popular choice for solving MILO problems. Therefore, B&C-based solution techniques have a rich literature especially for the linear case. This section presents a review of the basic components of a BCCP algorithm for MISOCO. We extend the discussion to methods that are specific for SOCO.

This section is divided into and presented in three parts: branching rules, searching rules, and linear and conic cuts.

2.2.1 Branching

Branching is the most fundamental component of B&B and B&C algorithms. In B&C, valid inequalities are added to the problem to improve the solutions of the subproblems. However, generating too many cuts leads to numerical problems, and thus branching must be performed at reasonable points in B&C algorithms.

The branching operation basically divides the problem N into subproblems $\mathrm{N}^1, \ldots, \mathrm{N}^k$. The union of these subproblems covers all feasible solutions of the problem, such as

$$\bigcup_{j=1}^{k} \mathrm{N}^j = \mathrm{N}.$$

Bounds obtained from the branching operation can be different from those obtained by cutting. For that reason, branching may be preferred to cutting inside a B&C algorithm [66].

There are many strategies to answer the question "How to branch" for MILO problems.

A popular and simple approach is to branch on variables where the candidate set consists of fractional variables. We can write subproblem i associated with variable x_i as

$$\mathrm{N}^i_- = \mathrm{N} \cap \left\{ x_i \leq \left\lfloor x_i^{\mathrm{N}} \right\rfloor \right\}, \quad \mathrm{N}^i_+ = \mathrm{N} \cap \left\{ x_i \geq \left\lceil x_i^{\mathrm{N}} \right\rceil \right\},$$

where x_i^{N} is the value of x_i in the relaxed solution of N. Alternatively, branching can be applied on a general disjunction in the form of $a^\top x \leq a_0$ and $a^\top x \geq a_0 + 1$, where the subproblems are written as

$$\mathrm{N}_- = \mathrm{N} \cap \left\{ a^\top x \leq a_0 \right\}, \quad \mathrm{N}_+ = \mathrm{N} \cap \left\{ a^\top x \geq a_0 + 1 \right\}. \tag{2.14}$$

In general, a candidate set C consists of such subproblems. Then, scoring of these candidates is performed, and the candidate pair with the highest score is selected and added to the set of active problems $\mathcal{L}$ as shown in Algorithm 1.

The following subsection reviews general-purpose branching rules, including both construction of the candidate set and the scoring methods.

General purpose branching rules

The content of this section is taken from Achterberg et al. [3] and Drewes [41].

1. Index-based branching: This branching rule is one of the simplest methods in the literature. The candidate set consists of variables that are currently fractional for the original problem. The variable with the lowest index is selected for branching.

2. Random branching: This rule does not apply any scoring. A variable is chosen randomly from the fractional candidates for branching.

3. Most fractional branching: This rule chooses the most fractional variable among the variables that are fractional for the original problem. Scoring is made based on the distance to the half, such as

$$\mathfrak{u}_i = \min\left\{\left\lceil x_i^{\mathrm{N}}\right\rceil - x_i^{\mathrm{N}}, x_i - \left\lfloor x_i^{\mathrm{N}}\right\rfloor\right\},$$

   where $\mathfrak{u}_i$ is the score of the candidate pair i. Achterberg et al. [3] show that the most fractional branching rule does not perform better than the random branching rule, in general.

4. Pseudocost branching: The pseudocost branching rule applies a more advanced rule. The idea of pseudocost branching is to keep a history of improvements that were obtained by branching on variables. We calculate a score for each variable based on the change in the objective for both directions after branching on them. A downside of pseudocost branching is that there is no history at the beginning of the problem. It usually takes time to collect enough observations for pseudocosts to work as desired.

5. Strong branching: The idea of strong branching is to test the candidates before branching on them. Full strong branching is a variation of the strong branching, where the constructed subproblems N^i_-, N^i_+ are solved to optimality for each candidate. Then a decision is made based on biggest improvement. This is a time-consuming strategy, usually only a partition of candidate subproblems is solved. Moreover, an iteration bound is applied when solving these problems. When these subproblems are not solved to optimality, the method is more time-efficient. Solving the subproblems in this way gives us a rough estimate about the improvement. Achterberg et al. [3] suggests that the number of dual simplex iterations in subproblems be limited to two times the number of average simplex iterations needed so far in the algorithm.

6. Reliability branching: Both pseudocost and strong branching rules have some advantages and disadvantages. Pseudocost branching starts poor, while the strong branching is a time-consuming approach. The reliability branching is a combination of these two branching rules, to benefit from their powerful features. This rule starts solving the subproblems with strong branching and then continues with pseudocost at the lower levels in B&B tree. A fixed reliability parameter is chosen before B&B is applied. Then, if the number of branchings on a variable is less than the reliability parameter, strong branching is applied; otherwise, pseudocost branching is applied to create subproblems.

7. Hybrid branching: Hybrid branching combines reliability branching and strong branching rules but uses a different measure than the reliability branching rule uses. Hybrid branching uses the depth of B&B tree as a measure. If a node is below a pre-selected level, pseudocost branching is applied. Otherwise, strong branching is applied.

8. General disjunctions: Alternatively, we can branch on general disjunctions in

the form given in (2.14). Karamanov and Cornuéjols [66] show that branching on a disjunction often performs better than branching on variables. As an example, they use Gomory cuts and show that branching on disjunctions produces smaller B&C trees compared to full strong branching on variables. They propose a heuristic to measure the disjunction quality. Although branching on disjunctions is not novel, their approach is a good indicator of how such a type of branching would be useful. The basic idea is to find a violated split disjunction that is used to create an intersection cut and then used for branching purposes.

A review of these methods can be found in Achterberg et al. [3] and Martin [76].

Achterberg et al. [3] show that there is a direct relation between pseudocost branching, strong branching, reliability branching, and a hybrid of pseudocost and strong branching. If we set the depth of the hybrid branching d and the reliability parameter of the reliability branching to 0, then these methods work exactly as pseudocost branching works. On the other hand, setting these parameters to infinity gives strong branching.

A discussion of branching methods for MISOCO

Branching rules from both MILO and MINLO literature are applicable to MISOCO problems. However, only some of these branching rules were applied to MISOCO problems in the literature. Drewes [41] uses general branching rules, such as index-based, random, most fractional, pseudocost, and combined fractional branching rules in their B&C algorithm to solve MISOCO problems. The same branching rules are also used by Góez [54] in a BCC framework. Muramatsu and Suzuki [83] use a simple branching rule in their B&B approach to solve max-cut problems. Vielma et al. [104] present a B&B algorithm for MISOCO, but they use CPLEX's default branching rules which were mainly developed for MILO.

Branching rules for MISOCO is an open research question. There is not a specific

branching rule aimed at general MISOCO problems in the literature.

2.2.2 Linear and conic cuts

Cutting plane methods use valid inequalities to improve the relaxation solution of optimization problems. Generating valid inequalities may improve the lower bound. B&C methods combine B&B and cutting plane methods. The former method partitions problems into subproblems, whereas the latter method improves the bound obtained from these subproblems by using valid inequalities. Generating efficient cuts may reduce the number of nodes in the B&B tree, which may decrease the solution time significantly. These kind of cuts are shown to be very powerful in MILO literature, hence they are available in all state-of-the-art solvers. Due to their similarities, such cuts are also studied for MISOCO, which is the main discussion of this section.

Most cuts in the MISOCO literature are extensions of disjunctive cuts for MILO problem which are developed by Balas [14]. In recent years, there is growing interest in the topic. Hence, most of the work mentioned in this section is recent. Recently, Kılınç-Karzan [68] establishes many aspects of improving cutting plane algorithms for MISOCO problems. A class of $\mathcal{K}$-minimal inequalities are introduced for general disjunctive conic optimization. The author also argues that these inequalities together with cone-implied inequalities are sufficient to describe the convex hull of mixed-integer conic optimization problems. This important result further implies the importance of cuts in a BCCP algorithm.

Çezik and Iyengar [35] extend valid inequalities for binary MILO to binary conic optimization problems. Their extension includes a more general cone, incuding the non-negative orthant, second-order, and semi-definite cones. One of the extensions presented in this paper is an extension of *Chvátal-Gomory cuts* to mixed binary conic optimization. Details of the cut can be found in Çezik and Iyengar [35]. This cut is a linear cut for MISOCO problems.

Atamtürk and Narayanan [10] present conic *mixed-integer rounding cuts* for MISOCO. Their approach is based on the reformulation of a SOC constraint with a *polyhedral second-order conic constraint* in a higher-dimensional space. By transformation, they obtain a SOC constraint that can also be written as a linear constraint. They generate their conic mixed-integer rounding cut on top of this constraint. Note that mixed-integer rounding cuts are linear in the higher-dimensional space, but they are nonlinear in the original space. In their numerical experiments, conic mixed-integer rounding cuts are added at the root node. They perform their experiments with CPLEX. They observe significant reduction in both time and number of nodes in the B&B tree after cuts are added to the problem.

Andersen and Jensen [9] show that *split cuts* and *intersection cuts* are the same in LO, but they are not necessarily equivalent in conic optimization. Split cuts are a subset of intersection cuts in conic optimization and are obtained from splits. Intersection cuts, on the other hand, are more general, and any inequality that has the same intersection with the given region can be an intersection cut.

Modaresi et al. [80] extend the discussion on split disjunctions to cross-sections of SOCs. They present a generalization of split, k-branch split, and intersection cuts for MINLO. They present split cuts for paraboloids and cones in their paper. In their follow-up paper [79], they show the relationship between split cuts and conic mixed-integer rounding cuts of Atamtürk and Narayanan [10]. They generalize the result of Atamtürk and Narayanan [10] and show that there is a link between conic mixed-integer rounding cuts (CMIRs) and split cuts for MISOCO. They also show that a single CMIR can be weaker than a single nonlinear split cut. Recently, Kılınç-Karzan and Yıldız [69] generalized split cuts, studied by Andersen and Jensen [9] and Modaresi et al. [79].

Yıldız and Cornuéjols [110] provide a unified representation of the general two-term disjunctions on a cross-section of the SOC. This study is an extension of the

general two-term disjunctions for the SOCs.

Stubbs and Mehrotra [97] extend the lift and project algorithm of Balas et al. [15] to mixed-integer convex optimization. Atamtürk and Narayanan [11] generalize the theory of lifting to conic integer optimization. They show how to derive conic valid inequalities by using conic inequalities of a mixed-integer conic optimization problem for its lower dimensional restrictions. A brief review of the methods mentioned in this section can be found at Benson and Sağlam [20].

Disjunctive conic and cylindrical cuts

Belotti et al. [16] focus on families of quadratic surfaces that have fixed intersections with two hyperplanes. They show that these families can be described by a single parameter. They aim to characterize the convex hull of union of the intersections of an ellipsoid with two half-spaces, when these intersections are disjunctive sets.

In a later paper [17], the same authors identify a procedure for constructing disjunctive conic (DCC) and cylindrical (DCyC) cuts. They prove that if there exists a cone $\mathcal{K}$ that has the same intersection with the boundary of disjunction $\mathcal{E}$, then the convex hull is the intersection of $\mathcal{E}$ with $\mathcal{K}$. Moreover, if it exists, such a cone is shown to be unique. The same conclusion is also obtained for cylinders.

Denote (Q, p, r) as the triplet to define the quadric $\mathcal{E} = \{x | x^\top Q x + 2p^\top x + \mathrm{r} \leq 0\}$. Suppose we have two half-spaces, $\mathcal{A} = \{x \in \mathbb{R}^n | a^\top x \geq \mathrm{a}\}$ and $\mathcal{B} = \{x \in \mathbb{R}^n | b^\top x \leq \mathrm{b}\}$. Let $\mathcal{A}^=$ and $\mathcal{B}^=$ denotes the hyperplanes that correspond to boundaries of half-spaces $\mathcal{A}$ and $\mathcal{B}$, respectively. A closed cone $\mathcal{K}$ is called *disjunctive conic cut* for the set $\mathcal{E}$ and the disjunctive set $A \cup B$ if

$$\mathrm{conv}(\mathcal{E} \cap (\mathcal{A} \cup \mathcal{B})) = \mathcal{E} \cap \mathcal{K}.$$

Theorem 2. *[54] Consider an ellipsoid $\mathcal{E}$ such as*

$$\mathcal{E} = \left\{x | x^\top Q x + 2p^\top x + \mathrm{r} \leq 0\right\}$$

denoted by triplet $\mathcal{E} = (Q, p, \mathrm{r})$. *Suppose hyperplanes* $\mathcal{A}$ *and* $\mathcal{B}$ *are parallel, such as* $b = a$. *Then a uniparametric family of quadrics* $\mathcal{Q}(\tau)$ *that has the same intersection with* $\mathcal{A}^{=}$ *and* $\mathcal{B}^{=}$ *is given by*

$$\begin{aligned} Q(\tau) &= Q + \tau a a^\top, \\ p(\tau) &= p - \tau \frac{\mathrm{a} + \mathrm{b}}{2} a, \\ \mathrm{r}(\tau) &= \mathrm{r} + \tau \mathrm{ab}. \end{aligned}$$

The family of quadrics $\mathcal{Q}(\tau)$ defines a cone if $Q(\tau)$ is a non-singular symmetric matrix with exactly one negative eigenvalue and

$$p(\tau)^\top Q(\tau)^{-1} p(\tau) - \mathrm{r}(\tau) = 0.$$

Rearranging the values, we reach a quadratic function. Let f be the numerator of this quadratic function. The quadric $\mathcal{Q}(\tau)$ can have different shapes based on roots of $f(\tau)$. Unless both roots make the denominator zero, either one of the roots gives us a cone. Disjunctive conic cuts can be generated in this way. See Góez [54] for a comparison between disjunctive conic cuts and mixed-integer rounding cuts. Belotti et al. [17] show some cases where disjunctive conic cuts are sufficient to reach integer optimal solution, whereas conic mixed-integer rounding cuts produce a non-integral solution. In Chapter 5, the effects of applying DCCs into asset allocation problems (AAP) is discussed in detail.

Discussion

Besides the cuts mentioned here, all linear cuts may be applied to linear constraints in MISOCO problems. See Conforti et al. [37] for a review of valid inequalities of MILO.

Despite the growing interest to generate valid inequalities to MISOCO problems, there is a gap between theoretical development and practical applications. Effects of the linear and conic cuts mentioned here are rather unknown. Although all these cuts

can improve the solution process, many aspects of their implementations are not clear. Open research topics of this subject are the cost of generating the cuts, their effects on the problem structure and the linear algebra of IPMs, their effects on warm-start strategies, and their efficiency under various branching rules. This dissertation takes a step toward closing the gap between theory and practice in this field. In Chapter 5, we show that carefully adding DCCs into AAPs can significantly reduce the number of nodes in a BCC tree.

Chapter 3

Warm-start of interior-point methods for second-order cone optimization via rounding over optimal Jordan frames

3.1 Introduction

Closely related SOCO problems appear at every node of B&B-based methods when solving MISOCO problems. Despite the efficiency of solving SOCO problems with IPMs, IPMs are not clearly dominating other methods as the default methods for solving subproblems when solving MISOCO problems inside off-the-shelf solvers. Indeed, polyhedral relaxations are still a prominent opponent of IPMs when solving such subproblems inside BB and BCC algorithms [31]. A variety of factors are in play here. First, subproblems obtained from the polyhedral approximation are linear optimization (LO) problems. Methodologies for solving MILO problems are well-developed; hence software that use these methodologies benefit from the well-established literature of MILO, such as cuts and heuristics. Second,

preprocessing for LO problems is far better developed than for SOCO. We have yet to see a significant work on preprocessing techniques for SOCO problems. Most importantly, warm-starting LO problems usually requires only a few dual simplex iterations in practice if the problem has changed slightly. This efficiency brings the total solution time down significantly for MILO problems. Despite recent interest in MISOCO and the existence of solvers deploying IPM-based MISOCO strategies [7, 46, 61, 64], there are still a need and considerable opportunities to improve the efficiency of these methods. Warm-starting is one of the main concerns for MISOCO, due to lack of efficient warm-starting methods. The issue of warm-starting has been carried over unsolved for three decades in using IPMs for solving LO problems.

Warm-starting of IPMs for SOCO is required to solve MISOCO problems more efficiently, and it remains an open research problem. Most of the previous work focused on perturbation-based changes [58, 95, 96, 107]. Only methods presented by Sivaramakrishnan et al. [94], Oskoorouchi and Mitchell [86] and Oskoorouchi et al. [87] can be considered as warm-starting of IPMs for modification-based changes, although their motivations are different from ours. Engau et al. [44] give straightforward extensions of some warm-start approaches that were originally developed for LO problems. Skajaa et al. [96] presented a simple warm-starting approach for homogeneous and self-dual IPMs when problems are perturbed by a limited magnitude. They use the optimal solution of the original problem and take a convex combination of it with the default initial point of the homogeneous IPM. The new point is then fed into the self-dual embedding IPM to warm-start the algorithm for a perturbed instance.

The main purpose of this chapter is to take a step toward filling the gap in the MISOCO literature to warm-start self-dual embedding IPMs. We present an efficient warm-start method that exploits the Jordan frame of a related instance, and efficiently warm-starts IPMs after adding linear cuts or after branching. Our proposed approach

is based on solving so-called primal and dual rounding problems for IPMs. This way, we generate a primal-dual feasible initial point. Then we use a new point, which is a convex combination of this primal-dual feasible point and an earlier IPM iteration of the original instance, to warm-start the new instance. Our way of using convex combinations is similar to Skajaa et al. [96]; however we have an extra step to find points that are primal and dual feasible, instead of using the previous optimal solution. We propose using this approach in solving SOCO subproblems inside BB and BCC trees when solving MISOCO problems. We measure the efficiency of our method by comparing the number of IPM iterations to solve new instances using our warm-start approach versus using the default initial point of IPMs, also called cold-start.

This chapter is structured as follows. In Section 3.2, relationships between rounding problems are presented. In Section 3.3, details of the warm-starting approach are presented. In Section 3.4, numerical experiments using randomly generated conic instances and the conic benchmark library (CBLIB) are provided. We give an overview of the chapter and present our conclusions in Section 3.5.

3.1.1 Self-dual embedding IPM

The warm-start approach we present in this chapter is based on self-dual embedding IPMs. For this reason, a primal-dual self-dual embedding IPM that uses Mehrotra's predictor-corrector method [77] is implemented in MATLAB. The implementation follows directions of Andersen et al. [8] and Sturm [99] for algorithmic choices.

A self-dual embedding IPM starts with a primal (P-SOCO) and dual (D-SOCO) pair. We introduce three auxiliary variables x_0, y_0, z_0 and write the following self-dual

problem:

$$
\begin{array}{lllllllll}
\text{minimize:} & \vartheta y_0 & & & & & & & \\
\text{subject to:} & & Ax & -bx_0 & +r_p y_0 & & & = & 0 \\
& -A^\top y & & +cx_0 & +r_d y_0 & -z & & = & 0 \\
& b^\top y & -c^\top x & & +r_g y_0 & & -z_0 & = & 0 \\
& -r_p^\top y & -r_d^\top x & -r_g x_0 & & & & = & -\vartheta \\
& x \in \mathcal{K},\ z \in \mathcal{K}^*,\ x_0 \in \mathbb{R}_+,\ y \in \mathbb{R}^m,\ z_0 \in \mathbb{R}_+, & & & & & & &
\end{array}
\tag{3.1}
$$

where

$$
r_p := \frac{bx_0 - Ax^{(0)}}{y_0^{(0)}}, \quad r_d := \frac{A^\top y^{(0)} + z^{(0)} - cx_0}{y_0^{(0)}}, \quad r_g := \frac{c^\top x^{(0)} - b^\top y^{(0)} + z_0^{(0)}}{y_0^{(0)}},
$$

$$
\vartheta = \frac{x^{(0)} z^{(0)} + x_0^{(0)} z_0^{(0)}}{y_0^{(0)}}.
$$

Default initial values $(x^{(0)}, y^{(0)}, z^{(0)})$ for a feasible IPM are

$$
x^{(0)} = \iota, \quad y^{(0)} = 0, \quad z^{(0)} = \iota, \quad x_0^{(0)} = 1, \quad y_0^{(0)} = 1, \quad z_0^{(0)} = 1,
$$

where $\iota^i = (1, 0, \ldots, 0)^\top \in \mathbb{R}^{n_i}$ for $i = 1, \ldots, k$ and $\iota = (\iota^1; \ldots; \iota^k) \in \mathbb{R}^n$. This initial solution is feasible for the self-dual embedding problem and is also on the central path for $\mu = 1$.

Observe that we can initialize the self-dual embedding IPM with any interior conic feasible solution for (3.1). This observation is key to our warm-start approach presented in the following section.

3.2 Rounding problems

Rounding problems introduced in Section 1.1.2 play a key role for the proposed warm-start approach. In this section, we derive relationships between rounding problems and their duals before presenting the warm-start approach. These relationships are shown in Figure 3.1 and summarized here.

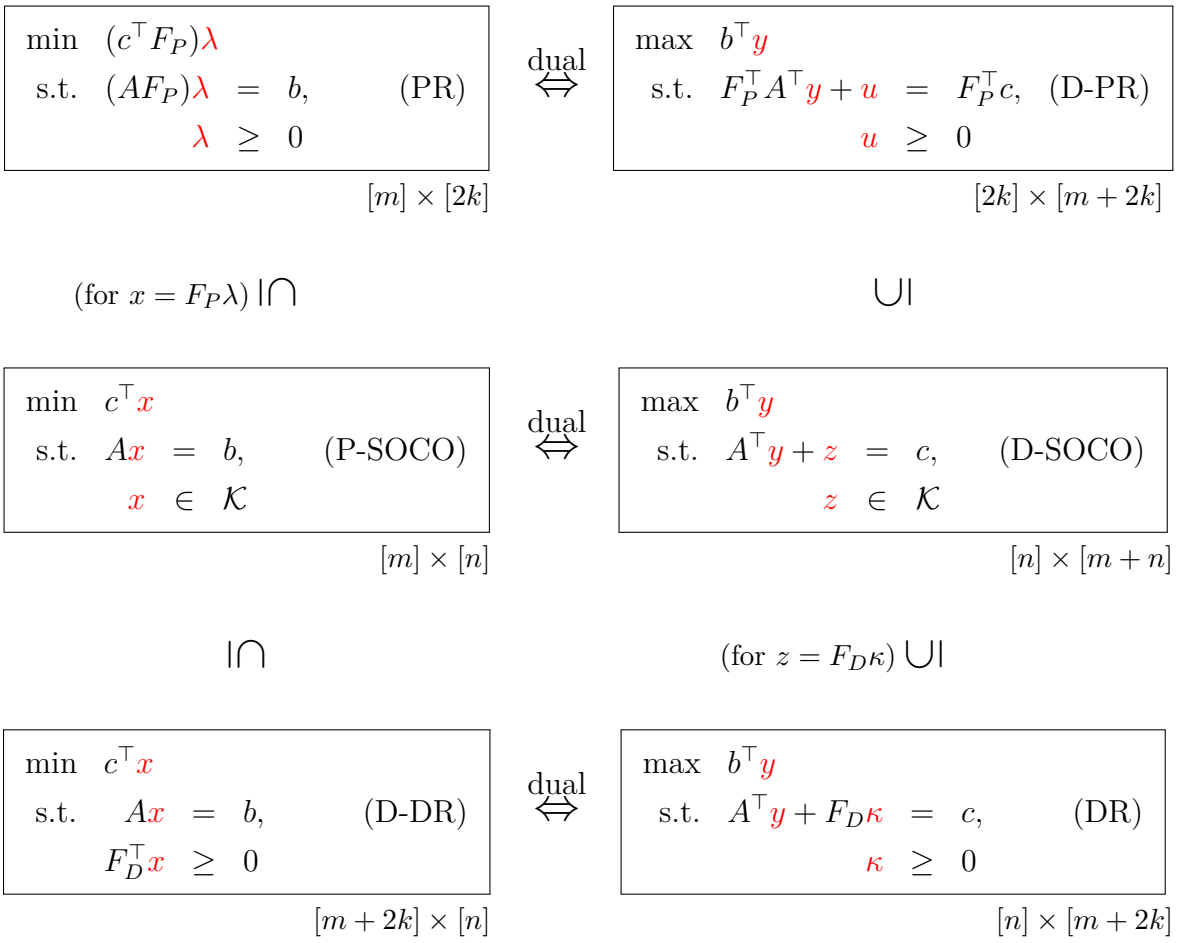


Figure 3.1: Feasibility and duality relationship between original SOCO problems and rounding LO problems.

We start with showing that any solution to (PR) is always a feasible solution to (P-SOCO).

Lemma 3. *Let F_P be the Jordan frame of a conic feasible solution $x \in \mathcal{K}$. Then $\tilde{x} = F_P\lambda$ is a feasible solution to (P-SOCO) for any λ that is feasible to (PR).*

Proof. Any feasible solution λ to (PR) satisfies $AF_P\lambda = b$ and $\lambda \geq 0$. In this case, $\tilde{x} = F_P\lambda$ satisfies $A\tilde{x} = A(F_P\lambda) = b$.

For conic feasibility, let us consider a single SOC $\mathbb{L}^{n_i}$. We have

$$\begin{aligned}\tilde{x}_i = f_{P_i}\lambda_i &= \begin{bmatrix} f^+_{P\,i} & f^-_{P\,i}\end{bmatrix}\begin{bmatrix}\lambda^+_i \\ \lambda^-_i\end{bmatrix} \\ &= \begin{bmatrix} 1/2 & 1/2 \\ \frac{x^i_{2:n_i}}{2\|x^i_{2:n_i}\|} & -\frac{x^i_{2:n_i}}{2\|x^i_{2:n_i}\|}\end{bmatrix}\begin{bmatrix}\lambda^+_i \\ \lambda^-_i\end{bmatrix} \\ &= \begin{bmatrix} \frac{\lambda^+_i+\lambda^-_i}{2} \\ \frac{\left(\lambda^+_i-\lambda^-_i\right)x^i_{2:n_i}}{2\|x^i_{2:n_i}\|}\end{bmatrix}.\end{aligned}$$

Since λ is feasible to (PR), using $\lambda^+_i \geq 0, \lambda^-_i \geq 0$ gives the conic feasibility:

$$\|\tilde{x}_{2:n_i}\| = \sqrt{\sum_{j=2}^{n_i} \frac{\left(\lambda^+_i - \lambda^-_i\right)^2 x^{i\,2}_j}{4\left(\|x_{2:n_i}\|\right)^2}} = \sqrt{\frac{\left(\lambda^+_i - \lambda^-_i\right)^2}{4}} \leq \frac{\lambda^+_i + \lambda^-_i}{2} = \tilde{x}^i_1.$$

Therefore, $\tilde{x}_i \in \mathbb{L}^{n_i}$. We get $\tilde{x}_i \in \mathbb{L}^{n_i}\ \forall i = 1, \ldots, k$, and therefore $\tilde{x} \in \mathcal{K}$.

Since $\tilde{x}$ satisfies both $A\tilde{x} = b$ and $\tilde{x} \in \mathcal{K}$, it is a feasible solution to (P-SOCO). □

Similarly, we can find a feasible solution to (D-SOCO) from any feasible solution to (DR). In the following lemma, we show the relationship between (P-SOCO) and (D-DR).

Lemma 4. *Let F_D be the Jordan frame of a conic feasible solution $z \in \mathcal{K}$ that is used to define* (D-DR)*. Then x is a feasible solution to* (D-DR) *for any x that is feasible to* (P-SOCO)*.*

Proof. Any feasible solution x to (P-SOCO) satisfies $Ax = b$. By definition of dual cones, we have $w^\top x \geq 0\ \forall w \in \mathcal{K}$. Since $z \in \mathcal{K}$ and hence every column in f_D is in $\mathcal{K}$, we get $F_D^\top x \geq 0$. Since x satisfies both constraints, it is a feasible solution to (D-DR). □

We combine results of these two lemmas in the following theorem, where we show the weak duality relationship between the primal and dual rounding problems.

Theorem 5 (Weak duality)**.** *Let $F_P, and F_D$ be the Jordan frames of a conic feasible primal and dual solution (x, y, z). Let λ^* and $y^*_{(DR)}$ be optimal solutions for* (PR) *and* (DR)*, respectively. Then $c^\top F_P \lambda^* \geq b^\top y^*_{(DR)}$.*

Proof. Lemma 3 shows that any feasible solution to (PR) corresponds to a feasible solution to (P-SOCO). Therefore, at optimality, we always have $c^\top F_P \lambda^* \geq c^\top x^*_{\text{(P-SOCO)}}$. Lemma 4 shows that any feasible solution to (P-SOCO) corresponds to a feasible solution to (D-DR). Therefore, at optimality we always have $c^\top x^*_{\text{(P-SOCO)}} \geq c^\top x^*_{\text{(D-DR)}}$. Because of strong duality of LO, at optimality for (DR) and its dual (D-DR) we have

$$c^\top x^*_{\text{(D-DR)}} = b^\top y^*_{\text{(DR)}}.$$

Combining all these implications gives

$$c^\top F_P \lambda^* \geq c^\top x^*_{\text{(P-SOCO)}} \geq c^\top x^*_{\text{(D-DR)}} = b^\top y^*_{\text{(DR)}}. \tag{3.2}$$

This completes the proof. □

Note that if F_P is the Jordan frame of an optimal solution $x^*_{\text{(P-SOCO)}}$ of (P-SOCO), then the equality $c^\top F_P \lambda^* = c^\top x^*_{\text{(P-SOCO)}}$ holds. Similarly, if f_D is the Jordan frame of an optimal solution $z^*_{\text{(D-SOCO)}}$ of (D-SOCO), then the equality $c^\top x^*_{\text{(P-SOCO)}} = c^\top x^*_{\text{(D-DR)}}$ holds.

Corollary 6. *If the objective value of problems* (PR) *and* (DR) *are equal to each other for a feasible solution, then all inequalities in* (3.2) *hold as equalities. In this case, an optimal solution of* (P-SOCO) *and* (D-SOCO) *can be obtained as $(x, y, z) = \left(F_P \lambda^*, y^*_{(DR)}, F_D \kappa^*\right)$ by solely solving rounding problems.*

Corollary 7. *If the problem* (DR) *is dual infeasible, then* (P-SOCO) *must be infeasible. This includes the case when* (DR) *is unbounded.*

These results can be also derived for the dual side. Note that we were able to derive these results while not using anything beyond weak duality about the duality properties of the original pair of SOCO problems.

3.3 Warm-starting

Warm-start for optimization methods plays a critical role in reducing solution time, especially in B&B and BCC methods, where related instances are solved consecutively. In fact, the popularity of the dual simplex method for solving MILO can be attributed to its efficiency in warm-start. By warm-start, we can think of using any information obtained from the original instance to solve a related instance. Although warm-start methods for simplex-like methods are limited only to using a previous optimal solution, we need to think beyond that for warm-starting IPMs. Using the optimal solution of the original instance for warm-starting IPMs directly is not a good idea. Such a point will be most likely be infeasible, which leaves only infeasible IPMs in play. Moreover, such a point will be on the boundary of the feasible region, which means IPMs are not directly applicable. Contrary to simplex-like methods, a well-centered point could work much better in practice, even if it is away from the previous optimal solution. This is the main reason why warm-starting for IPMs is often seen as unsuccessful or application-specific.

Researchers have tried various methods to design efficient warm-starting IPMs. Almost all approaches on this topic use either an optimal solution or a stored IPM iteration of the original problem and try to generate a good initial point. There are mainly two ways of approaching warm-starting for IPMs. The first is modifying the problems to prevent slow progress. These methods are also called shifted barrier methods, because they shift the boundaries of variables temporarily for faster progress [43]. The second is to use an intermediate procedure to obtain a relatively well-centered point and use it as an initial point to the IPM [92]. Researchers have tried various methods within this category, such as adding slack variables [44], generating a point by using a feasible point and the previous optimal solution [96], and using an exact primal-dual penalty method [21].

We present a warm-start approach that uses an intermediate procedure to formulate the self-dual embedding IPM with an initial point. Our procedure

consists of solving two auxiliary LO problems, which help us minimize the primal and dual infeasibilities, respectively. Then, similar to Skajaa et al. [96] we find a point as a convex combination of two points, in our case the optimal solution of the rounding LO problems and an IPM iterate of the original problem.

We present the steps of our proposed warm-start approach in the following subsections.

3.3.1 Solving rounding problems

Let us recall the primal (P-SOCO) and dual (D-SOCO) SOCO problems:

$$\begin{array}{lrcll} \text{minimize:} & c^\top x & & & \\ \text{subject to:} & Ax & = & b, & \text{(P-SOCO)} \\ & x & \in & \mathcal{K}, & \end{array} \qquad \begin{array}{lrcll} \text{maximize:} & b^\top y & & & \\ \text{subject to:} & A^\top y + z & = & c, & \text{(D-SOCO)} \\ & z & \in & \mathcal{K}. & \end{array}$$

Suppose we have an original problem in the form of (P-SOCO). Let (x^*, y^*, z^*) be a primal-dual optimal solution for the original problem. Based on the numerical values of x^* and z^*, each conic component of the solution can belong to one of the four classes: $\mathcal{B}, \mathcal{N}, \mathcal{R}, \mathcal{T}$ [102]. A cone i belongs to $\mathcal{B}$ if x^{*i} is inside the cone and $z^{*i} = 0$, and to it belongs $\mathcal{N}$ if z^{*i} is inside the cone and $x^{*i} = 0$. For the class $\mathcal{R}$, both x^{*i} and z^{*i} are on the boundary of the cone i and orthogonal to each other. Finally, for the $\mathcal{T}$ case, the sum $x^{*i} + z^{*i}$ is on the boundary of the cone i. If $\mathcal{T} \neq \emptyset$, then the optimal solution is not strictly complementary.

After identifying the classes of the cones, we derive the Jordan frames for both the primal and dual problems. If a cone belongs to the set $\mathcal{B}$, then we use the primal Jordan frame for the dual slack variable z^{*i} too. Similarly, if a cone belongs to the set $\mathcal{N}$, we use the dual Jordan frame for the primal variable x^{*i} too. If a cone is in $\mathcal{R}$, then the primal and dual Jordan frames are the same. We use this frame for F_P and F_D. If a cone is in $\mathcal{T}$, we use an earlier IPM iteration to choose a Jordan frame.

Denote F_P^* and F_D^* as primal and dual Jordan frames at an optimal solution (x^*, y^*, z^*).

Consider that the new instance is obtained after branching on a variable (say, x_j) such that

$$x_j \leq \lfloor x_j^* \rfloor$$

is added to the problem. We add a slack variable s to the problem to get the new constraint in the standard form, such as

$$x_j + s = \lfloor x_j^* \rfloor,\ s \in \mathbb{L}^1.$$

Denote

$$\bar{x} = \begin{bmatrix} x \\ s \end{bmatrix},\ \bar{y} = \begin{bmatrix} y \\ y_s \end{bmatrix},\ \bar{z} = \begin{bmatrix} z \\ z_s \end{bmatrix},\ \bar{A} = \begin{bmatrix} A & 0 \\ e_j^\top & 1 \end{bmatrix},\ \bar{b} = \begin{bmatrix} b \\ \lfloor x_j^* \rfloor \end{bmatrix},\ \bar{c} = \begin{bmatrix} c \\ 0 \end{bmatrix},\ \bar{\mathcal{K}} = \mathcal{K} \times \mathbb{L}^1,$$

where e_j is the j^{th} unit vector. The new primal-dual problems are

$$\begin{array}{rlcl} \text{minimize:} & \bar{c}^\top \bar{x} & & \\ \text{subject to:} & \bar{A}\bar{x} & = & \bar{b}, \\ & \bar{x} & \in & \bar{\mathcal{K}}, \end{array} \qquad \begin{array}{rlcl} \text{maximize:} & \bar{b}^\top \bar{y} & & \\ \text{subject to:} & \bar{A}^\top \bar{y} + \bar{z} & = & \bar{c}, \\ & \bar{z} & \in & \bar{\mathcal{K}}. \end{array}$$

After adding the slack variable into the primal and dual Jordan frames, F_P^* and F_D^*, we obtain Jordan frames for rounding problems. Now, we can replace $\bar{x}$ and $\bar{z}$ with their equivalent $\bar{F}_P\lambda$ and $\bar{F}_D\kappa$ in the primal and dual problems. This gives us primal and dual rounding problems for the new system, namely

$$\begin{array}{rlcl} \text{minimize:} & (\bar{c}^\top \bar{F}_P)\lambda & & \\ \text{subject to:} & (\bar{A}\bar{F}_P)\lambda & = & \bar{b}, \\ & \lambda & \geq & 0. \end{array} \qquad \begin{array}{rlcl} \text{maximize:} & \bar{b}^\top \bar{y} & & \\ \text{subject to:} & \bar{A}^\top \bar{y} + \bar{f}_D\kappa & = & \bar{c}, \\ & \kappa & \geq & 0. \end{array}$$

We solve these rounding problems for variables $(\lambda, \bar{y}, \kappa)$ to obtain the rounding solution of the system; that is, $(\bar{x}^*, \bar{y}^*, \bar{z}^*) = (\bar{F}_P\lambda^*, \bar{y}^*, \bar{f}_D\kappa^*)$. The primal rounding problem consists of m constraints and $2k$ variables, and the dual rounding problem consists of n constraints and $m + 2k$ variables. For a moderately sized MISOCO problem, sizes of the primal and dual rounding problems are considered to be small for any commercial off-the-shelf LO solvers. Solving the rounding problems take a fraction of a second for most cases in practice, so they are negligible compared to the time required to solve the SOCO problems.

3.3.2 Choosing a convex combination of solutions

Notice that our rounding solutions satisfy both $\bar{A}\bar{x}^* = \bar{b}$ and $\bar{A}^\top \bar{y}^* + \bar{z}^* = \bar{c}$ at optimality. Hence, $(\bar{x}^*, \bar{y}^*, \bar{z}^*)$ is a feasible primal-dual solution for the new problem, although it may not satisfy the complementarity condition. This solution is not suitable to start IPMs, since it is on the boundary of the feasible region.

Instead of using the rounding solution, we define an interior point as the convex combination of the rounding solution and an earlier IPM iteration of the original problem. Here, we know that the default initial point $(x, y, z) = (\iota, 0, \iota)$ is always interior feasible for the new problem after initializing the extra variables as $(s, y_s, z_s) = (1, 0, 1)$ due to the construction of the self-dual embedding model. We can find and choose a conic feasible IPM iteration ℓ, which is expressed as $(x^{(\ell)}, y^{(\ell)}, z^{(\ell)})$, and use it to generate a new initial solution for the IPM.

From a feasible IPM iteration ℓ, we generate an initial point $(\bar{x}^{(0)}, \bar{y}^{(0)}, \bar{z}^{(0)})$ such that

$$\bar{x}^0 = \alpha \bar{x}^* + (1-\alpha) \begin{bmatrix} x^{(\ell)} \\ 1 \end{bmatrix},$$
$$\bar{y}^0 = \alpha \bar{y}^* + (1-\alpha) \begin{bmatrix} y^{(\ell)} \\ 0 \end{bmatrix},$$
$$\bar{z}^0 = \alpha \bar{z}^* + (1-\alpha) \begin{bmatrix} z^{(\ell)} \\ 1 \end{bmatrix}.$$

Using an earlier IPM iteration will provide a better centered interior point, although primal-dual infeasibilities may be large. On the other hand, using a later IPM iteration is more likely to give us a smaller μ parameter and smaller primal-dual infeasibility, but it may lead to slow progress due to the risk of being closer to the boundary. From a practical point of view, one needs to consider the trade-off between closeness to optimality versus closeness to boundary when choosing which IPM iteration is used. Based on our numerical experiments, choosing an iterate from one-fourth to one-half of the original IPM iterations is a

safe choice. This choice is usually close to the central path for the new problem and provides a smaller primal-dual infeasibility for our purposes. A moderate choice of α usually works well. We observed that $\alpha = 0.6$ works consistently well across all tests.

3.3.3 Initialization

After completing the previous steps, now we have reached the point that the self-dual embedding model can be initiated by $(\bar{x}^{(0)}, \bar{y}^{(0)}, \bar{z}^{(0)})$. Conic infeasibilities can be fixed by increasing the leading variables with a small magnitude. At the beginning of the IPM steps, the generated initial point is evaluated for centrality and a corrector step is taken if needed. Variables inside the self-dual embedding model are initialized as $(x_0^{(0)}, y_0^{(0)}, z_0^{(0)}) = (1, 1, 1)$.

3.3.4 Solution approach

Our proposed warm-start method has the ability to frequently detect optimality and infeasibility right after solving rounding problems. There are four outcomes in total for each instance in our solution approach:

- Immediately infeasible (II) when the (DR) problem is dual infeasible.
- Immediately optimal (IO) when the objectives of the (PR) and (DR) are the same.
- Warm-start (WS) applicable when rounding problems are feasible but do not prove optimality or infeasibility.
- Cold-start (CS) restricted when rounding problems are infeasible.

We use the following steps as our solution approach after solving the original problem and obtaining the optimal Jordan frames:

1. Solve the dual rounding problem.

    - If the (DR) problem is dual infeasible, then conclude that the (P-SOCO) problem is infeasible and return II, see Corollary 7.
    - If the (DR) problem is feasible, then continue.

2. Solve the primal rounding problem.

    - If the (PR) problem is feasible and the objective values of the rounding problems are equal to each other, then we can conclude that the rounding solution is optimal and return IO, see Corollary 6.
    - Else, if the (PR) problem is feasible, choose an early iterate of the IPM iterations based on the number of IPM iterations of the original problem, and use a pre-defined α value to generate an initial point. We can warm-start the IPM from the generated point.
    - If the (PR) problem is infeasible, then use the default initial point of the self-dual embedding system with cold-start.

3.4 Numerical experiments

In this section, we provide details of experiments conducted to test our warm-start approach on a variety of problems. We start with a description of the method, and then provide numerical results.

3.4.1 Methodology

We use a similar approach to measure efficiency of the proposed warm-start strategy as Engau et al. [44] and Skajaa et al. [96]. We measure the ratio of warm-start iterations to cold-start iterations, and then find the geometric mean that covers all

test problems. We add one to each metric to be able to include cases where an optimal solution is obtained after solving rounding problems. The ratio for each problem is

$$\mathfrak{R}_i = \frac{(\#\text{ of IPM iterations with warm-start for Problem } i) + 1}{(\#\text{ of IPM iterations with cold-start for Problem } i) + 1}.$$

The metric for measuring the efficiency of the method is the geometric mean,

$$\mathfrak{G}_I = \sqrt[k]{\mathfrak{R}_1 \mathfrak{R}_2 \cdots \mathfrak{R}_k}.$$

We mainly use the number of IPM iterations to measure the performance of the warm-start, because solving rounding problems take a negligible amount compared to solving SOCO instances. The number of IPM iterations yields approximately the same ratio, which we found sufficient for our purposes here. Any value of $\mathfrak{G}_I$ less than 1 indicates that warm-start is more efficient than cold-start.

The warm-start approach is implemented in MATLAB on top of a self-dual embedding SOCO solver that we implemented. We used the primal simplex method of IBM ILOG CPLEX 12.7.0.0 to solve the rounding problems. The Conic Benchmark Library (CBLIB) problems and their random fixings are solved at the root node and then two subproblems are generated. Then both subproblems are solved with cold-start and warm-start IPMs. We limited time to 2 hours and memory to 8GB for the tests, and we discarded any results that took more than 50 IPM iterations.

As a rule of thumb, we choose one-fifth of total IPM iterations. We used $\alpha = 0.7$ for all experiments which worked well in practice.

3.4.2 Performance of warm-start for various branching variable types

We can categorize variables inside SOCO problems into three types: Non-negative variables, leading variables, and in-cone variables. By definition, non-negative variables belong to $\mathcal{L}^1$ and do not appear in any other cones. Leading variables are

the first index in an SOC that has two or more elements. In-cone variables are the remaining ones, which appear inside cones that have at least two elements. Only a few problem in the CBLIB library have integer variables that appear as leading variables of SOCs. No problems in CBLIB have an integer variable inside SOCs. For these reasons, we generated some variations of CBLIB test problems where we branch on leading and in-cone variables, even if they are not originally specified as integers. We discarded instances that hit the time or the iteration limit from the results.

Feasibility of the rounding problems depends heavily on which type of variable is used for branching. See Table 3.1 for the distribution of instances into methods we followed for warm-start based on variable type. We were able to use our warm-start approach (either IO, II, or WS) for 1829 out of 2539 problems when we branched on non-negative variables. This number is 1794 out of 2614 when we branched on leading variables and 1641 out of 2603 when we branched on in-cone variables. These results show that our warm-start approach works best for branching on non-negative variables for feasible cases.

Instances where we branched on leading and in-cone variables often led to infeasible rounding problems. The reason for this is the nature of our warm-start. By fixing the optimal Jordan frames, we were assuming that the new optimal solution would have the same or a similar Jordan frame, so that we could get closer to the solution by using a convex combination. However, the Jordan frame that corresponds to the cone in which the branching variable appears will most likely change after branching, which gives us infeasible rounding problems. Our use of rounding problems corresponds to outer-approximation for the dual side. Adding a new Jordan frame to the problem may make the primal rounding problem feasible, especially for in-cone variable branching, but it is hard to identify which frame will be feasible.

Table 3.2 shows efficiency of the warm-start approach in terms of geometric mean.

Type	Status	II	IO	WS	CS	Total
Non-negative		3	8	1819	709	2539
	Feasible	-	8	1819	709	2536
	Infeasible	3	-	-	0	3
Leading		2408	5	177	24	2614
	Feasible	-	5	177	21	203
	Infeasible	2408	-	-	3	2411
In-cone		1315	1	638	649	2603
	Feasible	-	1	638	521	1160
	Infeasible	1315	-	-	128	1443

Table 3.1: Distribution of instances based on variable type and problem status.

This table includes only cases where the actual warm-start on IPM is applied. The results show that we solved the instances in about 81% of the iterations compared to cold-start when we branch on non-negative variables. It means approximately 19% savings in terms of IPM iterations. The geometric mean $\mathfrak{G}_I$ is 51% for leading variables and 79% for in-cone variables, which corresponds to 49% and 21% reduction in IPM iterations, respectively. For a warm-start method for modification-based changes, this is a significant improvement.

When all cases (IO, II, and CS) are included, there is a huge improvement in terms of IPM iterations because of early detection. Table 3.3 shows the geometric means after including IO and II cases, and including CS cases. On average, the warm-start approach that includes early detection reduces the number of IPM iterations by 15%, 88%, and 70% for non-negative, leading, and in-cone branching cases. Note that these averages are heavily skewed due to infeasible instances. On average, the warm-start approach reduces the number of IPM iterations needed by 69% when all cases and

Type	# Prob.	$\mathfrak{G}_I$
Non-negative	1819	0.8124
Leading	177	0.5178
In-cone	638	0.7918
Total	2634	0.7833

Table 3.2: Geometric mean of the ratio of warm-start iterations to cold-start iterations among only warm-started instances.

all 7756 problems are included.

	WS, IO, II		All	
Type	# Prob.	$\mathfrak{G}_I$	# Prob.	$\mathfrak{G}_I$
Non-negative	1830	0.8029	2539	0.8536
Leading	2590	0.1224	2614	0.1248
In-cone	1954	0.2046	2603	0.3039
Total	6374	0.2459	7756	0.3157

Table 3.3: Geometric mean of the ratio of warm-start iterations to cold-start iterations among all instances.

The benefit obtained from warm-starting could be tuned for a problem-specific class by changing which IPM iteration and α value to use for warm-starting. When working on specific instances, it is possible to reduce the number of IPM iterations up to 60% by fine-tuning the warm-start parameters.

Table 3.4 shows geometric means for problem types. It is apparent that problem type plays a significant role in determining the benefit obtained from the warm-start. For problem type `ck`, warm-start reduces the number of IPM iterations by half for

leading and in-cone branching. A similar effect can be seen for `sssd` problems, where the geometric mean is around 67%. On the other hand, for some instances the number of IPM iterations are higher compared to cold-start (for example, `pp` and `estein`). This table shows the need for tuning parameters for a specific problem instance. Since problem shape and behavior plays a major role on the central path, choosing a general IPM iteration and α is difficult. Despite this, the benefit from the overall warm-start approach is visible. Moreover, Table 3.4 clearly shows that feasibility of instances after branching on variable type depends on the problem type. Notice that none of `shortfall` instances were feasible after branching on leading variables.

	Non-negative		Leading		In-cone		Total	
Pr.Type	# Prob.	$\mathfrak{G}_I$	# Prob.	$\mathfrak{G}_I$	# Prob.	$\mathfrak{G}_I$	# Prob.	$\mathfrak{G}_I$
ck			158	0.4904	85	0.4797	243	0.4866
classical	555	0.8195			104	0.8263	659	0.8205
estein	14	0.9430	1	1.0000	9	1.0216	24	0.9741
pp	6	1.5309	5	1.3271			11	1.4347
robust	411	0.7747			21	0.9725	432	0.7833
shortfall	797	0.8260			399	0.8625	1196	0.8380
sssd	28	0.6933	10	0.6312	14	0.6674	52	0.6740
turbine	8	0.8256	3	0.7884	6	0.7700	17	0.7990
Total	1819	0.8124	177	0.5178	638	0.7918		0.7833

Table 3.4: Geometric means for warm-started instances classified by CBLIB problem type.

3.4.3 Comparison to cold-start and other warm-start methods

As shown in Section 3.4.2, our warm-start approach provides two benefits. The first is when we solve the problem right after solving rounding problems, and the second is choosing a different initial point for a self-dual embedding system if rounding problems provide feasible solutions. Figure 3.2 shows comparison of warm-start iterations to cold-start iterations. As seen from the scatter plot with jitter, we have a significant advantage when the rounding problems are feasible. The median line indicates that the number of warm-start IPM iterations is equal to 74.41% of number of IPM cold-start iterations on average.

We compared our method to the warm-start approach of Skajaa et al. [96]. Their warm-start approach performs better than the cold-start for our test problems despite being originally developed for perturbation-based changes. They use a ratio of 0.99 (previous optimal) to 0.01 (default initial point) to find an initial point for perturbation-based changes, but this choice performs poorly for most test problems because the problems change more significantly after branching than do small perturbations on the problem parameters. Therefore, we tried different values and decided to use parameter values of 0.15 (previous optimal) to 0.75 (default initial) for comparison. These values prevent initial points being too close to the boundary of the feasible region. Since we used cold-start for cases where the rounding problems are infeasible, we compare our method to theirs for only cases where we used an initial point to warm-start. Figure 3.3 compares the number of IPM iterations to Skajaa et al.'s warm-start method as a scatter plot with jitter. As shown in Figure 3.3, our method performs better than the warm-start method of Skajaa et al. on average. The median line indicates that the number of warm-start IPM iterations is equal to 82.52% of the number of IPM iterations in Skajaa et al. on average.

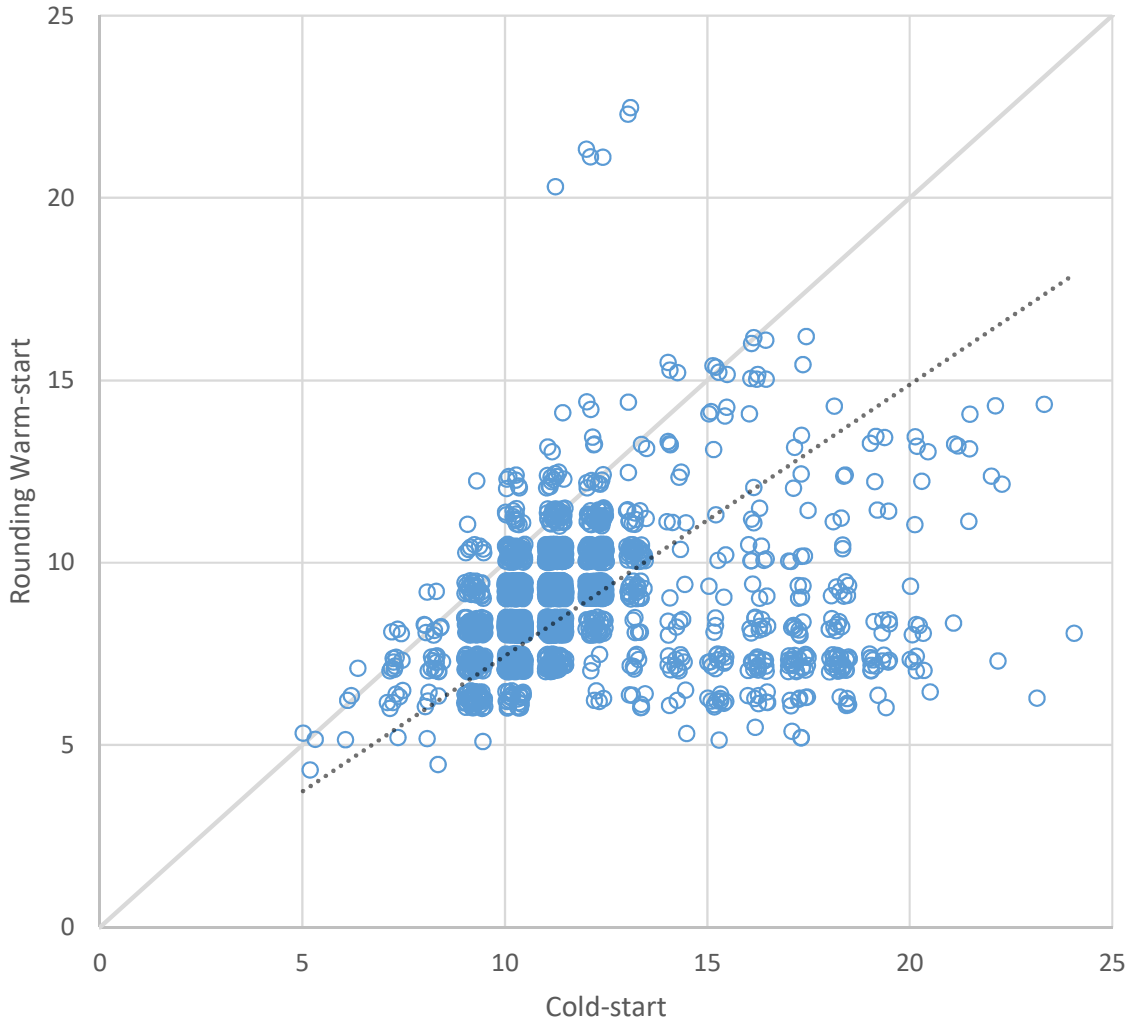


Figure 3.2: Comparison of warm-start IPM iterations versus cold-start IPM iterations for feasible instances.

3.4.4 Effect of warm-starting for infeasible cases

We see a huge benefit of solving rounding problems for infeasible cases. To the best of our knowledge, there is no warm-start method for infeasible cases in the literature. Our warm-start procedure is able to identify infeasible cases before solving the new instance for a majority of instances we looked at. For non-negative variable branching

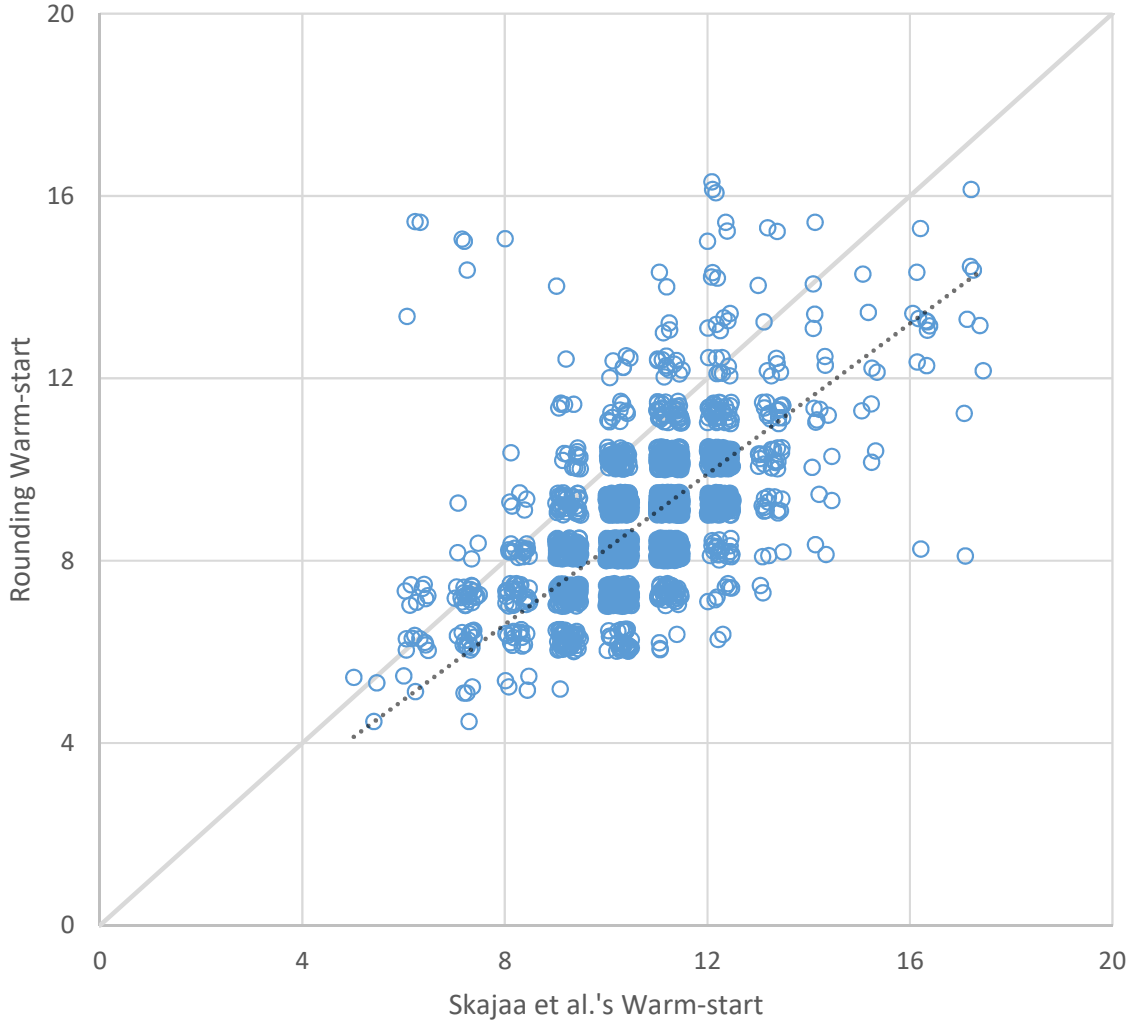


Figure 3.3: Comparison of IPM iterations of our approach versus warm-start of Skajaa et al. on feasible instances.

3 out of 3, for leading variable branching 2408 out of 2411, and for in-cone variable branching 1315 out of 1443 instances are concluded to be infeasible right after solving the rounding problems, which is approximately 96.60% of all infeasible instances. This means a huge saving in terms of solution time and IPM iterations.

3.5 Conclusions and future work

In this chapter, we introduced a novel warm-start approach for self-dual embedding IPMs to solve SOCO problems that appear in a BCC tree when solving MISOCO problems. To our knowledge, this is the first study about warm-starting SOCO instances after a modification-based change of the problem instance. Moreover, it is the first study on providing an earlier detection for infeasible cases before starting IPM iterations. Such an approach is not even available for LO problems.

In our experiments, we demonstrate the performance of our approach on the CBLIB test set. We are able to solve test problems taken from CBLIB using around 78% of the total IPM iterations that were required to cold-start on feasible instances. The ability of the warm-start approach to identify optimal and infeasible cases is also very significant. We are able to identify the new optimal solution in a few instances when branching on non-negative variables just by solving two LO rounding problems. In addition, we are able to identify infeasible primal SOCO instances in 96.60% of all infeasible cases. Improvement of such a magnitude is quite uncommon for a warm-start approach to IPMs.

Our approach is limited to the cases where rounding problems are feasible or provide a useful conclusion. Although our approach performs well on infeasible instances to detect infeasibility, the primal rounding problem gets infeasible for a significant number of cases. Such cases might be addressed in the future with an alternative approach of initialization of self-dual embedding IPMs.

There are a few open questions for future studies. First of all, we have yet to try this warm-start approach in a full BCC framework. Only minor changes are needed to deploy our proposed warm-start approach after a conic cut is added to the problem. Moreover, we can use the rounding problems for pruning by bound inside a B&B tree. If we have a dual feasible solution for (D-SOCO) with an objective value that is worse than the current incumbent objective in the tree, then we can prune the node by bound because it cannot yield a better solution due to the weak

duality in SOCO. So if the optimal value of the (DR) problem is greater than the objective of the incumbent solution, we can draw the same conclusion without solving the node. A final open question is about warm-starting of infeasible instances. The default initialization of the self-dual embedding framework is biased towards feasible instances. When infeasibility is suspected, it may be advantageous to initialize the model differently.

Chapter 4

The first heuristic specifically for mixed-integer second-order cone optimization

4.1 Introduction

Primal heuristics are one of the most important elements of search tree methods. Despite the fact that primal heuristics are not exact methods, their contribution to the efficiency of commercial solvers is significant. Therefore, commercial MILO solvers are packed with heuristics [7, 46, 61, 64].

Development of solution methodologies for MISOCO problems is an active research area. As MISOCO is a generalization of MILO, researchers mainly focus on translating existing MILO techniques to MISOCO. The main reason behind the popularity of MISOCO problems is twofold. The first reason is the availability of MISOCO formulations for a variety of problems from different sectors, such as portfolio optimization problems [20, 32] and option pricing problems [88] from finance, the turbine balancing problem [41] from energy, and the stereotactic surgery treatment planning with isocenter selection problem [53] from healthcare.

Another reason is the existence of efficient solution methodologies to solve underlying SOCO subproblems.

The ongoing research has been focused on generating valid inequalities for MISOCO problems [9–11, 16–18, 35, 68, 80, 110]. Despite recent advances in warm-start of MISOCO problems [33], vital elements of a full BCC framework are still missing, such as pre-processing and primal heuristics. Notably MILO has a remarkably rich literature on this topic. Since there is no available study on heuristics specific to MISOCO, we give a brief review of the MILO and MINLO literature.

Primal heuristics play an important role in state-of-the-art MILO solvers. Their main role is to provide an upper bound early in the search tree. This upper bound proves feasibility of the problem and can help reduce search tree size significantly by pruning nodes by bound early [4]. There are three types of primal heuristics. The first set of heuristics is called *diving heuristics*, which dive inside the search tree with the aim of finding a feasible MILO solution as quickly as possible. The second set of heuristics is called *rounding heuristics*, where the aim is to generate a feasible solution by rounding a fractional LO solution. The third set of heuristics is called *improvement heuristics*, where one or more primal feasible solutions are used to construct a better feasible solution. For an extensive discussion, see Achterberg [1] and Berthold [23].

Fischetti et al. [48] propose a heuristic called feasibility pump (FP). In simple terms, FP generates two sets of solutions: the first set consists of points that satisfy feasibility constraints except integrality, whereas the second set consists of points that are integer but possibly infeasible. These solutions are generated consecutively by using rounding and then solving an auxiliary problem. This heuristic is shown to be effective for binary MILO problems. FP's efficiency has led to several variations of the heuristic are discussed in the literature. Bertacco et al. [22] work on improving the efficiency of FP for general MILO problems. Moreover, they provide a restart method to prevent cycling. Achterberg and Berthold [2] propose a variation of FP

that takes the objective function into consideration when finding solutions. Their aim is to find a better feasible solution for MILO instead of an arbitrary one, and their modification is shown to be more effective for the majority of the test problems in terms of providing a better bound. Bonami et al. [30] and Bonami and Gonçalves [27] extend FP for convex MINLO problems in different ways. Bonami and Gonçalves [27] discussed extensions of several primal heuristics for convex MINLO problems and showed that variants of FP can provide a solution for 93—94% of instances. Primal heuristics for MINLO problems help them to reduce the total solution time about 11% on average. Finally, Berthold [24] gives an extensive review of heuristics for MINLO and presents a variation of FP for non-convex MINLO problems.

Berthold et al. [25] extend the SCIP solver to solve mixed-integer quadratic optimization (MIQO) problems by using two simple primal heuristics. They compare SCIP against other solvers on MIQO problems, where some of them are MISOCO instances available in the CBLIB [50] test set.

It is apparent that many of the available MILO heuristics can be used for MISOCO problems. However, MISOCO formulations allow us to explore further heuristics that are specific to MISOCO. To our knowledge, there is no research on development of heuristics for general MISOCO problems.

The purpose of this chapter is to present novel heuristics that are specific to MISOCO problems to generate feasible solutions in BCC search tree. We consider a general MISOCO formulation, where the Jordan frames of the SOCO subproblem can be obtained easily. These Jordan frames are used to create linear constraints that enforce conic feasibility in a limited feasible region. By solving the rounding MILO problems, it is possible to generate a feasible solution for many problems in practice. Then the generated solution can be improved by changing continuous variables. This way, we can generate a feasible solution even at the root node of a BCC search tree, which helps prune more nodes, prove feasibility, and provide an upper bound to estimate the optimality gap throughout the search. Our method consists of two

parts: in the first part (rounding), we solve several SOCO problems to construct a feasible MISOCO solution and in the second part (improvement), we use this solution to improve the solution.

The rest of the chapter is structured as follows: In Section 4.2, we give descriptions of conic rounding heuristics and their translation to quadratic optimization formulations. This is followed by the results of the numerical experiments on CBLIB test problems in Section 4.3. Finally, summary of the chapter, implications, and future research directions are discussed in Section 5.5.

4.2 Conic rounding heuristics

The main purpose of conic rounding heuristics is to provide a feasible solution for the original MISOCO problem. We can write a MISOCO in the standard form of

$$\begin{aligned} \text{minimize:} \quad & c^\top x \\ \text{subject to:} \quad & Ax = b, \\ & x \in \mathcal{K}, \\ & x_j \in \mathbb{Z} \qquad \forall j \in J \subseteq N, \end{aligned} \tag{4.1}$$

where $A \in \mathbb{R}^{m\times n}$ is a full row rank matrix, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $N = \{1, 2, \ldots, n\}$, $\mathcal{K}$ is the Cartesian product of SOCs of various dimensions — that is, $\mathcal{K} = \mathbb{L}^{n_1} \times \mathbb{L}^{n_2} \times \cdots \times \mathbb{L}^{n_k}$, where $\mathbb{L}^{n_i} = \{x^i \in \mathbb{R}^{n_i} \mid x^i_1 \geq \|x^i_{2:n_i}\|\}$, for $i = 1, \ldots, k$, with $\sum_{i=1}^k n_i = n$ for $x = \left((x^1)^\top, (x^2)^\top, \ldots, (x^k)^\top\right)^\top$ — and $x^i \in \mathbb{R}^{n_i}$. We begin this section with preliminaries on Jordan algebra and rounding problems. We use these components to describe four heuristics in the following subsections: the primal rounding heuristic, the dual rounding heuristic, the primal-dual rounding heuristic, and a hybrid heuristic.

4.2.1 The primal rounding heuristic

The main difficulties of finding a feasible solution for a MISOCO comes from two sources: integer variables and conic constraints. To deal with the latter source, one

can use a restricted problem of the original instance by using a set of feasible Jordan frames. Primal rounding (PR) with integrality constraints can be used to generate feasible solutions to the original MISOCO instance. This idea is the underlying foundation of the *primal rounding heuristic*, where we solve a series of mixed-integer primal rounding and auxiliary SOCO problems to generate a feasible solution.

The primal rounding heuristic starts with an optimal solution of the continuous relaxation of MISOCO. The Jordan frame that corresponds to this solution is stored for the iterative steps. Then we solve the mixed-integer primal rounding problem (MIPR), which is written as follows:

$$\begin{aligned} \text{minimize:} \quad & c^\top x \\ \text{subject to:} \quad & Ax &&= b, \\ & x &&= F^*\lambda, \\ & x_1^i &&\geq 0, && i \in 1, \ldots, k \\ & x_j &&\in \mathbb{Z}, && j \in J \subseteq N \\ & \lambda &&\in \mathbb{R}_+^{2k}. \end{aligned} \tag{MIPR}$$

If the (MIPR) problem is feasible, then its solution x^* is a feasible solution to the original MISOCO problem as well. This solution can be further improved by fixing the integer variables and solving the remaining SOCO problem to optimality. This problem is called fix-and-relax (FR) and is written as follows:

$$\begin{aligned} \text{minimize:} \quad & c^\top x \\ \text{subject to:} \quad & Ax &&= b, \\ & x_j &&= x_j^* && \forall j \in J \subseteq N, \\ & x &&\in \mathcal{K}. \end{aligned} \tag{FR}$$

After solving (FR), we can update the best known upper bound. Moreover, the optimal solution $\bar{x}$ of (FR) is used to obtain a new Jordan frame. This way new solutions, which are obtained in consecutive iterations, are guaranteed to be bounded by this solution.

A penalty problem is solved to generate more Jordan frames for the problem in each iteration regardless of whether (MIPR) is feasible. Denote $\hat{F} = F^+ - F^-$ and

the penalty problem is written as follows:

$$\begin{aligned} \text{minimize:} \quad & \varphi \tfrac{c^\top}{\|c\|} x + (1-\varphi) \left\| \hat{F}^\top x \right\| \\ \text{subject to:} \quad & Ax = b, \\ & x \in \mathcal{K}, \end{aligned}$$

where φ represents the tradeoff between the original objective and the penalty term. The reason for using the penalty term $\hat{F}^\top x$ is to obtain a different solution and a Jordan frame. The term $\|\hat{F}^\top x\|$ reaches its minimum value 0 if x is orthogonal to $\hat{F}$ and to its maximum value if the solution of the penalty problem x^p is equal to the (MIPR) solution. Because the transformation $x = F_P^\top \lambda$ and the constraint $\lambda \geq 0$ are used as a way to enforce conic feasibility and because new Jordan frames could be added to F, obtaining a larger feasible region is vital by obtaining different Jordan frames. Minimizing this term enlarges the feasible region for (MIPR) in consecutive iterations. The penalty term could be 1-norm, 2-norm, or infinity-norm and can be solved as a SOCO. The objective function of the penalty problem is similar to the objective proposed in [2], a penalty problem with a tradeoff between maximizing the feasible region and minimizing the original objective function. After the penalty problem is solved, (MIPR) is solved again with new Jordan frames, and the loop continues until a predefined termination criteria is reached, such as iteration or gap. All Jordan frames obtained in consecutive iterations are collected in a pool. Since the solution of the penalty problem is requested to be different from existing solutions, all Jordan frames can be added to the problem. The general form of the penalty problem (PEN) is written as follows:

$$\begin{aligned} \text{minimize:} \quad & \varphi \tfrac{c^\top}{\|c\|} x + (1-\varphi) \sum_\ell \left\| \hat{F}_\ell^\top x \right\| \\ \text{subject to:} \quad & Ax = b, \\ & x \in \mathcal{K}, \end{aligned} \tag{PEN}$$

where $\hat{F}_\ell$ is the penalty term that corresponds to the ℓ^{th} Jordan frame in the pool.

When (MIPR) is infeasible, it means that there are no feasible points in the restricted region defined by $x = F^*\lambda$. One can add more Jordan frames to F, which

is equivalent to expanding the feasible region of (MIPR). In theory, one can add infinitely many unique Jordan frames to (MIPR) and solve the original MISOCO problem as a series of (MIPR) problems. Our aim is to keep this number at a reasonable level and still be able to produce feasible results. As shown in Section 4.3, we rarely need more than a few frames even for larger cones. In fact, three Jordan frames are enough to obtain a feasible solution for 98% of all test instances, so the generated Jordan frames are far from being an inner-approximation of SOCs.

An overview of the flow of the primal rounding heuristic is given in Figure 4.1. In the figure, C represents conic feasible solutions, I represents integer feasible solutions, and IC represents both integer and conic feasible solutions. Potential solution outcomes are indicated on the top right corner of blocks. The heuristic steps are given in Algorithm 4.

Example

We provide a numerical example in this subsection to describe how the primal rounding heuristic works in practice. Consider the following optimization model:

$$\begin{aligned} \text{minimize:} \quad & 2x_1 + x_2 - 2x_3 \\ \text{subject to:} \quad & 10\,x_1 + x_2 = 19, \\ & (x_1, x_2, x_3) \in \mathbb{L}^3, \\ & x_1, x_2 \in \mathbb{Z}. \end{aligned}$$

Following are the steps of the primal rounding heuristic, which are illustrated in Figure 4.2. Numerical values are given up to three digits of precision.

1. Solve SOCO subproblem:

$$\begin{aligned} \text{minimize:} \quad & 2x_1 + x_2 - 2x_3 \\ \text{subject to:} \quad & 10\,x_1 + x_2 = 19, \\ & (x_1, x_2, x_3) \in \mathbb{L}^3. \end{aligned}$$

   The solution of the SOCO subproblem is $x^s = (1.991, -0.907, 1.772)$. For illustration, see Figure 4.2a.

2. Obtain the Jordan frames. The F matrix is

$$F = \begin{bmatrix} 0.5 & 0.5 \\ -0.228 & 0.228 \\ 0.445 & -0.445 \end{bmatrix}.$$

3. Solve the (MIPR) problem:

$$\begin{aligned} \text{minimize:} \quad & 2x_1 + x_2 - 2x_3 \\ \text{subject to:} \quad & 10\, x_1 + x_2 = 19, \\ & x_1 = 0.5\lambda_1 + 0.5\lambda_2, \\ & x_2 = -0.228\lambda_1 + 0.228\lambda_2, \\ & x_3 = 0.445\lambda_1 - 0.445\lambda_2, \\ & x_1 \geq 0, \\ & x_1, x_2 \in \mathbb{Z}, \\ & \lambda_1, \lambda_2 \in \mathbb{R}_+. \end{aligned}$$

(MIPR) is infeasible at this step. Supposing that we have budget for iteration, we continue. For illustration, see Figure 4.2b.

4. Solve the penalty problem (PEN) for $\varphi = 0.5$:

$$\begin{aligned} \text{minimize:} \quad & \varphi(x_1 - x_3) + (1 - \varphi)|x_3| \\ \text{subject to:} \quad & 10\, x_1 + x_2 = 19, \\ & (x_1, x_2, x_3) \in \mathbb{L}^3. \end{aligned}$$

The optimal solution of (PEN) is $x^p = (1.745, 1.553, 0.795)$. For illustration, see Figure 4.2c.

5. Obtain the Jordan frames. The F matrix is updated:

$$F = \begin{bmatrix} F^+_{(1)} & F^-_{(1)} & F^+_{(2)} & F^-_{(2)} \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ -0.228 & 0.228 & 0.445 & -0.445 \\ 0.445 & -0.445 & 0.228 & -0.228 \end{bmatrix}.$$

6. Solve the (MIPR) problem:

$$\begin{aligned}
\text{minimize:} \quad & 2x_1 + x_2 - 2x_3 \\
\text{subject to:} \quad 10\, x_1 + x_2 &= 19, \\
x_1 &= 0.5\lambda_1 + 0.5\lambda_2 + 0.5\lambda_3 + 0.5\lambda_4, \\
x_2 &= -0.228\lambda_1 + 0.228\lambda_2 + 0.445\lambda_3 - 0.445\lambda_4, \\
x_3 &= 0.445\lambda_1 - 0.445\lambda_2 + 0.228\lambda_3 - 0.228\lambda_4, \\
x_1 &\geq 0, \\
x_1, x_2 &\in \mathbb{Z}, \\
\lambda_1, \lambda_2, \lambda_3, \lambda_4 &\in \mathbb{R}_+.
\end{aligned}$$

The optimal solution of (MIPR) is $x^* = (2, -1, 1.506)$, the current upper bound is –0.012. For illustration, see Figure 4.2d.

7. Solve the (FR) problem:

$$\begin{aligned}
\text{minimize:} \quad & 2x_1 + x_2 - 2x_3 \\
\text{subject to:} \quad 10\, x_1 + x_2 &= 19, \\
x_1 &= 2, \\
x_2 &= -1, \\
(x_1, x_2, x_3) &\in \mathbb{L}^3,
\end{aligned}$$

The (FR) problem improves the solution and provides a feasible MISOCO solution $x^r = (2, --1, \sqrt{3})$ with objective value -0.464. For illustration, see Figure 4.2e.

8. Obtain Jordan frames. The Jordan frame is

$$F = \begin{bmatrix} 0.5 & 0.5 \\ -0.25 & 0.25 \\ 0.433 & -0.433 \end{bmatrix}.$$

9. Solve the (MIPR) problem:

$$\begin{aligned} \text{minimize:} \quad & 2x_1 + x_2 - 2x_3 \\ \text{subject to:} \quad & 10\, x_1 + x_2 = 19, \\ & x_1 = 0.5\lambda_1 + 0.5\lambda_2, \\ & x_2 = -0.25\lambda_1 + 0.25\lambda_2, \\ & x_3 = 0.433\lambda_1 - 0.433\lambda_2, \\ & x_1 \geq 0, \\ & x_1, x_2 \in \mathbb{Z}, \\ & \lambda_1, \lambda_2 \in \mathbb{R}_+. \end{aligned}$$

(MIPR) gives the same solution $x^* = (2, -1, \sqrt{3})$.

The relaxation objective for this problem is –0.4697, where the bound generated by the conic rounding heuristic is –0.4641. Despite the small gap, we cannot guarantee optimality in general. However, the solution provided by the heuristic is the unique optimal solution for this example.

Discussion

Despite following a simple logic, the primal rounding heuristic works well in practice. However, the performance depends highly on the feasible region of the problem. For problems where feasible solutions appear inside SOC, it is relatively easy to find Jordan frames where a feasible solution lies inside their convex combinations. On the other hand, if all the feasible solutions appear on, or close to the boundary of the SOCs, then we need to find the exact Jordan frame to obtain a feasible solution. Finding the exact Jordan frame is not always possible. In our experiments with CBLIB test problems, we observed that the primal rounding heuristic is not able to produce feasible solutions for stochastic service system design (`sssd`) problems. To provide an alternative to the primal rounding heuristic for problems where solutions are close to the boundary of SOCs, we propose the dual rounding heuristic in the following subsection.

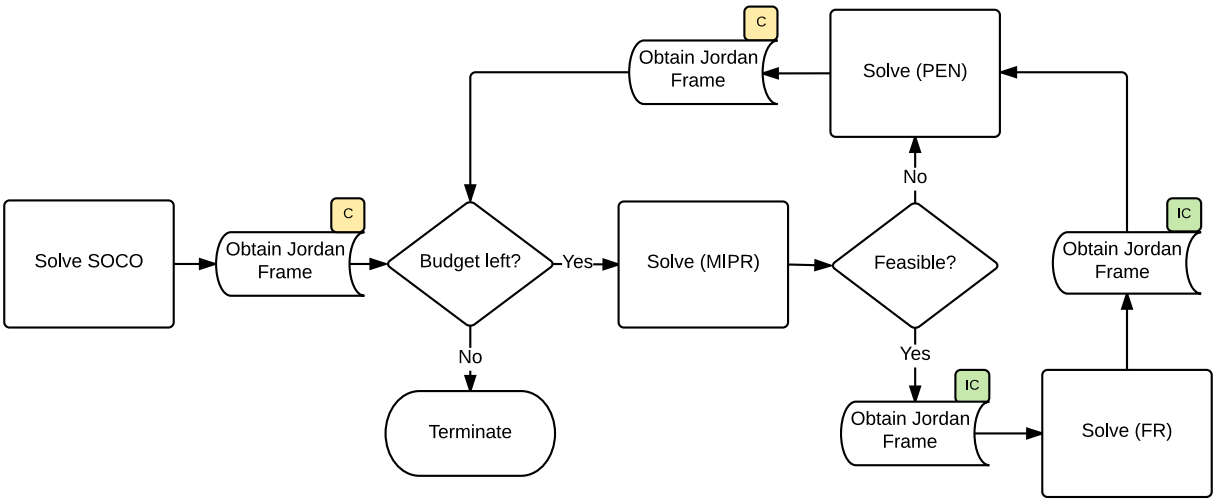
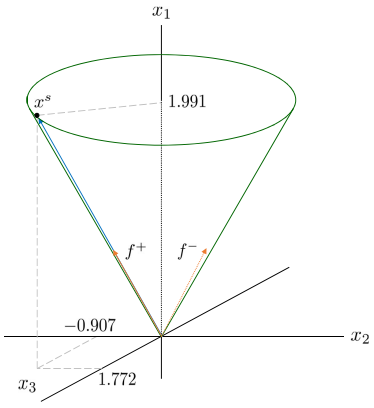


Figure 4.1: Flow of the primal rounding heuristic.

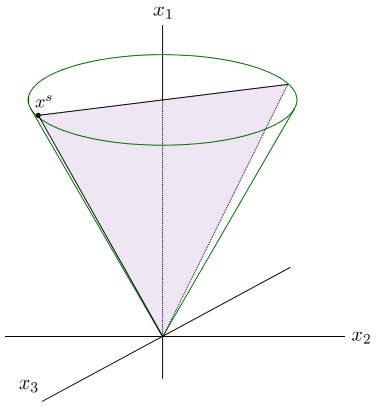
Algorithm 4 The primal rounding heuristic for MISOCO

Require: A MISOCO instance (4.1),
maximum number of iterations t
Ensure: A feasible solution $\tilde{x}$ to MISOCO, if found

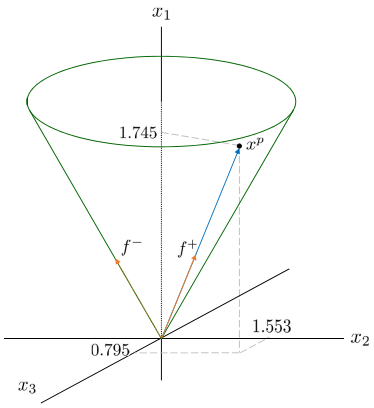
Set $\tilde{c} = \infty$, $\varphi = 0.5$
Solve the continuous relaxation of MISOCO, obtain its solution x^s
Add F^s to the Jordan frame pool
while $i \leq t$ **do**
  Solve (MIPR), obtain its solution x^* if exists
  if (MIPR) is feasible **then**
    Add F^* to the Jordan frame pool
    Solve (FR) using x^*, obtain its solution x^r
    Add F^r to the Jordan frame pool
    if $c^\top x^r \leq \tilde{c}$ **then**
      $\tilde{c} = c^\top x^r$, $\tilde{x} = x^r$
    $\varphi = \frac{1+\varphi}{2}$
  else
    $\varphi = \frac{\varphi}{2}$
  Solve the penalty problem (PEN), obtain its solution x^p
  Add F^p to the Jordan frame pool
  $i = i + 1$
return $\tilde{x}$



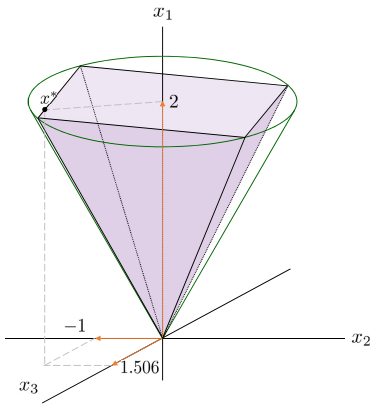
(a) Solution of the continuous relaxation,



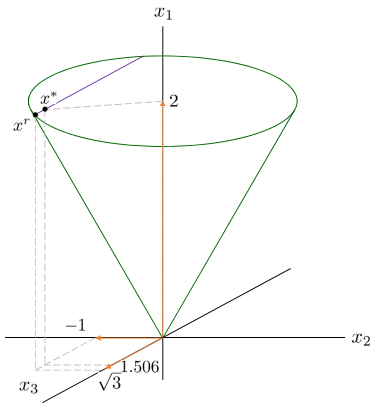
(b) Conic feasible region for (MIPR)



(c) Solution of the penalty problem (PEN), $x^p = (1.745, 1.553, 0.795)$.



(d) Conic feasible region for (MIPR), $x^* = (2, -1, 1.506)$.



(e) Solution of (FR), $x^r = (2, -1, \sqrt{3})$.

Figure 4.2: Steps of the primal rounding heuristic on the example problem.

4.2.2 The dual rounding heuristic

Using the relationship between the rounding problems and the original MISOCO instances, we can approach the problem of finding a feasible solution from the dual side. As shown in Figure 3.1, (D-DR) is a relaxation of the original MISOCO instance. The underlying idea for the dual rounding heuristic is to solve a series of (D-DR) problems with integrality constraints and (FR) problems.

The dual rounding heuristic starts with the solution of the continuous relaxation of the original instance. Next, the mixed-integer dual rounding problem (MIDR) is solved by using the Jordan frame matrix that was obtained from the relaxation solution, which is written as follows:

$$\begin{aligned} \text{minimize:} \quad & c^\top x & & & & \\ \text{subject to:} \quad & Ax & = & b, & & \\ & F^\top x & \geq & 0, & & \\ & x_1^i & \geq & 0, & i \in 1, \ldots, k & \\ & x_j & \in & \mathbb{Z}, & j \in J \subseteq N. & \end{aligned} \tag{MIDR}$$

If the solution is conic feasible, then the solution is an optimal solution to the MISOCO problem, and we terminate. Otherwise, Jordan frames obtained from the solution are added to the matrix F. Note that we can still obtain the Jordan frame that corresponds to the projection of the (MIDR) solution onto $\mathcal{K}$. Adding this Jordan frame to matrix F cuts off the current (MIDR) solution for the next iteration. By using the conic infeasible solution, we solve an (FR) problem by fixing integer variables. If the problem is feasible, then we obtained a feasible solution to MISOCO. We add the corresponding Jordan frame to F, and continue to the next iteration.

Figure 4.3 shows an overview of the dual rounding heuristic, and Algorithm 5 describes the dual rounding heuristic steps.

In theory, one can keep adding unique Jordan frames until an outer-approximation of the $\mathcal{K}$ to the desired precision is achieved because the second-order cones are self-dual and the constraint $F^\top x \geq 0$ provides a supporting hyperplane for $\mathcal{K}$. The set

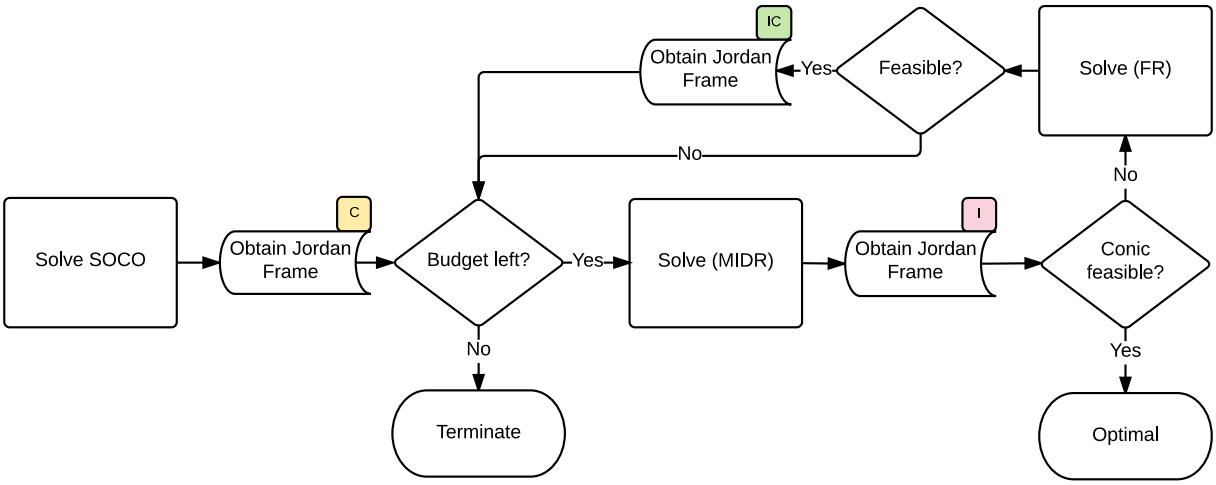


Figure 4.3: Flow of the dual rounding heuristic.

Algorithm 5 The dual rounding heuristic for MISOCO

Require: A MISOCO instance (4.1),
maximum number of iterations t
Ensure: A feasible solution $\tilde{x}$ to MISOCO if found, a global lower bound c_L

Set $\tilde{\mathrm{c}} = \infty$
Solve the continuous relaxation of MISOCO, obtain its solution x^s, set $\mathrm{c}_L = c^\top x^s$
Add F^s to the Jordan frame pool
while $i \leq t$ **do**
  Solve (MIDR), obtain solution x^*
  Add F^* to the Jordan frame pool
  if $c^\top x^* \geq \mathrm{c}_L$ **then**
    $\mathrm{c}_L = c^\top x^*$
  if $x^* \in \mathcal{K}$ **then**
    $\tilde{x} = x^*$, $\tilde{\mathrm{c}} = c^\top x^*$
    Terminate with an optimal solution to MISOCO x^*.
  else
    Solve (FR) using x^*, obtain its solution x^r if exists
    if (FR) is feasible **then**
      Add F^r to the Jordan frame pool
      if $c^\top x^r \leq \tilde{\mathrm{c}}$ **then**
        $\tilde{\mathrm{c}} = c^\top x^r$, $\tilde{x} = x^r$
  if $\mathrm{c}_L = \tilde{c}$ **then**
    Terminate with an optimal solution to MISOCO $\tilde{x}$.
  $i = i + 1$
return $\tilde{x}, \mathrm{c}_L$

of Jordan frames provides a rough outer-approximation of the SOCs since we often have a limited number of Jordan frames as shown in our numerical results.

Because of its relationship to the original MISOCO problem, (MIDR) is feasible at each iteration of the heuristic for a feasible MISOCO problem. Therefore, if (MIDR) becomes infeasible at any iteration of the dual rounding heuristic, then the original MISOCO problem is infeasible. Moreover, if the solution to (MIDR) satisfies conic feasibility $x \in \mathcal{K}$, then it is an optimal solution for MISOCO.

One of the major benefits of the dual rounding heuristic is the detection of infeasible cases. The other benefit is that the dual rounding heuristic provides a lower bound for the MISOCO problem, thus providing a better gap for the optimality information. A lower bound is provided even if the heuristic fails to find a feasible solution. As shown in our numerical experiments, the dual rounding heuristic works well for certain problem types and provides a good solution in a few iterations. The main difference between the dual rounding heuristic and the primal rounding heuristic is the existence of penalty problems. Since any solution of (MIDR) cuts off the current solution and effectively decreases the feasible region, there is no need to solve a separate penalty problem.

Example

In this subsection, we provide a numerical example to illustrate how the dual rounding heuristic works in practice. The steps of the dual rounding heuristic are illustrated in Figure 4.4.

Consider the following optimization model:

$$\begin{aligned} \text{minimize:} \quad & -15x_2 - 8x_3 \\ \text{subject to:} \quad & x_1 = 3, \\ & x_2, x_3 \leq 3, \\ & (x_1, x_2, x_3) \in \mathbb{L}^3, \\ & x_1, x_3 \in \mathbb{Z}. \end{aligned}$$

Following are the steps of the dual rounding heuristic. Numerical values are given up to three digits precision.

1. Solve the SOCO subproblem:

$$\begin{aligned} \text{minimize:} \quad & -15x_2 - 8x_3 \\ \text{subject to:} \quad & x_1 = 3, \\ & x_2, x_3 \leq 3, \\ & (x_1, x_2, x_3) \in \mathbb{L}^3. \end{aligned}$$

   The solution of the SOCO subproblem is $x^* = (3, 2.647, 1.412)$ with objective value –51. For illustration, see Figure 4.4a.

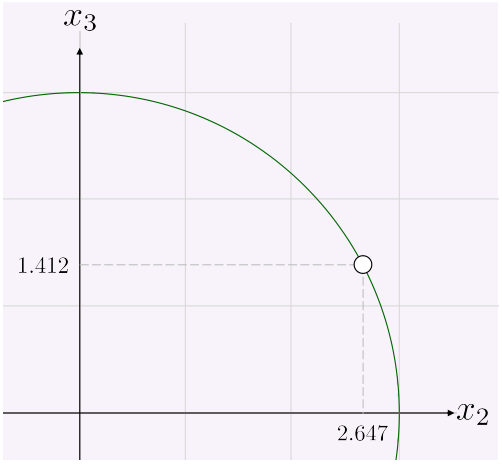
2. Obtain the Jordan frames. The F matrix is

$$F = \begin{bmatrix} 0.5 & 0.5 \\ 0.471 & -0.471 \\ 0.167 & -0.167 \end{bmatrix}.$$

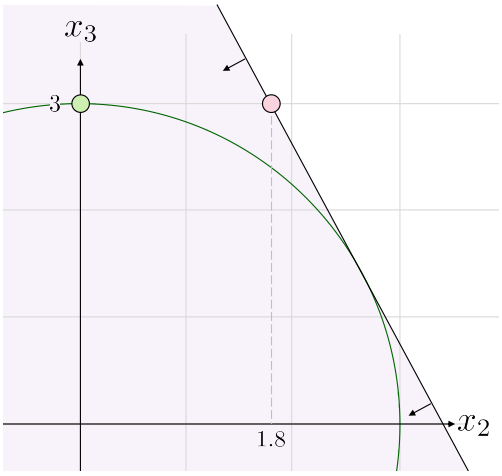
3. Solve the (MIDR) problem:

$$\begin{aligned} \text{minimize:} \quad & -15x_2 - 8x_3 \\ \text{subject to:} \quad & x_1 = 3, \\ & x_2, x_3 \leq 3, \\ & 0.5x_1 + 0.471x_2 + 0.167x_3 \geq 0, \\ & 0.5x_1 - 0.471x_2 - 0.167x_3 \geq 0, \\ & x_1 \geq 0, \\ & x_1, x_3 \in \mathbb{Z}. \end{aligned}$$

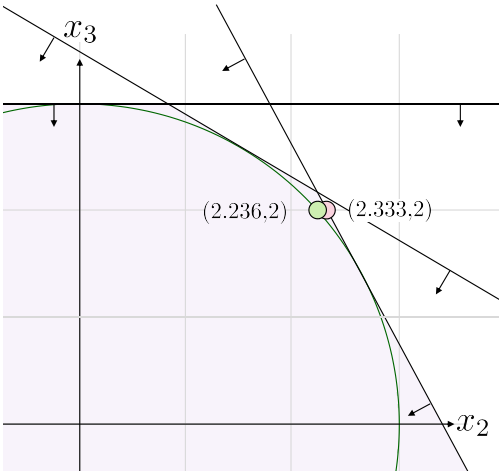
   (MIDR) provides a solution $\bar{x} = (3, 1.8, 3)$, which is conic infeasible. We fix the integer variables $x_1 = 3, x_3 = 3$ and solve the (FR) problem. The (FR) problem provides a conic and integer feasible solution $x^* = (3, 0, 3)$ with objective value –24. This is the best feasible solution for the MISOCO problem so far. For illustration, see Figure 4.4b.



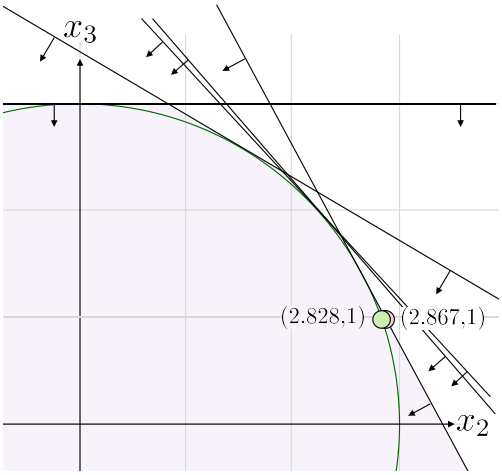
(a) Solution of the original continuous relaxation, $x^s = (3, 2.647, 1.412)$.



(b) Solutions obtained after the first iteration, $x^* = (3, 1.8, 3)$, $x^r = (3, 0, 3)$.



(c) Solutions obtained after the second iteration, $x^* = (3, 2.333, 2)$, $x^r = (3, 2.236, 2)$.



(d) Solutions obtained after the third iteration, $x^* = (3, 2.867, 1, 3)$, $x^r = (3, 2.828, 1)$.

Figure 4.4: Steps of the dual rounding heuristic on the sample problem for the cross-section at $x_1 = 3$.

4. Obtain Jordan frames from both the optimal solution of (MIDR) and the solution of (FR). The F matrix is

$$F = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0.471 & -0.471 & 0.257 & -0.257 & 0 & 0 \\ 0.167 & -0.167 & 0.429 & -0.429 & 0.5 & -0.5 \end{bmatrix}.$$

5. Solve the (MIDR) problem:

$$\begin{aligned} \text{minimize:} \quad & -15x_2 - 8x_3 \\ \text{subject to:} \quad & x_1 = 3, \\ & x_2, x_3 \leq 3, \\ & 0.5x_1 + 0.471x_2 + 0.167x_3 \geq 0, \\ & 0.5x_1 - 0.471x_2 - 0.167x_3 \geq 0, \\ & 0.5x_1 + 0.257x_2 + 0.429x_3 \geq 0, \\ & 0.5x_1 - 0.257x_2 - 0.429x_3 \geq 0, \\ & 0.5x_1 + 0.5x_3 \geq 0, \\ & 0.5x_1 - 0.5x_3 \geq 0, \\ & x_1 \geq 0, \\ & x_1, x_3 \in \mathbb{Z}. \end{aligned}$$

The solution of (MIDR) is $x^* = (3, 2.333, 2)$. We fix the integer variables $x_1 = 3, x_3 = 2$ and solve (FR). The (FR) provides a feasible solution $x^* = (3, 2.236, 2)$ with objective value –49.541. This is the best feasible solution for the MISOCO problem so far. For illustration see Figure 4.4c.

6. Obtain the Jordan frames.

$$F = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0.471 & -0.471 & 0.257 & -0.257 & 0 & 0 & 0.379 & -0.379 & 0.372 & -0.372 \\ 0.167 & -0.167 & 0.429 & -0.429 & 0.5 & -0.5 & 0.325 & -0.325 & 0.333 & -0.333 \end{bmatrix}.$$

7. Solve the (MIDR) problem:

$$
\begin{aligned}
\text{minimize:} \quad & -15x_2 - 8x_3 \\
\text{subject to:} \quad & x_1 = 3, \\
& x_2, x_3 \leq 3, \\
& 0.5x_1 + 0.471x_2 + 0.167x_3 \geq 0, \\
& 0.5x_1 - 0.471x_2 - 0.167x_3 \geq 0, \\
& 0.5x_1 + 0.257x_2 + 0.429x_3 \geq 0, \\
& 0.5x_1 - 0.257x_2 - 0.429x_3 \geq 0, \\
& 0.5x_1 + 0.5x_3 \geq 0, \\
& 0.5x_1 - 0.5x_3 \geq 0, \\
& 0.5x_1 + 0.379x_2 + 0.325x_3 \geq 0, \\
& 0.5x_1 - 0.379x_2 - 0.325x_3 \geq 0, \\
& 0.5x_1 + 0.372x_2 + 0.333x_3 \geq 0, \\
& 0.5x_1 - 0.372x_2 - 0.333x_3 \geq 0, \\
& x_1 \geq 0, \\
& x_1, x_3 \in \mathbb{Z}.
\end{aligned}
$$

   The solution of (MIDR) is $x^* = (3, 2.867, 1)$, which violates the conic constraint slightly. We fix the integer variables $x_1 = 3, x_3 = 1$ and solve (FR). The (FR) produces the solution $x^r = (3, 2.828, 1)$ with objective value –50.426. This is the best feasible solution for the MISOCO so far. For illustration, see Figure 4.4d.

Repeating the heuristic for one more iteration results in an (MIDR) solution of $x^* = (3, 2, 828, 1)$, which proves that the solution is the true optimal solution for the MISOCO. This example demonstrates a good case where only a few iterations provided the unique optimal solution. In general, there is no guarantee that a feasible solution can be found in a finite number of iterations.

Discussion

The dual rounding heuristic is useful for finding a feasible solution when the integer feasible solutions of MISOCO are close to the boundary of the SOCs. A

pathological case occurs if there is a continuous region near the both sides of the boundary. Without (FR), infinitely many iterations are required to find a feasible solution in this case.

The dual rounding heuristic is similar to the primal rounding heuristic in terms of utilizing Jordan frames to generate feasible solutions. The main difference lies in where the solutions of the mixed-integer rounding problems appear. If feasible, (MIPR) produces a feasible solution to MISOCO. On the other hand, if MISOCO is feasible, then (MIDR) is always feasible, but its solution is likely to be infeasible for MISOCO because of violated conic constraints. These two different heuristics are useful for different cases. The primal rounding heuristic works better when feasible solutions are inside the SOCs, whereas it fails for cases where the feasible solutions are close to or lie on the boundary of the SOCs. The dual rounding heuristic works better when feasible solutions are close to the boundary of SOCs, but fails when there is a continuous feasible region alongside a cone.

In the next subsection we present the primal-dual rounding heuristic, where the iteration budget is shared equally between the primal rounding and the dual rounding without needing penalty problems.

4.2.3 The primal-dual rounding heuristic

As we discussed in previous subsections, the primal rounding heuristic and the dual rounding heuristic are useful for different types of feasible regions. For a problem with known structure, one can decide which heuristic to deploy. However, one often does not have enough knowledge about the feasible region, making the decision of which heuristic to use. In this subsection, we propose a combination of two heuristics. This heuristic is called the *primal-dual rounding heuristic* and consists of components of both heuristics.

The primal-dual rounding heuristic starts with the solution of the SOCO problem and its Jordan frames. Then, the (MIPR) problem is solved using the frames available

in the pool. If it is feasible, then an (FR) step follows to improve the solution. The generated solution is fed to the set of Jordan frames, and the best solution is updated. Then, whether the (MIPR) is feasible or not, the (MIDR) problem is solved. The (MIDR) problem is guaranteed to be always feasible unless the original problem is infeasible; hence the solution can be added to the Jordan frame pool. If the solution to (MIDR) is conic feasible, it means that we found an optimal solution to the original MISOCO problem. If not, then we use the solution in an (FR) step to see if the (MIDR) can lead to a feasible solution. If (FR) provides a feasible solution, then we add the corresponding Jordan frame to the pool. We iterate by solving (MIPR) until we reach a predefined iteration limit.

Figure 4.5 shows an overview of the primal-dual rounding heuristic, and Algorithm 6 describes the steps of the primal-dual rounding heuristic.

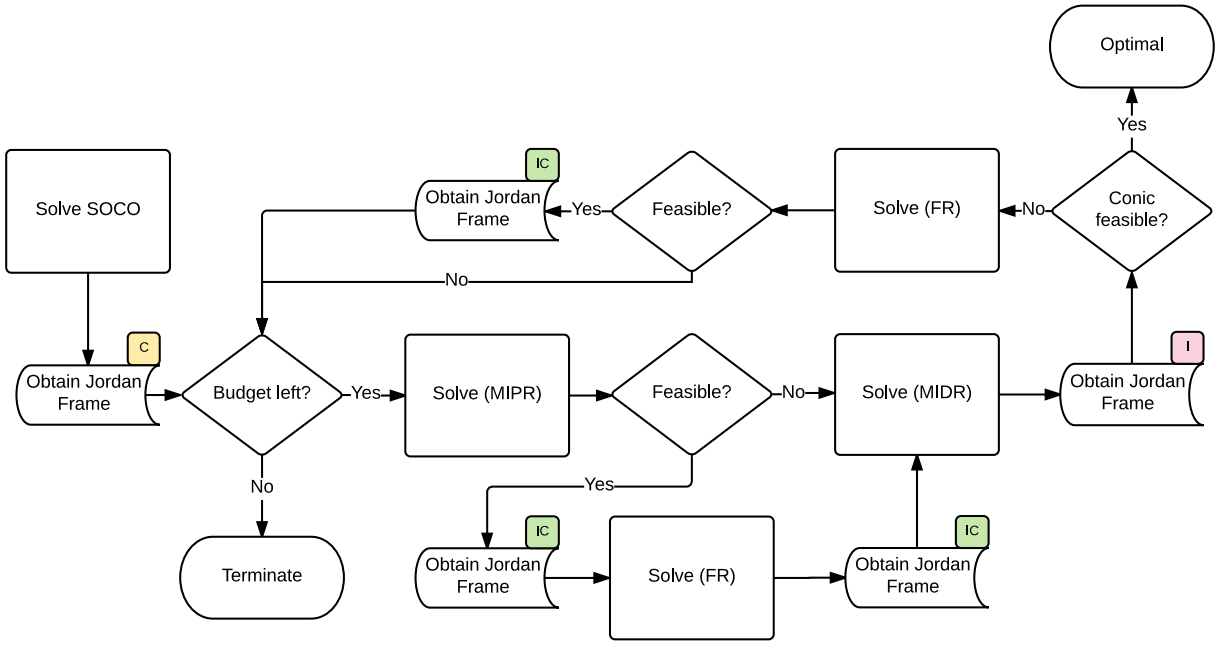


Figure 4.5: Flow of the primal-dual rounding heuristic.

Although the primal-dual rounding heuristic is a combination of both of the primal and dual rounding heuristics that were introduced earlier, there are some differences

Algorithm 6 The primal-dual rounding heuristic for MISOCO

Require: A MISOCO instance (4.1),
maximum number of iterations t
Ensure: A feasible solution $\tilde{x}$ to MISOCO if found, a global lower bound c_L

Set $\tilde{\mathrm{c}} = \infty$
Solve the continuous relaxation of MISOCO, obtain its solution x^s, set $\mathrm{c}_L = c^\top x^s$
Add F^s to the Jordan frame pool
while $i \leq t$ **do**
    Solve (MIPR), obtain its solution x^* if it exists
    if (MIPR) feasible **then**
        Add F^* to the Jordan frame pool
        Solve (FR) using x^*, obtain its solution x^r
        Add F^r to the Jordan frame pool
        if $c^\top x^r \leq \tilde{\mathrm{c}}$ **then**
            $\tilde{\mathrm{c}} = c^\top x^r$, $\tilde{x} = x^r$
    Solve (MIDR), obtain solution x^*
    Add F^* to the Jordan frame pool
    if $x^* \in \mathcal{K}$ **then**
        $\tilde{\mathrm{c}} = c^\top x^*, \tilde{x} = x^*$
        Terminate with an optimal solution to MISOCO.
    if $c^\top x^* \geq \mathrm{c}_L$ **then**
        $\mathrm{c}_L = c^\top x^*$
    Solve (FR) using x^*, obtain its solution x^r if it exists
    if (FR) feasible **then**
        Add F^r to the Jordan frame pool
        if $c^\top x^r \leq \tilde{\mathrm{c}}$ **then**
            $\tilde{\mathrm{c}} = c^\top x^r$, $\tilde{x} = x^r$
    if $\mathrm{c}_L = \tilde{\mathrm{c}}$ **then**
        Terminate with an optimal solution to MISOCO
    $i = i + 1$
return $\tilde{x}, c_L$

in terms of the flow. The most significant difference between the primal rounding and the primal-dual rounding heuristic is how we generate new Jordan frames. A penalty problem is solved in every iteration of the primal rounding heuristic to generate new frames. This means that we need to solve two SOCOs in every iteration if we have a feasible solution. For the primal-dual rounding heuristic, we benefit from the frames generated by the (MIDR) solutions. Another advantage is that there is no need to solve a separate penalty problem to generate new frames. However, whereas the the primal rounding heuristic does not provide a lower bound, the dual rounding heuristic provides a lower bound at every iteration. By combining these two heuristics, we aim to provide a feasible solution and a better lower bound at the same time.

4.2.4 Hybrid strategy

The primal-dual rounding heuristic allocates budget equally among primal and dual rounding heuristics. Moreover, primal rounding uses only Jordan frames that are generated by the dual rounding heuristic, specifically by (MIDR) and dual (FR) problems. In our experiments we noticed that the primal rounding heuristic provides a feasible solution to the majority of the test problems within a few iterations. Therefore, we present a hybrid strategy (HS) as a combination of two heuristics, where a few iterations are pure primal rounding heuristic and the remaining are dual rounding heuristic only. They do not interact, except we carry Jordan frames obtained in the primal side to the dual rounding heuristic and start it from the existing Jordan frames. Although this sounds intuitive, the benefit of carrying existing Jordan frames might be controversial, because it sometimes changes the frames obtained from the dual side significantly.

4.2.5 Extending heuristics to convex quadratic optimization

It is well known that convex quadratic optimization problems can be written as equivalent SOCO formulations. By enforcing the integrality, we can write any mixed-integer quadratically constrained quadratic optimization (MIQCQO) problem in terms of an equivalent MISOCO formulation and translate the conic rounding heuristic.

Without loss of generality, suppose we have a convex quadratic constraint in a MIQCQO formulation of the form

$$x^\top Q x + 2p^\top x + \mathrm{r} \leq 0, \tag{4.2}$$

where $Q \in \mathbb{R}^{n\times n}$ is a symmetric and positive semidefinite matrix, $p \in \mathbb{R}^n$ and $\mathrm{r} \in \mathbb{R}$. Such a constraint can also be obtained after moving a quadratic objective into constraints and adding an auxiliary variable to x. There are two ways of transforming

this constraint into a conic constraint based on positive definiteness of Q, which are inspected separately in the following subsections.

$Q \succ 0$

If Q is a positive definite matrix, then we can transform (4.2) as follows:

$$\begin{aligned} x^\top Q x + 2p^\top x &\leq -\mathrm{r} \\ x^\top Q^{1/2} Q^{1/2} x + 2(Q^{-1/2}p)^\top Q^{1/2} x + p^\top Q^{-1} p &\leq p^\top Q^{-1} p - \mathrm{r} \\ (Q^{1/2}x + Q^{-1/2}p)^2 &\leq p^\top Q^{-1} p - \mathrm{r}. \end{aligned}$$

Introducing two auxiliary variables $u := Q^{1/2}x + Q^{-1/2}p$ and $t := \sqrt{p^\top Q^{-1}p - \mathrm{r}}$ gives us the equivalent form of (4.2) as a set of linear and SOC constraints:

$$x^\top Q x + 2p^\top x + \mathrm{r} \leq 0 \quad \equiv \quad \begin{cases} u = Q^{1/2}x + Q^{-1/2}p \\ t = \sqrt{p^\top Q^{-1}p - \mathrm{r}} \\ (t,u) \in \mathbb{L}^{n+1} \end{cases}.$$

For the general MIQCO of the form

$$\begin{aligned} \text{minimize:} \quad & c^\top x \\ \text{subject to:} \quad & Ax = b, \\ & x^\top Q x + 2p^\top x + \mathrm{r} \leq 0, \\ & x_j \in \mathbb{Z} \quad j \in J \subseteq N, \end{aligned} \tag{4.3}$$

where $Q \succ 0$, we can write the corresponding MISOCO as follows:

$$\begin{aligned} \text{minimize:} \quad & c^\top x \\ \text{subject to:} \quad & Ax = b, \\ & Q^{1/2}x - u = -Q^{-1/2}p, \\ & t = \sqrt{p^\top Q^{-1}p - \mathrm{r}}, \\ & (t,u) \in \mathbb{L}^{n+1}. \end{aligned}$$

For a solution for MIQCO (4.3) x^*, we can find the corresponding values of u^*, t^*, and Jordan frame corresponds to the solution in MISOCO form:

$$F^* = \begin{bmatrix} 0.5 & 0.5 \\ \frac{u^*}{2\|u^*\|} & -\frac{u^*}{2\|u^*\|} \end{bmatrix}.$$

Using this Jordan frame, now we can write any of the problems defined for our heuristics below. For example, the (MIPR) problem that corresponds to solution x^* is

$$\begin{aligned}
\text{minimize:} \quad & c^\top x \\
\text{subject to:} \quad & Ax = b, \\
& Qx - Q^{1/2}u = -p, \\
& t = \sqrt{p^\top Q^{-1} p - \mathrm{r}}, \\
& t - 0.5\lambda_1 - 0.5\lambda_2 = 0, \\
& u - \delta\lambda_1 + \delta\lambda_2 = 0, \\
& \lambda_1, \lambda_2 \geq 0, \\
& x_j \in \mathbb{Z} \qquad\qquad j \in J \subseteq N,
\end{aligned}$$

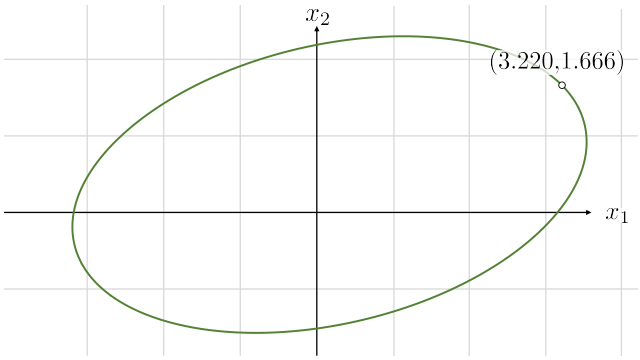
where $\delta = {}^{u^*}/_{2\|u^*\|}$. Putting everything together in terms of the original variable x and solution x^* gives us the (MIPR) problem without needing an intermediate step:

$$\begin{aligned}
\text{minimize:} \quad & c^\top x \\
\text{subject to:} \quad & Ax = b, \\
& \lambda_1 + \lambda_2 = 2\sqrt{p^\top Q^{-1} p - \mathrm{r}}, \\
& 2(x^{*\top} Q x^*)Qx - Qx^*\lambda_1 + Qx^*\lambda_2 = -2(x^{*\top} Q x^*)p, \\
& \lambda_1, \lambda_2 \geq 0, \\
& x_j \in \mathbb{Z} \qquad\qquad j \in J \subseteq N.
\end{aligned} \tag{4.4}$$

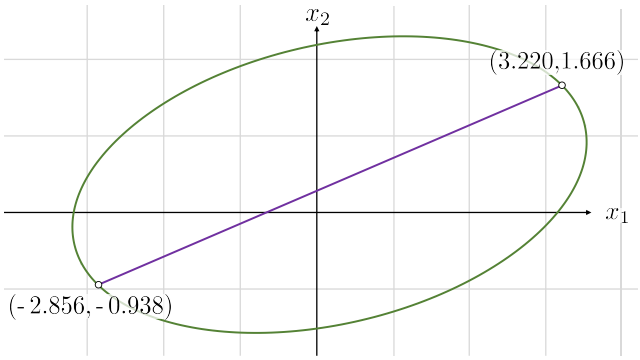
Here, the second constraint satisfies λ values to change in a convex region, while the third constraint gives the point x as a convex combination of two end points.

Example Let us show how the heuristic can be applied for MIQCQO problems in practice on a small example. See Figure 4.6 that accompanies the following steps. Suppose we have the following MIQCQO problem:

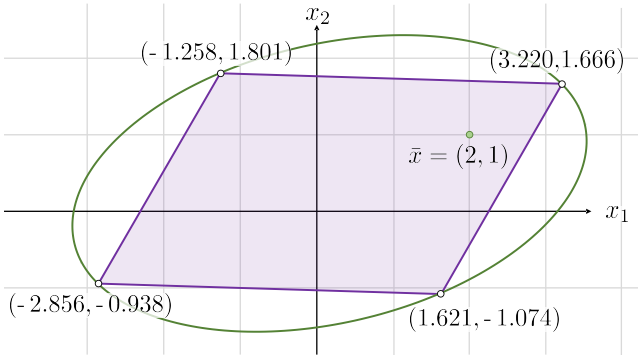
$$\begin{aligned}
\text{minimize:} \quad & -x_1 - x_2 \\
\text{subject to:} \quad & x_1^2 + 3x_2^2 - x_1x_2 - 2x_2 - 10 \leq 0, \\
& x_1, x_2 \geq 0, \\
& x_1, x_2 \in \mathbb{Z}.
\end{aligned}$$



(a) Solution of the continuous relaxation $x^* = (3.220, 1.666)$.



(b) Conic feasible region for (MIPR) (infeasible).



(c) Conic feasible region for (MIPR), optimal solution $\bar{x} = (2, 1)$.

Figure 4.6: Steps of the primal rounding heuristic on a convex quadratic sample problem.

The optimal solution of the continuous relaxation is $x = (3.220, 1.666)$ as shown in Figure 4.6a. The (MIPR) problem (4.4) at x^* is

$$\begin{aligned}
\text{minimize:} \quad & -x_1 - x_2 \\
\text{subject to:} \quad & \lambda_1 + \lambda_2 = 6.438, \\
& 0.982x_1 - 0.185x_2 - 0.426\lambda_1 + 0.426\lambda_2 = 0.111, \\
& -0.185x_1 + 1.722x_2 - 0.261\lambda_1 + 0.261\lambda_2 = 0.593, \\
& x_1, x_2 \geq 0, \\
& \lambda_1, \lambda_2 \geq 0 \\
& x_1, x_2 \in \mathbb{Z}.
\end{aligned}$$

For $\lambda_2 = 0$, we obtain the endpoint of the convex combination $x = (3.220, 1.666)$, which was the solution of the continuous relaxation. For another extreme at $\lambda_1 = 0$, we get $x = (-2.856, -0.939)$. See Figure 4.6b. From the earlier solution we have

$$F^* = \begin{bmatrix} 0.5 & 0.5 \\ \frac{u}{2\|u\|} & -\frac{u}{2\|u\|} \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 \\ 0.426 & -0.426 \\ 0.261 & -0.261 \end{bmatrix}$$

If we solve the corresponding penalty problem

$$\begin{aligned}
\text{minimize:} \quad & \varphi(-0.707x_1 - 0.707x_2) + (1-\varphi)|0.852u_1 + 0.522u_2| \\
\text{subject to:} \quad & 0.982x_1 - 0.185x_2 - u_1 = 0.111, \\
& -0.185x_1 + 1.722x_2 - u_2 = 0.593, \\
& t = 3.219, \\
& x_1, x_2 \geq 0, \\
& (t, u_1, u_2) \in \mathbb{L}^3,
\end{aligned}$$

for $\varphi = 0.5$, we get $x^* = (0.545, 0.001)$ with a Jordan frame of

$$F^* = \begin{bmatrix} 0.5 & 0.5 \\ 0.261 & -0.261 \\ -0.426 & 0.426 \end{bmatrix}.$$

In the second iteration, the (MIPR) problem becomes as follows:

$$
\begin{aligned}
\text{minimize:} \quad & -x_1 - x_2 \\
\text{subject to:} \quad & \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 6.438, \\
& 0.982x_1 - 0.185x_2 - 0.426\lambda_1 + 0.426\lambda_2 - 0.261\lambda_3 + 0.261\lambda_4 = 0.111, \\
& -0.185x_1 + 1.722x_2 - 0.261\lambda_1 + 0.261\lambda_2 + 0.426\lambda_3 - 0.426\lambda_4 = 0.593, \\
& x_1, x_2 \geq 0, \\
& \lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0 \\
& x_1, x_2 \in \mathbb{Z},
\end{aligned}
$$

which gives the feasible solution $\bar{x} = (2, 1)$. See Figure 4.6c for the feasible region of the (MIPR). The same trick can also be applied for the dual rounding and primal-dual rounding heuristics.

$Q \succeq 0$

We consider a case where Q is positive semidefinite and singular. Since the inverse Q^{-1} is not available, we need to derive another set of constraints to convert quadratic constraint (4.2) into a SOC constraint. We can write

$$
\begin{aligned}
x^\top Q x &\leq -2p^\top x - \mathrm{r}, \\
x^\top Q^{1/2} Q^{1/2} x &\leq \left(\frac{-2p^\top x - \mathrm{r} + 1}{2}\right)^2 - \left(\frac{-2p^\top x - \mathrm{r} - 1}{2}\right)^2, \\
\left\| \frac{-2p^\top x - \mathrm{r} - 1}{2} \quad Q^{1/2} x \right\| &\leq \frac{-2p^\top x - \mathrm{r} + 1}{2}
\end{aligned}
$$

Introducing two auxiliary variables $u := Q^{1/2}x$ and $t := -2p^\top x - \mathrm{r}$ gives us the equivalent form of (4.2) as a set of linear and SOC constraints as

$$
x^\top Q x + 2p^\top x + \mathrm{r} \leq 0 \quad \equiv \quad \begin{cases} u = Q^{1/2}x \\ t = -2p^\top x - \mathrm{r} \\ ((t+1)/2, (t-1)/2, u) \in \mathbb{L}^{n+2} \end{cases}.
$$

For the general MIQCO of the form

$$\begin{aligned} \text{minimize:} \quad & c^\top x \\ \text{subject to:} \quad & Ax = b, \\ & x^\top Q x + 2p^\top x + \mathrm{r} \leq 0, \\ & x_j \in \mathbb{Z} \quad j \in J \subseteq N, \end{aligned} \tag{4.5}$$

where $Q \succeq 0$, we can write the corresponding MISOCO as follows:

$$\begin{aligned} \text{minimize:} \quad & c^\top x \\ \text{subject to:} \quad & Ax = b, \\ & Q^{1/2}x - u = 0, \\ & 2p^\top x + t = -\mathrm{r}, \\ & ((t+1)/2, (t-1)/2, u) \in \mathbb{L}^{n+2}, \\ & x_j \in \mathbb{Z} \qquad j \in J \subseteq N. \end{aligned}$$

For a solution for MIQCO (4.5) x^*, we can find the values of u^*, t^* and the Jordan frame that correspond to the solution in MISOCO form:

$$F^* = \begin{bmatrix} 0.5 & 0.5 \\ \frac{(t^*-1)}{4\|(t^*-1)/2 \quad u^*\|} & -\frac{(t^*-1)}{4\|(t^*-1)/2 \quad u^*\|} \\ \frac{u^*}{2\|(t^*-1)/2 \quad u^*\|} & -\frac{u^*}{2\|(t^*-1)/2 \quad u^*\|} \end{bmatrix}.$$

Using this Jordan frame, now we can write any of the problems defined for our heuristics below. For example, the (MIPR) problem that corresponds to solution x^* is

$$\begin{aligned} \text{minimize:} \quad & c^\top x \\ \text{subject to:} \quad & Ax = b, \\ & Q^{1/2}x - u = 0, \\ & 2p^\top x + t = -\mathrm{r}, \\ & t - \lambda_1 - \lambda_2 = -1, \\ & \delta t - (t^*-1)\lambda_1 + (t^*-1)\lambda_2 = \delta, \\ & \delta u - u^*\lambda_1 + u^*\lambda_2 = 0, \\ & \lambda_1, \lambda_2 \geq 0, \\ & x_j \in \mathbb{Z} \qquad j \in J \subseteq N, \end{aligned}$$

where $\delta = 2\|(t^*-1)/2 \quad u^*\|$. Putting everything together in terms of the original variable x and solution x^* gives us the (MIPR) problem without needing an

intermediate step:

$$
\begin{aligned}
\text{minimize:} \quad & c^\top x \\
\text{subject to:} \quad & Ax = b, \\
& -2p^\top x - \lambda_1 - \lambda_2 = \mathrm{r} - 1, \\
& -2\delta p^\top x - (-2p^\top x^* - \mathrm{r} - 1)\lambda_1 + (-2p^\top x^* - \mathrm{r} - 1)\lambda_2 = \delta(\mathrm{r}+1), \\
& \delta Q x - Q x^* \lambda_1 + Q x^* \lambda_2 = 0, \\
& \lambda_1, \lambda_2 \geq 0, \\
& x_j \in \mathbb{Z} \quad j \in J \subseteq N.
\end{aligned}
$$

When $Q \succeq 0$, there is a risk of having a point where one of the frames cannot be obtained exactly, such as $x = (0, 2)$ for $x_1^2 - x_2 \leq 0$. In this special case, one of the λ values gets a fixed value. In our example, λ_2 gets value 1 since we have

$$
\begin{aligned}
\lambda_1 + \lambda_2 - x_2 &= 1, \\
\lambda_1 - \lambda_2 - x_2 &= -1, \\
x_1 &= 0,
\end{aligned}
$$

in the (MIPR) problem as constraints. The free Jordan value λ_1 is used as a magnitude to x_2; hence we end up with a fixed point (0,0) and an unbounded direction. See Figure 4.7. For a bounded case in the same constraint, consider $x = (0.5, 1.2)$. This gives us two endpoints on the boundary of the quadric, $(1.219, 1.487)$ and $(-0.819, 0.672)$. Notice that both frames pass through the focus of the parabola, $(0, 1)$.

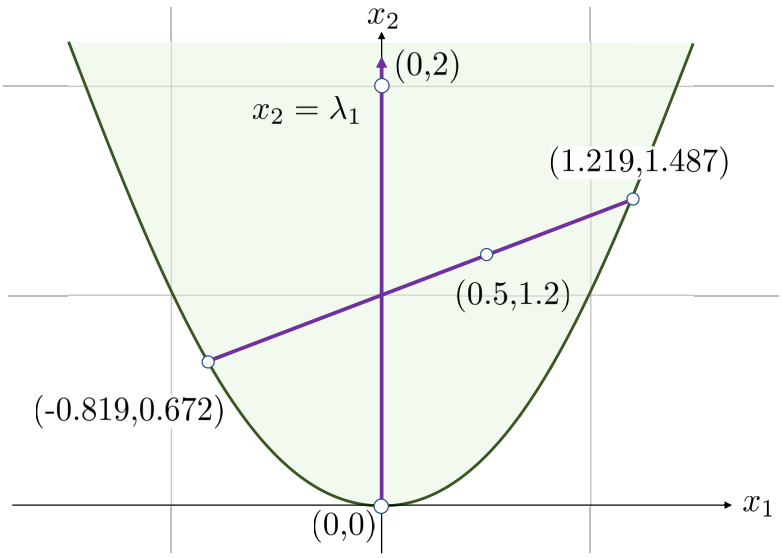


Figure 4.7: Bounded and unbounded Jordan frames in a positive semidefinite case.

4.3 Numerical results

4.3.1 Implementation and test set

For computational experiments, we implemented all heuristics in MATLAB 2014b. The SOCO problems are solved with MOSEK 7.1, and the MILO problems are solved with CPLEX 12.7 via their respective MATLAB API. Problems are solved on an AMD Opteron 2.0 GHz with 8GB of memory.

One of the most used test sets for MISOCO instances is CBLIB [50]. We tested our heuristics on all MISOCO instances in this test set; however, because the memory is limited to 8 GB, only 1322 out of 1754 test problems are included in the numerical experiments. We also converted 6 convex binary quadratic optimization problems from the QPLIB test set [51] into MISOCO using MOSEK and applied our heuristics on these problems as well. The problem types and corresponding cone sizes are shown in Table 4.1.

		Variables		Integers		Cones		Cone sizes	
Pr. Types	#P	Min	Max	Min	Max	Min	Max	Min	Max
ck	90	611	3271	25	75	10	20	27	77
classical	399	146	356	20	50	1	1	21	51
estein	9	125	246	9	18	9	18	3	3
pp	3	72	702	10	100	10	100	3	3
robust	400	198	468	21	51	2	2	22	52
shortfall	400	194	464	21	51	2	2	21	51
sssd	14	273	785	72	264	12	24	3	3
turbine	7	121	512	11	56	25	119	3	3
QPLIB	6	3033	13538	20	400	1	1	802	4502
Summary	1328	72	13538	9	400	1	119	3	4502

Table 4.1: Details of the problem test set.

Table 4.1 shows a huge variety of problems, where some problems have a few relatively big cones (QPLIB) and some problems have multiple small cones (`pp` and `turbine`). All problems are converted to the standard MISOCO form using MOSEK. After conversion, only three problems in the test set (`turbine07, turbine07_aniso` and `turbine54`) have integer variables that appear in multiple-dimensional cones. All of these problems have 11 integers and all of them are leading variable in their cones. No problems in the standard form have integer variables as an in-cone variable, but it is likely that integer variables are related to in-cone variables in some of the test problems.

4.3.2 Efficiency of the heuristics

First, we experimented with the heuristics to see how often they provide a feasible solution and how many iterations it takes to produce a feasible solution. Table 4.2

shows how many iterations it takes for heuristics to generate the first feasible solution for MISOCO problems. We limited the number of MILO problems to be solved by 10. For the hybrid strategy, we allocate three out of 10 MILO problem budget to primal rounding heuristic and the remaining to dual rounding heuristic.

The most spectacular results in Table 4.2 are the percentage of problems where the primal rounding heuristic provided a solution and the number of iterations to generate them. The primal rounding heuristic fails to find a feasible solution for only 19 out of 1329 problems. Out of the 98.5% of problems that the primal rounding heuristic provided a solution for, a feasible solution is obtained for 48% and 96% problems in only one and two iterations, respectively. This is a remarkable result for a primal heuristic in general. Moreover, the hybrid strategy fails only for three problems in total.

In terms of feasibility, the dual rounding heuristic provides a feasible solution for 78% of all test problems, where it provided a solution in one iteration for 33% and in two iterations for 44% of all test problems. The primal-dual rounding heuristic provides a feasible solution for 80% of all test problems with 65% feasible in two iterations. The primal-dual rounding heuristic is expected to perform between the primal rounding heuristic and the dual rounding heuristic, since the iteration budget is essentially halved between these two except minor differences.

As discussed after introducing the primal rounding heuristic, an interesting result can be seen from the Table 4.2 for `sssd` problems. The primal heuristic fails to find a feasible solution in 10 iterations for all problems in this set. Coincidentally, the dual rounding heuristic and also the primal-dual rounding heuristic provide a feasible solution in only one iteration for all problems in this class.

One can conclude that there is little chance for the primal rounding heuristic to provide a solution after two iterations. However, the dual rounding and the primal-dual rounding heuristics may provide a solution in subsequent iterations.

Overall, there is only one problem, `robust_50_51`, where all four heuristics

		# Iters											
Heur	P.Type	1	2	3	4	5	6	7	8	9	10	Failed	Total
P		647	628	28	2	3			1		1	18	1328
	ck	90											90
	classical	5	393	1									399
	estein	9											9
	pp		2	1									3
	robust	136	231	24	2	3			1		1	2	400
	shortfall	400											400
	sssd											14	14
	turbine	2	2	2								1	7
	QPLIB	5										1	6
D		445	146	73	80	50	57	52	46	50	44	285	1328
	ck	10	15	13	8	8	5	6	9	3		13	90
	classical	140	42	19	30	15	19	16	9	18	14	77	399
	estein	9											9
	pp		2					1					3
	robust	120	40	24	33	16	16	19	20	17	13	82	400
	shortfall	142	45	17	8	11	17	10	8	12	17	113	400
	sssd	14											14
	turbine	4	2		1								7
	QPLIB	6											6
PD		647	230	8	68		36	2	53	1	23	260	1328
	ck	90											90
	classical	5	138		42		18		30		15	151	399
	estein	9											9
	pp			3									3
	robust	136	73	5	25		18	2	23	1	8	109	400
	shortfall	400											400
	sssd		14										14
	turbine	2	4		1								7
	QPLIB	5	1										6
HS		647	629	27	16	2	2			2		3	1328
	ck	90											90
	classical	5	393	1									399
	estein	9											9
	pp		3										3
	robust	136	231	24	2	2				2		3	400
	shortfall	400											400
	sssd				13		1						14
	turbine	2	2	2			1						7
	QPLIB	5			1								6

Table 4.2: Number of iterations where the first feasible solution to MISOCO is reported.

failed to provide a solution with the given iteration limit. The primal rounding heuristic, the dual rounding heuristic, and the primal-dual rounding heuristic are able to find a solution for this problem in 12 iterations, 19 iterations, and 19 iterations, respectively. To sum up, all heuristics, especially the primal rounding heuristic, work well in practice in terms of finding solutions in a few iterations. This is a remarkable performance for heuristics of this type.

4.3.3 Quality of the provided solutions

In this part, we evaluate the quality of the provided solution at the termination. Notice that all heuristics may terminate early if the objective value of the generated solution is equal to the global lower bound, hence proving its optimality. This is more likely for the dual and the primal-dual rounding heuristics, since they keep generating new lower bounds for the problem over iterations.

Table 4.3 shows detailed performance of the heuristics over problem types. In the table, number of problems (#P), number of problems solved (#S), average reported gap (RG), average true gap (TG), average number of iterations to reach the first feasible solution (IFS), average number of MILOs solved (#MILO), average number of SOCOs solved (#SOCO), and the total number of instances where the best solution is provided by the primal (FR) problem, the dual (FR) problem, and the (MIDR) problem are presented. It is apparent that the reported gap of the primal rounding heuristic is significantly higher compared to its true gap from the optimal solution for some problem types, `estein`, `turbine`, and `QPLIB`. This is mainly due to the gap between the solution of the continuous relaxation and the solution of the MISOCO problem.

Another interesting result is that the dual rounding heuristic provides an optimal solution for `ck` problems, whenever feasible. Since (FR) fails to find a solution in most of the cases, it is very likely that integer variables are directly related to in-cone variables.

P.Type	Heur	#P	#S	RG	TG	IFS	#MILO	#SOCO	FR-P	FR-D	MIDR
ck	P	90	90	75.36%	71.67%	1.00	10.00	20.00	90	0	0
	D	90	77	0.00%	0.00%	4.18	5.02	4.22	0	5	72
	PD	90	90	35.94%	35.90%	1.00	7.36	6.79	37	3	50
	HS	90	90	38.59%	38.54%	1.00	8.11	9.57	41	2	47
classical	P	399	399	10.28%	8.56%	1.99	9.80	18.43	399	0	0
	D	399	322	20.86%	19.52%	3.34	10.00	10.00	0	322	0
	PD	399	248	16.44%	15.03%	3.82	9.91	6.84	117	131	0
	HS	399	399	15.04%	13.42%	1.99	9.83	10.71	366	33	0
estein	P	9	9	99.48%	0.09%	1.00	10.00	20.00	9	0	0
	D	9	9	0.15%	0.00%	1.00	9.11	9.11	0	9	0
	PD	9	9	1.69%	0.00%	1.00	10.00	10.00	2	7	0
	HS	9	9	1.26%	0.04%	1.00	9.56	11.56	3	6	0
pp	P	3	3	0.22%	0.13%	2.33	10.00	19.00	3	0	0
	D	3	3	1.27%	1.26%	3.67	9.33	9.00	0	2	1
	PD	3	3	0.00%	0.00%	3.00	10.00	8.67	2	0	1
	HS	3	3	33.33%	33.34%	2.00	10.00	11.00	3	0	0
robust	P	400	398	10.84%	10.34%	1.79	7.65	14.19	398	0	0
	D	400	318	20.30%	19.91%	3.64	9.97	9.97	0	318	0
	PD	400	291	19.43%	19.07%	2.72	8.18	5.92	215	76	0
	HS	400	397	11.59%	11.09%	1.78	7.64	8.52	370	27	0
shortfall	P	400	400	1.65%	1.49%	1.00	9.72	19.44	400	0	0
	D	400	287	1.70%	1.56%	3.08	10.00	10.00	0	287	0
	PD	400	400	1.41%	1.25%	1.00	9.76	9.76	340	60	0
	HS	400	400	1.44%	1.28%	1.00	9.73	11.71	365	35	0
sssd	P	14	0	-	-	-	10.00	10.00	0	0	0
	D	14	14	0.00%	0.00%	1.00	8.29	7.29	0	6	8
	PD	14	14	0.22%	0.06%	2.00	10.00	8.86	7	7	0
	HS	14	14	3.33%	0.95%	4.14	10.00	8.93	7	7	0
turbine	P	7	6	43.79%	4.51%	2.00	10.00	17.43	6	0	0
	D	7	7	0.00%	0.00%	1.71	4.00	3.57	0	5	2
	PD	7	7	0.00%	0.00%	2.00	4.86	3.57	3	2	2
	HS	7	7	0.00%	0.00%	2.57	5.71	5.86	4	1	2
QPLIB	P	6	5	90.15%	73.91%	1.00	10.00	18.17	5	0	0
	D	6	6	70.40%	46.53%	1.00	10.00	10.00	0	6	0
	PD	6	6	70.42%	46.51%	1.17	10.00	8.83	1	5	0
	HS	6	6	72.69%	51.02%	1.50	10.00	11.00	1	5	0

Table 4.3: Detailed performance of the heuristics on problem types.

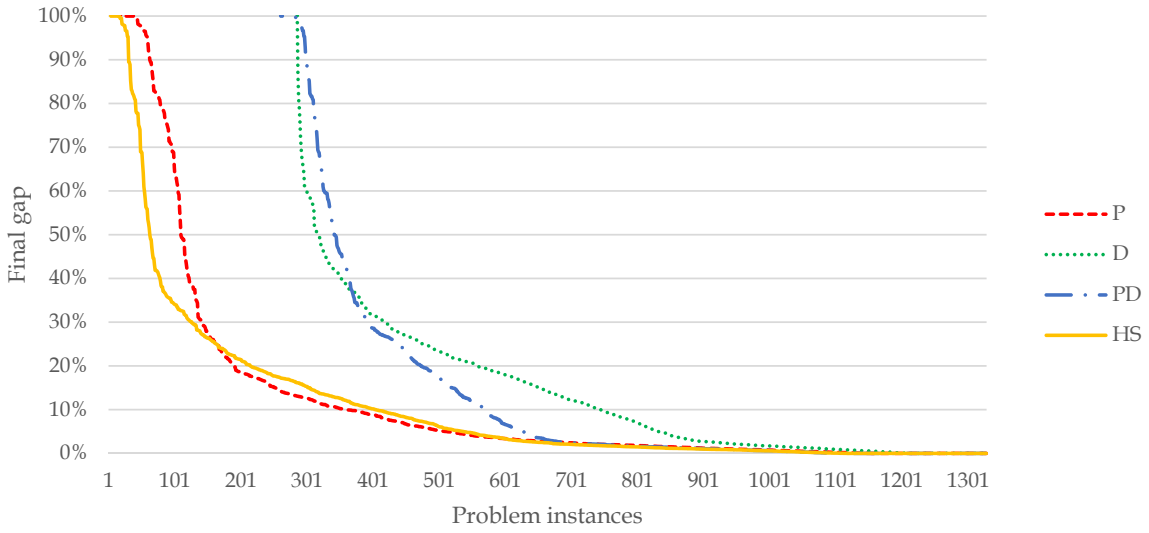


Figure 4.8: Comparison of gaps to the true optimal across heuristics.

See Figure 4.8 for a representation of the gaps to the true optimal solution across all problems. Problems are sorted based on the gap provided by the heuristic, and the plots are shifted to show failed problems. Also see Figure 4.9 for a comparison of gap to the true optimal with the first feasible solutions versus the number of iterations to first feasible solution. We can conclude that the primal rounding heuristic provides smaller gaps in fewer steps despite failing to provide a solution for certain problem types. The dual rounding heuristic usually takes more iterations, but the gap provided is usually close to the true optimal solution, unlike the primal rounding heuristic where many solutions provide a gap close to 100% at the first feasible solution. The optimal strategy is likely a combination of primal and dual heuristics on average, but it is up to the user's knowledge to find the best combination. See Figure 4.10 for a comparison of the gap to the true optimal with the best solution obtained versus the number of iterations to get the best feasible solution. It is apparent that having multiple iterations helps reduce the gap significantly. The solution provided by the heuristic keeps getting better as we keep iterating. The HS dominates other heuristics in terms of final gaps.

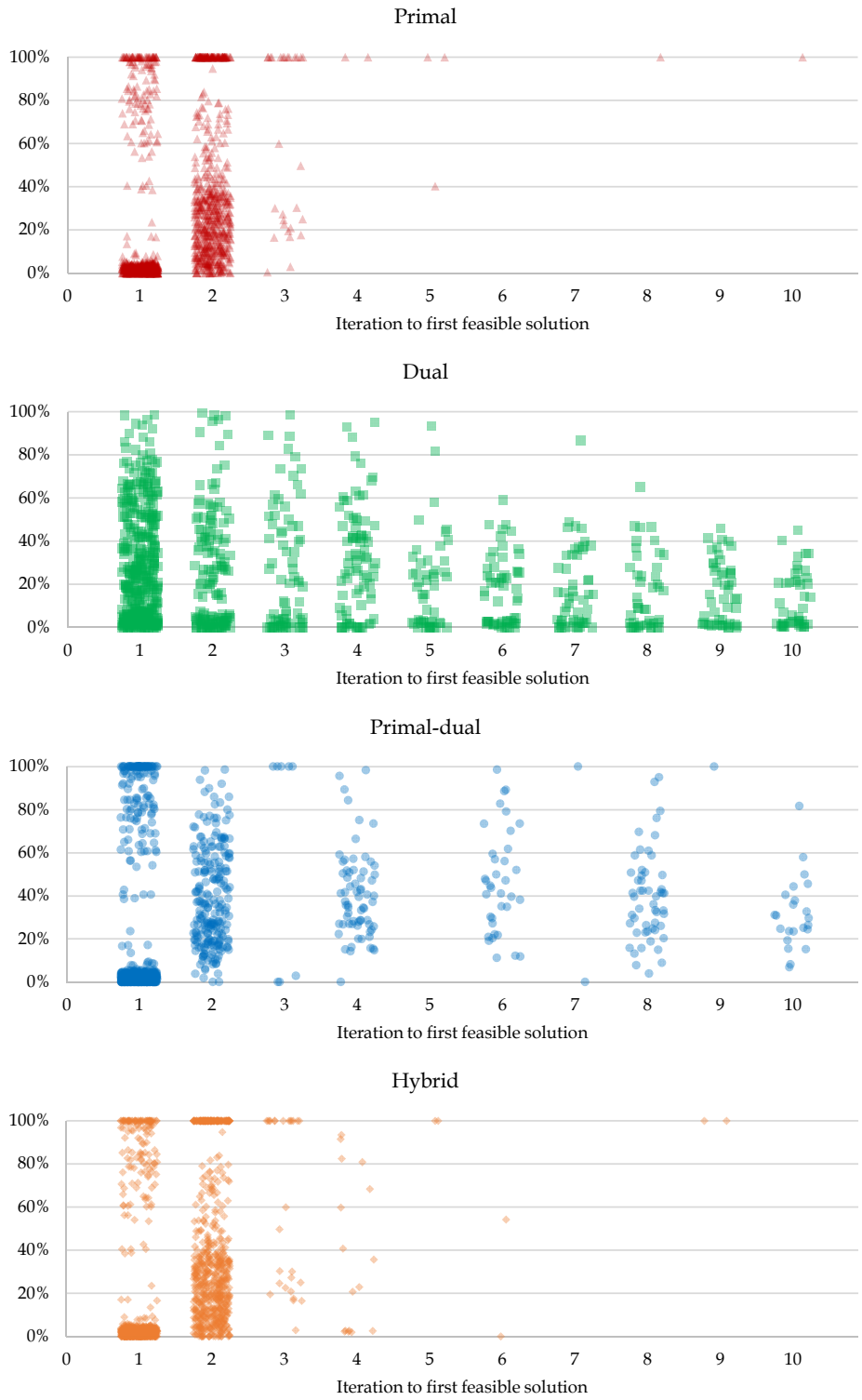


Figure 4.9: Gap to the true optimal versus the number of iterations to first feasible solution for each heuristic.

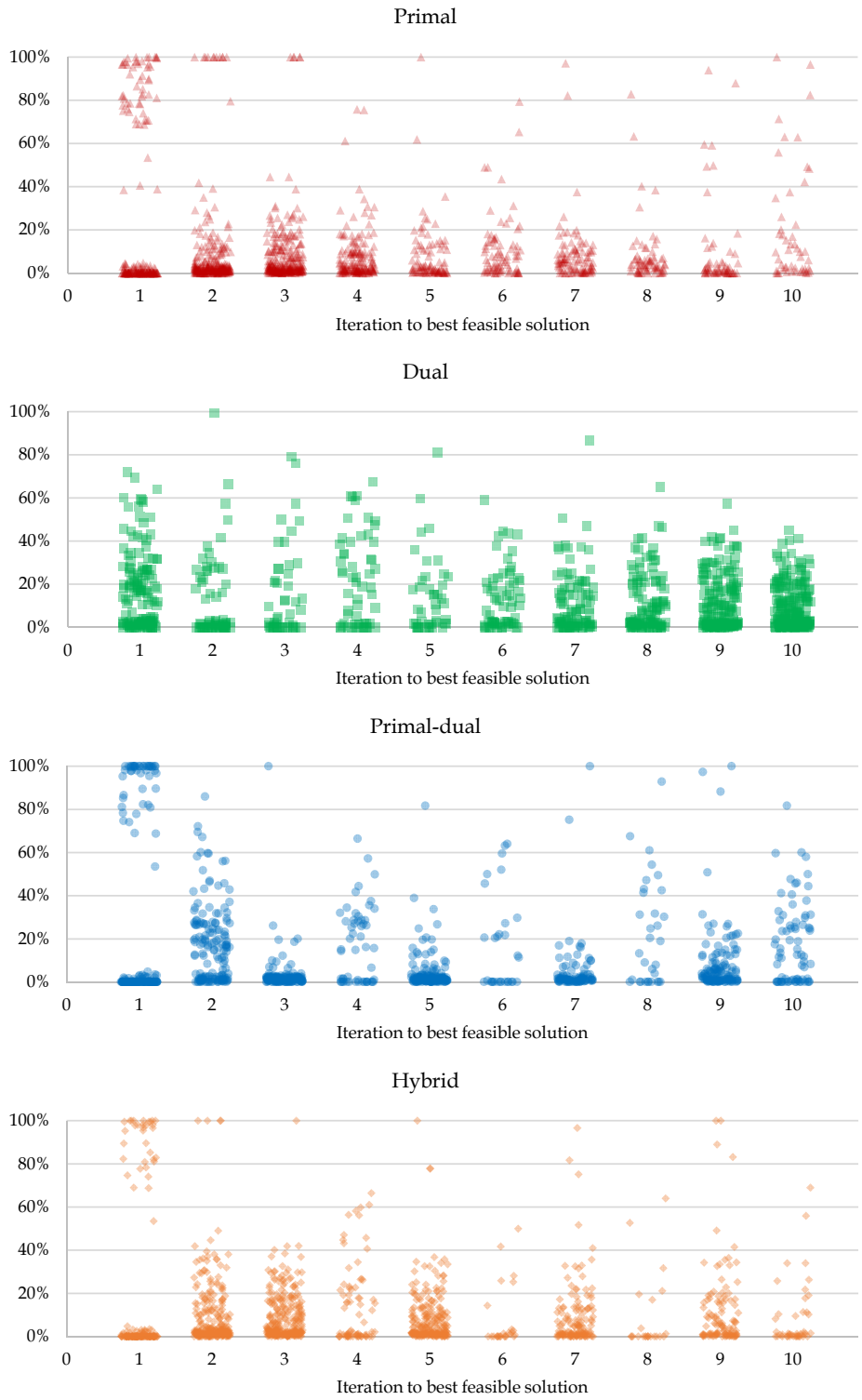


Figure 4.10: Gap to the true optimal versus the number of iterations to best feasible solution for each heuristic.

In some cases, the heuristic stops early and returns the solution obtained if the objective value is equal to the best known lower bound. In these cases, we prove the optimality of the solution. Table 4.4 shows the number of instances and iterations where the true optimal solution is obtained. Note that there are instances where an optimal solution is found but cannot be proved because the lower bound is not sufficient, which is a common case for the primal rounding heuristic. These instances are not included in this table.

The primal-dual heuristic is capable of finding more optimal solutions because primal and dual rounding heuristics work together to provide upper and lower bounds, respectively. This result verifies our intuition about the hybrid strategy.

4.3.4 Effect of iterations on solution quality

One can terminate the heuristics whenever a feasible solution is obtained. However, taking more iterations often provides a much better solution.

See Figure 4.11 for a scatter plot of the gap of the first feasible solution and the final solution after a maximum of 10 MILO problems.

These plots show that having more iterations helps the primal rounding heuristic very significantly, especially for cases where the initial gap is around 100%. The effect of multiple iterations is very significant for the primal-dual rounding heuristic, where the gap to true optimality decreases from the 60—100% interval to 0%.

The solutions obtained after having even a few iterations are significantly better. Even for the problems where the first feasible solution is obtained in the first iteration, the gap to the true optimal is 33%, 64%, 51%, and 36% better when we obtain the best known solution in the second iteration for primal, dual, primal-dual rounding heuristics, and the hybrid strategy, respectively. Having more iterations gradually improves the gap compared to the first feasible solution. On average, the primal rounding heuristics provide 54% better gap, while the number is 45% for the dual rounding and 54% for the primal-dual rounding. Overall, having multiple iterations

		# Iters										
Heur	P.Type	1	2	3	4	5	6	7	8	9	10	Total
P		107	13	4	3	1		2			2	132
	classical	5	2	1	2						2	12
	robust	93	8	3		1		2				107
	shortfall	9	3		1							13
D		14	19	13	11	9	8	12	13	7	6	112
	ck	10	15	13	8	8	5	6	9	3		77
	estein	2	2		2	1						7
	pp								1			1
	robust						1	2		2	5	10
	sssd						2	4	3	2	1	12
	turbine	2	2		1							5
PD		107	14	13	16	2	12	3	8		9	184
	ck		10		15	1	12		8		8	54
	classical	5						1				6
	estein		2									2
	pp										1	1
	robust	91		9		1		2				103
	shortfall	9		4								13
	turbine	2	2		1							5
HS		109	16	3	18	1	13	6	15	2	7	190
	ck				15		12	1	15	1	7	51
	classical	5	2	1				1				9
	estein		1		1							2
	robust	93	10	2				4				109
	shortfall	9	3			1				1		14
	turbine	2			2		1					5

Table 4.4: Number of instances where an optimal solution is found.

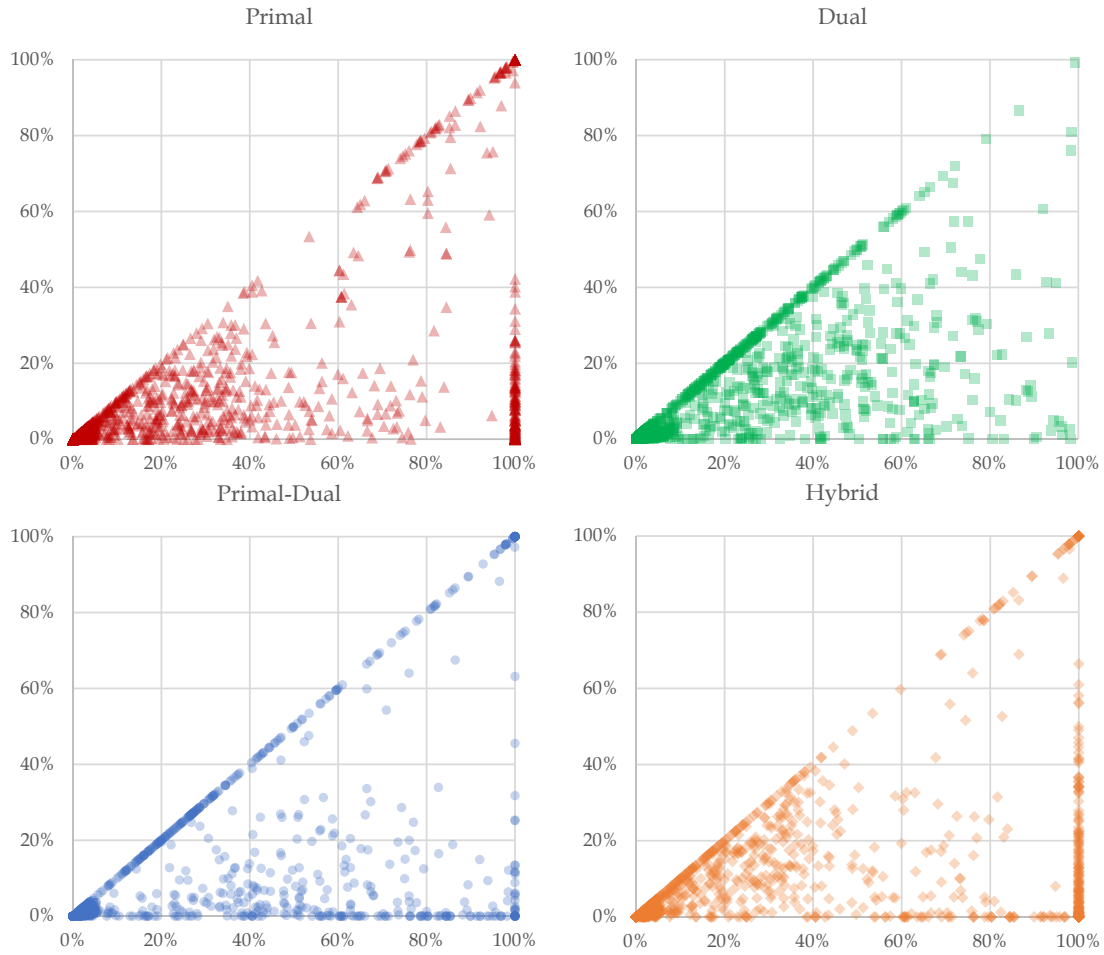


Figure 4.11: Final gap to the true optimal (vertical axis) versus the first gap (horizontal axis) for each instance.

results in an average of 52% better gap at return. These numbers show the significance of running the heuristic for a few iterations.

4.4 Conclusions and future work

In this chapter, we presented four novel rounding heuristics for MISOCO problems. The proposed heuristics are specific to MISOCO and provide major contributions to the existing body of research. As the first of its kind, the conic rounding heuristics

complement the theoretical developments in the MISOCO area and their implementations provide a cost-efficient and novel strategy for finding feasible solutions.

Based on our computational results on the CBLIB and QPLIB test sets, HS finds a feasible solution for almost all test problems. This is a significant result for a heuristic method, it would be significant even for linear optimization. Overall, using a budget of 10 MILO problems, at least one of the heuristics found a feasible solution to 1327 out of 1328 total problems.

This study is an important step to open new research directions for future studies. Due to the close relationship between SOCO and SDO, these heuristics can also be applied to SDO problems. However, it might be expensive to obtain eigenvectors for larger and dense matrices for SDO problems. Therefore, more work is needed to develop efficiently computable versions of these heuristics for SDO problems.

Chapter 5

Effects of disjunctive conic cuts within a branch-and-conic-cut algorithm to solve asset allocation problems

5.1 Introduction

The interest in MISOCO has been growing recently both in academia and industry. Undoubtedly, the application-driven community has helped to speed up the theoretical and computational developments on the MISOCO technology. However, a gap exists between what the solvers offer today and the methodological advances that we have recently achieved. To close the gap between theoretical developments and practical implementations, we studied the effectiveness of recently developed disjunctive conic cuts (DCC) and disjunctive cylindrical cuts (DCyC) [16–18] for MISOCO. More specifically, we applied these cuts to the asset allocation problems (AAPs) in a BCC framework.

The motivation for this chapter comes from the recent methodological

developments in the MISOCO field. These developments have shown promising results to provide faster solutions to MISOCO problems, which are of great interest for practitioners. Most of the theoretical effort on the MISOCO field has focused on generating valid inequalities. As mentioned in Section 2.2.2, recent developments include cuts for binary conic optimization [35], conic mixed-integer rounding cuts [10], intersection cuts for mixed-integer conic quadratic sets [9], separation for conic knapsack constraints [12], split cuts for mixed-integer conic quadratic problems [79], and disjunctive cuts for cross-sections of the second-order cones [110]. Finally, one of the most significant development for MISOCO solution algorithms is the discovery of disjunctive conic cuts (DCC) and disjunctive cylindrical cuts (DCyC) [16–18, 54]. In general, a conic cut is a cone that cuts off some non-integer solutions but none of the feasible integer solutions. Under mild assumptions, adding DCCs and DCyCs gives the convex hull of a closed convex set and its intersection with a linear disjunction. These are shown to be the tightest cuts that can be generated for such disjunctions.

Beyond the theoretical benefits of DCCs and DCyCs to obtain tighter formulations, providing strategies for efficient implementation has utmost importance for research on solution techniques. For this reason, we are interested in exploring the effect of DCCs and DCyCs on real-life applications. Recent work has laid down strong methodological foundations for this family of cuts for MISOCO problems. However, there is much less research on implementation challenges and the effects of DCCs and DCyCs in practice within a full MISOCO solution framework. Even for general conic cuts, previous studies do not provide enough computational results. In these work, cuts are added only at or very close to the root note [10, 35] or not implemented at all [9, 11]. Turning our interest to DCCs, Góez [54] provides preliminary results where DCCs are added both at the root node and inside the BCC tree. Presented by Bonami et al. [29], constrained layout problems have convex quadratic constraints, which can be written as a SOC. Góez

[54] showed that binary variables in these convex quadratic constraints can be used as a disjunction to generate DCCs, which are tighter than the original constraint. Generating these DCCs and replacing the original constraints with them as a preprocessing step reduced the solution time significantly in almost all cases. The author also presented experiments with randomly generated problems, where DCCs are added within the BCC tree by using simple decisions. Promising results of these experiments motivated us to investigate further for a better management of DCCs within a BCC framework. See Bulut [31] for the implementation of various conic cuts inside a BCC framework based on polyhedral relaxations.

Our goal in this chapter is to enhance the understanding on how to use the novel DCC technology. In general, one can generate DCCs and DCyCs by using the disjunctions obtained by the interim solutions in a BCC algorithm. We demonstrate the effectiveness of DCCs and DCyCs within a BCC framework. For this purpose, we revisit the design of most major components of a BCC framework: branching, cutting, and cut management. Also, we explore different decisions in these elements to maximize the benefit gained from the DCCs and DCyCs.

Developing a general purpose BCC algorithm that includes DCCs and DCyCs requires detailed work on the implementation. Therefore, we limit the scope of this chapter to a specific problem class. We derive DCCs and DCyCs for AAPs, which are variations of Markowitz' classical portfolio optimization model [75]. We study an AAP formulation that minimizes the risk of the portfolio while satisfying a prescribed return level. The practical setting of this problem for different markets requires some additional constraints. For instance, in the business market, companies often buy stocks in large batches for liquidity [28]. Such constraints require integer variables in the optimization model. Bonami and Lejeune [28] and Saglam [93] reviewed various similar constraints which arise in portfolio optimization that require integer and binary variables. In all these cases, the common denominator is that the resulting optimization models are MISOCO

problems. Here, we present results for four of these MISOCO problems. These results can be extended to other problems formulated as MISOCO. Benefits of using these cuts in a BCC framework can be observed as lower number of nodes in the BCC tree, lower total solution time, and higher numerical accuracy in some cases.

The rest of the chapter is structured as follows. In Section 5.2, we provide background information about mixed-integer AAP models. This is followed by the description of our BCC framework and the cut generation process in Section 5.3. Cut addition, branching, and searching strategies for our BCC framework are also discussed in Section 5.3. We present our experiments, computational results, and their interpretation in Section 5.4. Finally, we summarize our findings in Section 5.5.

5.2 MISOCO for AAPs

The classical Markowitz mean-variance portfolio optimization problem for n risky assets is formulated as follows:

$$\begin{aligned} \text{minimize:} \quad & x^\top Q x & & & \\ \text{subject to:} \quad & a_0 x_0 + a^\top x & = & \text{r} & \\ & x_0 + \sum_{i=1}^{n} x_i & = & 1 & \\ & x_i & \geq & 0 \qquad & i = 1, \ldots, n, \end{aligned} \tag{5.1}$$

where $Q \in \mathbb{R}^{n \times n}$ is a positive definite variance-covariance matrix, r is the target return level from the investment, $a \in \mathbb{R}^n$ is a vector of the expected returns a_i for each risky asset, and $x \in \mathbb{R}^n$ represents the proportion of the capital invested in each risky asset. Also, x_0 and a_0 denote the fraction of capital invested in the money market and its expected return level, respectively. Money market investment is usually a low-return but risk-free investment type [28].

The AAP is a variation of the portfolio optimization problem, where one allocates a given amount of capital into various types of investments, such as

equities, bonds, currencies, and cash to achieve a certain return level while minimizing the risk. Notice that until now there are no discrete variables in the problem formulation. In the rest of this section, we describe three different AAP formulations that require discrete variables in the problem formulation, as shown by Bonami and Lejeune [28], Cornuejols and Tütüncü [38], and Saglam [93].

5.2.1 Round-lot-constrained AAP

Our first type of mixed-integer AAP formulation appears due to a special set of constraints, called round-lot or minimum transaction lot constraints [28, 74]. In financial settings, large investors often buy risky assets in large batches, which are called even lots. In the business market, trading even lots is considered to be easier than trading small batches.

Before presenting the model, we need to define the relation between the even lots and the vector x. Denote c_i as the fixed batch size and g_i as the market value of each asset i. Let d denote the total capital, and define variable z_i as the number of even lots needed to buy of asset i. Then, the round-lot relationship between x_i and z_i is defined as

$$x_i = \frac{c_i g_i}{\mathrm{d}} z_i, \quad i = 1, \ldots, n, \tag{5.2}$$

which defines the proportion of capital invested in asset i in terms of z_i. When discussing this problem, Bonami and Lejeune [28] simplified the presentation by considering a common batch size for all assets.

After adding the round-lot restriction (5.2) to the models, the expected return may not be satisfied as an equality. However, one can still aim at a return level r that

is a lower bound for the selected portfolio. As a result, we can write the problem as

$$\begin{array}{lrcll} \text{minimize:} & x^\top Q x & & & \\ \text{subject to:} & a_0 x_0 + a^\top x & \geq & \mathrm{r} & \\ & x_0 + \sum_{i=1}^{n} x_i & = & 1 & \\ & x_i & = & \dfrac{c_i g_i}{\mathrm{d}} z_i & i = 1, \ldots, n \\ 0 \leq & x_i & \leq & 1 & i = 1, \ldots, n \\ & z & \in & \mathbb{Z}_+^n. & \end{array} \tag{5.3}$$

Here the constraints $x_i \leq 1, i = 1, \ldots, n$ are redundant, but they are required when allowing a shorting operation. Shorting is a common practice by which investors can sell financial instruments that they do not own and purchase them at a later time. This practice is applied if the investor expects a decrease in value of the investment. In (5.3) one can allow shorting by dropping the non-negativity on x, the investment fractions, and on z, the number of batches purchased.

We may simplify the discussion further by using (5.2) to express problem (5.3) all in terms of z. Let $b \in \mathbb{R}^n$ be defined as

$$b_i = \frac{c_i g_i}{\mathrm{d}}, \quad i = 1, \ldots, n,$$

which is the constant batch size for investment in asset i due to the round-lot constraint. Let $\hat{Q} = \mathrm{diag}(b)^\top \cdot Q \cdot \mathrm{diag}(b)$, and allow us to introduce a new variable t to move the quadratic objective function to the constraints set. Moreover, note that

$$x_0 = 1 - \sum_{i=1}^{n} x_i,$$

thus x_0 can be substituted out from the formulation. Also, let $\hat{a} = -\mathrm{diag}(a)b + a_0 b$, and $\hat{\mathrm{r}} = a_0 - \mathrm{r}$; then we obtain the following equivalent formulation of (5.3):

$$\begin{array}{lrcll} \text{minimize:} & t & & & \\ \text{subject to:} & \hat{a}^\top z & \leq & \hat{\mathrm{r}} & \\ & b^\top z & \leq & 1 & \\ 0 \leq & b_i z_i & \leq & 1 & i = 1, \ldots, n \\ & z^\top \hat{Q} z & \leq & t & \\ & z & \in & \mathbb{Z}_+^n. & \end{array} \tag{5.4}$$

Note that formulation (5.4) may be written as an equivalent MISOCO problem by replacing the convex quadratic constraint $z^\top \hat{Q} z \leq t$ by its conic equivalent. Since matrices Q and $\hat{Q}$ are symmetric and positive definite, we can replace $\hat{Q}$ by $\hat{Q}^{1/2}\hat{Q}^{1/2}$, where $\hat{Q}^{1/2}$ is the symmetric square root of matrix $\hat{Q}$. If we use the equivalences

$$z^\top \hat{Q} z = \left\| \hat{Q}^{1/2} z \right\|^2, \quad t = \left(\frac{t+1}{2} \right)^2 - \left(\frac{t-1}{2} \right)^2;$$

then constraint $z^\top \hat{Q} z \leq t$ can be replaced by its second order conic form of

$$\sqrt{\left\| \hat{Q}^{1/2} z \right\|^2 + \left(\frac{t-1}{2} \right)^2} \leq \frac{t+1}{2}.$$

We use this transformation in the rest of this section whenever we are going to rewrite a problem as an equivalent MISOCO.

5.2.2 Cardinality and diversification-constrained AAP

Investors may want to limit the number of assets to be invested in for various reasons such as to avoid costs of monitoring [47] and transactions. In this case, the investor's aim is to find the best asset allocation that allows a certain return level to be obtained while keeping the focus on a fixed number of assets. One's motivation to restrain from investing in too many assets at the same time is to avoid situations where a small amount is invested in an asset. This might bring new costs, such as tracking, that are often neglected in the portfolio optimization settings. Bienstock [26] shows that cardinality-constrained quadratic optimization problems are NP-hard.

We model cardinality-constrained AAPs by introducing new binary variables z to the original model (5.1). We ignore the money market investment for simplicity. After moving the objective function into the constrains as we did earlier, and requesting a

minimum return level as a lower bound, we obtain

$$
\begin{array}{lrcll}
\text{minimize:} & t & & & \\
\text{subject to:} & a^\top x & \geq & \mathrm{r} & \\
& \sum_{i=1}^{n} x_i & = & 1 & \\
& x_i & \leq & z_i & i = 1, \ldots, n \\
& \sum_{i=1}^{n} z_i & \leq & k & \\
& x^\top Q x & \leq & t & \\
& x_i & \geq & 0 & i = 1, \ldots, n \\
& z_i & \in & \{0, 1\} & i = 1, \ldots, n,
\end{array}
\tag{5.5}
$$

where k is the number of assets to invest in.

A refinement on (5.5) may be to limit the number of assets bought that are related to each other, which is a common practice in AAPs. This is usually done by classifying assets into sectors or countries and imposing a minimum and maximum number of stocks to invest in for each of these classes. In this way, an investor can prevent buying too many stocks from the same sector, which are known to be related beforehand. This is known as the diversification constraint; it is a general form of the cardinality constraint that considers various subsets. For a subset $\bar{\mathcal{N}}_j \subseteq \mathcal{N} = \{1, \ldots, n\}$ and cardinality k_j, we represent diversification constraints as

$$
\sum_{i \in \bar{\mathcal{N}}_j} z_i \leq k_j \qquad \forall j, \tag{5.6}
$$

where $\cup_{\forall j} \bar{\mathcal{N}}_j = \mathcal{N}$. Thus, a cardinality constraint is simply a specialized diversification constraint when $\bar{\mathcal{N}}_j = \mathcal{N}$.

We now present two alternatives to reformulate (5.5) to take advantage of the DCCs and DCyCs technology. The first alternative is to replace the linear cardinality constraint by the convex quadratic constraint:

$$
\sum_{i=1}^{n} z_i^2 \leq k. \tag{5.7}
$$

Observe that, since z is a binary vector, this reformulation does not change the feasible region of the problem, which gives us a variation of the cardinality-constrained AAP. Hence, we have the following equivalent reformulation for the AAP with a quadratic-cardinality constraint:

$$\begin{array}{lrcll} \text{minimize:} & t & & & \\ \text{subject to:} & a^\top x & \geq & \mathrm{r} & \\ & \sum_{i=1}^{n} x_i & = & 1 & \\ & x_i & \leq & z_i & i = 1, \ldots, n \\ & \sum_{i=1}^{n} z_i^2 & \leq & k & \\ & x^\top Q x & \leq & t & \\ & x_i & \geq & 0 & i = 1, \ldots, n \\ & z_i & \in & \{0, 1\} & i = 1, \ldots, n. \end{array} \tag{5.8}$$

Some of the assets in (5.7) can be left linear in the constraint to obtain the following general form for quadratic-cardinality constraint:

$$\sum_{i \in \mathcal{U}} z_i^2 + \sum_{i \in \mathcal{N} \setminus \mathcal{U}} z_i \leq k, \tag{5.9}$$

where $\mathcal{U} \subseteq \mathcal{N}$. The same, equivalent quadratic reformulation can be applied to each of the diversification constraints (5.6).

The second alternative is obtained when any or all of the bound constraints $x_i \leq z_i$ are replaced by $x_i^2 \leq z_i$. Hence, we obtain the following reformulation:

$$\begin{array}{lrcll} \text{minimize:} & t & & & \\ \text{subject to:} & a^\top x & \geq & \mathrm{r} & \\ & \sum_{i=1}^{n} x_i & = & 1 & \\ & x_i^2 & \leq & z_i & i = 1, \ldots, n \\ & \sum_{i \in \mathcal{U}} z_i^2 + \sum_{i \in \mathcal{N} \setminus \mathcal{U}} z_i & \leq & k & \\ & x^\top Q x & \leq & t & \\ & x_i & \geq & 0 & i = 1, \ldots, n \\ & z_i & \in & \{0, 1\} & i = 1, \ldots, n. \end{array} \tag{5.10}$$

This variation is equivalent to (5.5) due to the constraint that variables z_i are binary.

In the next section; we show how one can generate DCCs for the alternative quadratically-constrained reformulations (5.8) and (5.10).

5.3 Methodology

We present our methodology to solve the AAPs that were introduced in the previous section. We begin with discussing the components of our BCC framework. Then, we describe the cut generation procedure that is used within the BCC framework. Next, we present the cut generation strategies we use for our experiments followed by the branching and searching strategies we test.

5.3.1 Branch-and-conic-cut framework

There are several options to solve AAPs from both the academic side [98, 103] and the commercial side [7, 61, 64]. Unfortunately, none of the off-the-shelf commercial solvers provide a user API to implement nonlinear cuts in their branch-and-cut algorithms. For these reasons, we developed our own BCC framework.

Our implementation is based on the BCC framework shown in Algorithm 2. In the current implementation, we use the IPM solver of MOSEK [7] to solve the SOCO subproblems in the tree formed by the branching process. It is well known that IPMs solve SOCO relaxations in polynomial time. Our main focus in this work is the implementation of DCC and DCyCs in the BCC framework, but note that Algorithm 2 allows the use of other types of conic cuts in the literature. As a subset of conic cuts, linear cuts from the MILO literature can be also used here.

5.3.2 Disjunctive conic and cylindrical cut generation

In this subsection, we show how DCCs and DCyCs may be generated for the problems introduced in Section 5.2. Our goal is to provide the procedure to generate the cuts

for a subproblem in Step 12 of Algorithm 2. We follow the cut generation procedures described by Belotti et al. [16, 17, 18] and Góez [54].

DCyCs for round-lot-constrained AAP

We focus here on how to generate DCyCs for problem (5.4) by using the quadratic constraint

$$z^\top \hat{Q} z \leq t. \tag{5.11}$$

Let z^*, t^* be the optimal solution to the continuous relaxation of a subproblem. For any asset $i = 1, \ldots, n$ where $z_i^* \notin \mathbb{Z}^+$, we can write the following disjunction:

$$z_i \leq \lfloor z_i^* \rfloor \quad \vee \quad z_i \geq \lceil z_i^* \rceil . \tag{5.12}$$

Let us introduce the following notation:

$$w = \begin{bmatrix} t \\ z \end{bmatrix}, \; P = \begin{bmatrix} 0 & 0 \\ 0 & \hat{Q} \end{bmatrix} \text{ and } p = \begin{bmatrix} -1/2 \\ 0_{n\times 1} \end{bmatrix}.$$

Hence, we can rewrite (5.11) in the quadric form as

$$w^\top P w + 2p^\top w \leq 0 \tag{5.13}$$

Inequality (5.13) defines a paraboloid because matrix P has exactly one non-positive eigenvalue, which is zero. Since the quadratic constraint is a paraboloid, we can generate disjunctive cylindrical cuts as described in [16–18, 54].

To simplify the DCyC derivation, we transform the quadric into a standard representation. Using eigenvalue decomposition, the matrix P may be decomposed into $P = V^\top D V$, where D is a diagonal matrix, that is built with the eigenvalues of P, and V is an orthogonal matrix. Note that D is singular because the first element in its diagonal is zero. Let

$$\bar{D} = \begin{bmatrix} 1 & 0 \\ 0 & D_{2:n+1} \end{bmatrix} \quad \text{and} \quad J = \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix};$$

where $\bar{D}$ is simply the matrix obtained after the first diagonal element of D is replaced by 1. By doing that, we obtain a non-singular matrix $\bar{D}$. Let $\overline{V} = \bar{D}^{1/2}V$; then inequality (5.13) can be rewritten as

$$w^\top \overline{V}^\top J \overline{V} w + 2\left(p^\top \overline{V}^{-1}\right)(\overline{V}w) \leq 0.$$

Let $u = \overline{V}w$ and $\bar{p} = \left(p^\top \overline{V}^{-1}\right)^\top$, which allows us to rewrite (5.11) as

$$u^\top J u + 2\bar{p}^\top u \leq 0. \tag{5.14}$$

Now, consider the parallel hyperplanes

$$\mathcal{A}^= = \left\{w \in \mathbb{R}^{n+1} \mid e_i^\top w = \lfloor z_i^* \rfloor\right\} \text{ and } \mathcal{B}^= = \left\{w \in \mathbb{R}^{n+1} \mid e_i^\top w = \lceil z_i^* \rceil\right\},$$

where e_i is the $(i+1)^{th}$ unit vector, taking into account that the first position belongs to variable t in w. Note that $\mathcal{A}^=$ and $\mathcal{B}^=$ are the two hyperplanes that define the boundaries of disjunction (5.12). To keep $\mathcal{A}^=$ and $\mathcal{B}^=$ consistent with the standardization of (5.11), we may use the equality $u = \overline{V}w$. Hence, $w = \overline{V}^{-1}u$, and $\mathcal{A}^=$ and $\mathcal{B}^=$ may be written in terms of u as

$$\mathcal{A}^= = \left\{u \in \mathbb{R}^{n+1} \mid e_i^\top \overline{V}^{-1} u = \lfloor z_i^* \rfloor\right\} \text{ and } \mathcal{B}^= = \left\{u \in \mathbb{R}^{n+1} \mid e_i^\top \overline{V}^{-1} u = \lceil z_i^* \rceil\right\},$$

where $e_i\overline{V}^{-1}$ is the $i+1^{th}$ row of $\overline{V}^{-1}$. Let $h = \left(e_i^\top \overline{V}^{-1}\right)^\top$, and

$$\mathrm{a} = \frac{\lfloor z_i^* \rfloor}{\|h\|} \text{ and } \mathrm{b} = \frac{\lceil z_i^* \rceil}{\|h\|}.$$

Our final step is to identify our DCyC in the uniparametric family of quadratics $\mathcal{Q}(\tau)$ that have the same intersection with $\mathcal{A}^= \cup \mathcal{B}^=$ and quadric (5.14) [16], which is characterized by

$$P(\tau) = J + \tau \frac{h}{\|h\|}\left(\frac{h}{\|h\|}\right)^\top, \qquad p(\tau) = \bar{p} - \tau \frac{\mathrm{a}+\mathrm{b}}{2}\frac{h}{\|h\|}, \qquad \mathrm{p}(\tau) = \tau \mathrm{ab}.$$

Note that $Pw^c = -p$ is not solvable and that the first element of e_i is always zero. Therefore, by setting $\tau = -1$, we obtain a cylinder in the family of quadrics. Specifically, the parabolic cylinder

$$u^\top P(-1)u + 2p(-1)^\top u + \mathrm{p}(-1) \leq 0, \tag{5.15}$$

which cuts off the solution of the continuous relaxation. The cylinder (5.15) is a DCyC, and it may be written in terms of original variables as

$$w^\top \overline{V}^\top P(-1)\overline{V}w + 2p(-1)^\top \overline{V}w + \mathrm{p}(-1) \leq 0.$$

Our round-lot-constrained asset allocation subproblem after a single DCyC is added becomes

$$\begin{array}{lrcll}
\text{minimize:} & t & & & \\
\text{subject to:} & a^\top z & \leq & \hat{\mathrm{r}} & \\
& b^\top z & \leq & 1 & \\
0 \leq & b_i z_i & \leq & 1 & i = 1, \ldots, n \\
& z^\top \hat{Q} z & \leq & t & \\
& \begin{bmatrix} t \\ z \end{bmatrix}^\top \overline{V}^\top P(-1)\overline{V} \begin{bmatrix} t \\ z \end{bmatrix} + 2p(-1)^\top \overline{V} \begin{bmatrix} t \\ z \end{bmatrix} + \mathrm{p}(-1) & \leq & 0 & \\
& z & \in & \mathbb{Z}^n_+. &
\end{array}$$

Notice that the costly part of this cut generation is the eigenvalue decomposition of $\hat{Q}$ at the beginning of the operation. However, the same eigenvalue decomposition can be used for all cuts that are generated on this constraint. Therefore, in our implementation, we apply the eigenvalue decomposition only once and use it everywhere in the BCC tree.

To illustrate DCyCs on round-lot-constrained AAPs, consider the constraint

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}^\top \begin{bmatrix} 8.529 & 5.850 \\ 5.850 & 8.805 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \leq t,$$

which is in the form of (5.11)[1]. Solving the continuous relaxation of the corresponding round-lot-constrained AAP, we get the optimal solution $z_1 = 0.286, z_2 = 0$. Using the parallel disjunction

$$z_1 \leq 0, \qquad \vee \qquad z_1 \geq 1$$

[1]Approximate numbers up to three digits precision are given here.

and following the procedure described in this section, we obtain the following the DCyC:

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}^\top \begin{bmatrix} 3.887 & 5.850 \\ 5.850 & 8.805 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + 4.642 z_1 \leq t.$$

Adding this DCyC to the problem formulation cuts off the continuous relaxation solution. A 3D plot of the original constraint and the corresponding DCyC of this example is given in Figure 5.1. Figure 5.2 shows the projection of the quadratic and DCC onto the $t - z_1$ plane.

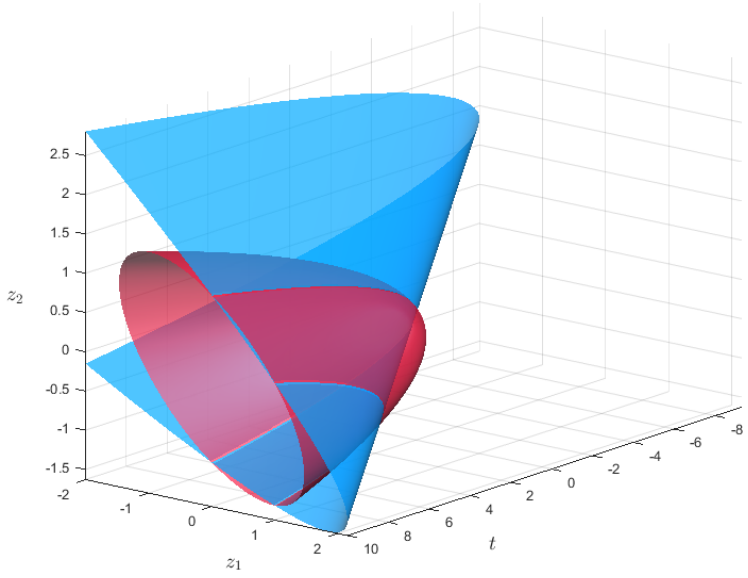


Figure 5.1: Illustrative DCyC on the quadric of an instance of a round-lot-constrained AAP.

DCC for quadratic-cardinality-constrained AAP

For the AAP with quadratic-cardinality constraint (5.8), we can define the disjunctions on the binary variables z. Let z^* be a solution in a subproblem, where $z_i^* \notin \{0, 1\}$, for some $i \in \{1, \ldots, n\}$. We use here a parallel disjunction that is

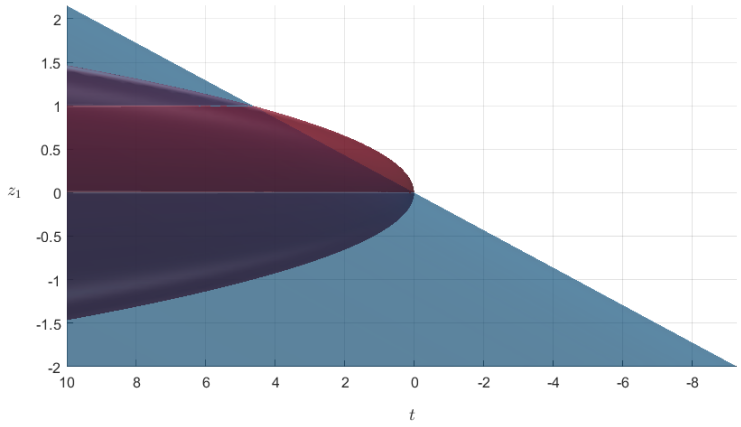


Figure 5.2: Projection of original quadratic and DCyC onto the $t - z_1$ plane. DCyC cuts off some of non-integer points.

defined as

$$z_i \leq 0 \ \vee \ z_i \geq 1.$$

Let us write the constraint $\sum_{i=1}^{n} z_i^2 \leq k$ as

$$z^T z - k \leq 0.$$

Notice that this time our quadric is a ball around the origin with radius $\sqrt{k}$. Hence, we can generate a DCC by using the uniparametric family of quadrics in [16], which in this case is given by

$$P(\tau) = I + \tau e_i (e_i)^\top, \qquad p(\tau) = -\frac{\tau}{2} e_i, \qquad \mathrm{p}(\tau) = -k,$$

where e_i is a unit vector with 1 at the i^{th} position, and 0 elsewhere.

To find the DCC for this constraint, we need to solve the following equation:

$$p(\tau)^\top P(\tau) p(\tau) - \mathrm{p}(\tau) = 0, \tag{5.16}$$

which is a quadratic equation in terms of τ. There are two cones in this family, and the root of (5.16) with the larger value gives us the DCyC that tightens the formulation [16–18, 54]. Using that result we obtain a valid cut for this case when

$$\tau = 2k\left(\sqrt{1-\frac{1}{k}}-1\right).$$

Finally, we can write our DCC as follows:

$$z^\top z + \tau z_j^2 - \tau z_j \leq k.$$

Figure 5.3 illustrates the intersection of the quadratic cardinality constraint $z_1^2 + z_2^2 + z_3^2 \leq 2$ and the generated DCC, which is based on the disjunction $z_1 \leq 0$ or $z_1 \geq 1$. Note that since the z variables are binary, there are a total of n such DCCs.

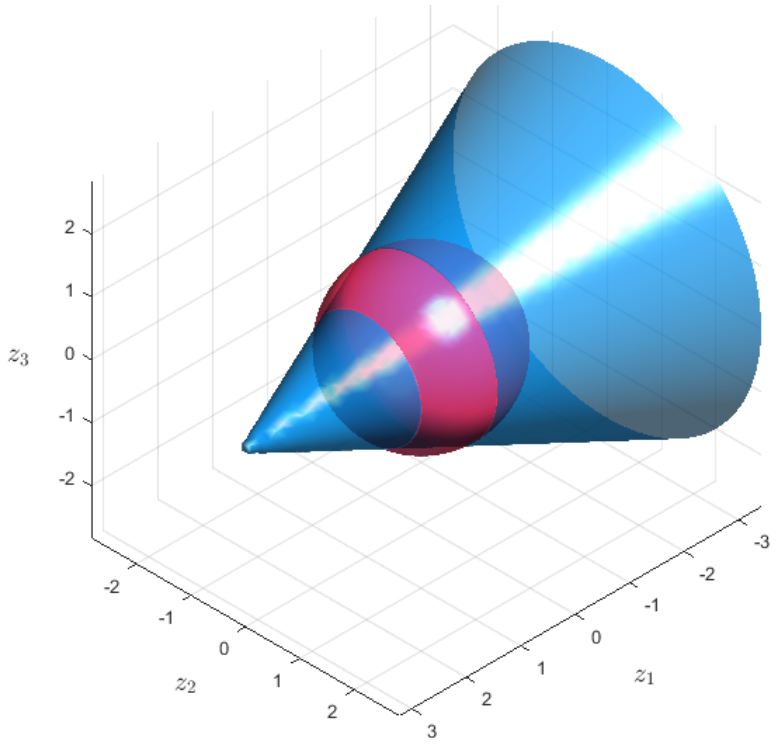


Figure 5.3: DCC on the quadratic cardinality constraint $z_1^2 + z_2^2 + z_3^2 \leq 2$.

Keeping some of the terms linear in the quadratic cardinality constraint (5.9) allows us to derive DCCs for cardinality-constrained AAP of various dimensions. An

important observation is that if we have only one asset in the portfolio (that is, $n = 1$) or if we keep all but one of the assets linear, then the DCC we obtain is simply the original linear cardinality constraint (5.5).

Diversification constraints are similar to the case of cardinality constraints. This case is equivalent to having multiple cardinality constraints on smaller subsets. We follow the DCC generation procedure described by Belotti et al. [16, 17, 18] and Góez [54] for any asset i that belongs to country ℓ, on the diversification constraint

$$\sum_{j \in \bar{\mathcal{N}}_\ell} z_j^2 \leq k_\ell$$

by using the disjunction $z_i \leq 0$ or $z_i \geq 1$. The DCC that corresponds to this case is

$$\sum_{j \in \bar{\mathcal{N}}_\ell} z_j^2 - 2z_i^2 + 2z_i \leq k_\ell.$$

DCCs for quadratic-bound-constrained AAP

The use of quadratic bound constraints allows for a simplified process to derive a DCC. Similar to the previous sections, we start with a disjunction on a binary variable z_i. We generate our DCC on the quadratic cardinality constraint:

$$\begin{bmatrix} z_j \\ x_j \end{bmatrix}^\top \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} z_j \\ x_j \end{bmatrix} + 2 \begin{bmatrix} -0.5 \\ 0 \end{bmatrix} \begin{bmatrix} z_j \\ x_j \end{bmatrix} \leq 0.$$

The shape of the quadric is paraboloid, since since the coefficient matrix of the quadratic term in the constraint has an eigenvalue 0, and the system $Pw = p$ is not solvable for w where

$$P = \begin{bmatrix} 0 & 0 \\ 0 & \hat{\Sigma} \end{bmatrix} \text{ and } p = \begin{bmatrix} -1/2 \\ 0_{n \times 1} \end{bmatrix}.$$

Since we have a finite intersection with the parallel hyperplanes $z_j = 0$ and $z_j = 1$, the DCC in this case is

$$x_j^2 - z_j^2 \leq 0,$$

which is illusrated in Figure 5.4. This constraint is equivalent to the two linear constraints $x_j \leq z_j$ and $x_j \leq -z_j$.

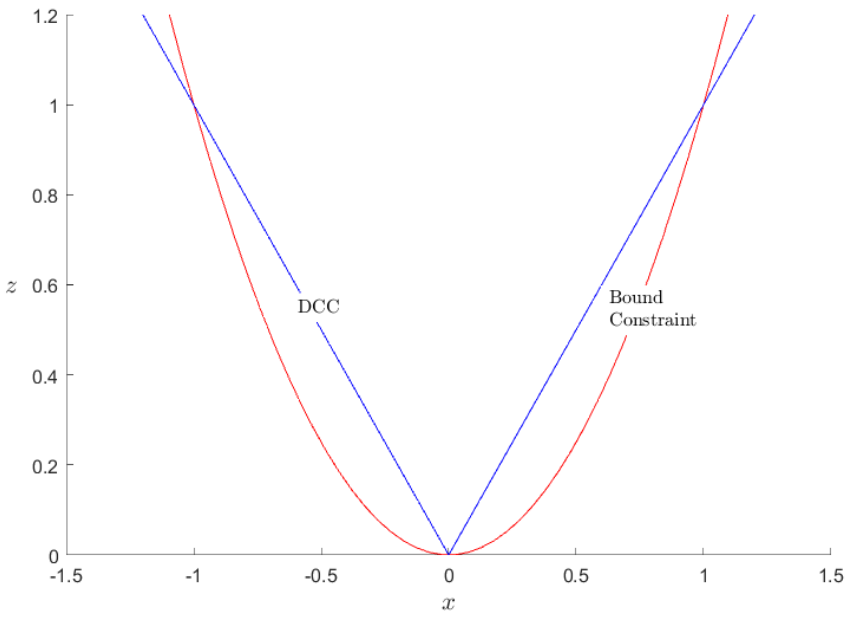


Figure 5.4: DCC for quadratic-bound-constrained AAP.

Note that the DCC makes the quadratic-bound constraints redundant in this problem formulation. Similar to the previous case, there are total of n such DCCs that can be generated in this way. In the actual implementation, the redundant constraints are removed.

5.3.3 Cut management strategies

Adding DCCs and DCyCs requires many decisions inside the BCC algorithm. First, we need to decide when and how many cuts are inserted to the subproblems. Second, we need to decide about the order of cut generation.

We consider five different cut application strategies in our BCC framework.

- The first strategy is to disable all cut generation procedures, hence solving the problem simply with a pure B&B method. This solution strategy is considered as the base case, and all improvements obtained with DCCs and DCyCs are

measured against this strategy. This method is abbreviated as B&B in the numerical results.

- The second cut application strategy is to add a fixed number of cuts to the subproblems until a cut limit is reached on a subproblem. In this method, the order of the cuts that is applied to the subproblem is determined with an ordering rule. Details of this rules are presented in Section 5.3.4. This method is abbreviated as BCC-F in the numerical results.

- The third strategy is to keep adding DCCs and DCyCs as long as the objective is improving sufficiently and then solving the relaxations iteratively. The improvement in the objective is compared to a predefined parameter ϵ. In our experiments, we chose $\epsilon = 10^{-3}$. If the procedure fails to obtain an improvement in the objective value, then the framework proceeds with branching. This method is abbreviated as BCC-I in the numerical results.

- Our fourth method is to add all possible DCCs and DCyCs at the root node. We use this rule for cardinality-constrained and bound-constrained AAPs, where all possible DCCs can be easily obtained at the root node. This method is abbreviated as BCC-R in the numerical results.

- Our final cut application strategy is to produce all the available cuts at a subproblem and then order them in terms of their depth. We use the violation in the generated cut as a measure of its depth. This method is abbreviated as BCC-D in the numerical results.

In the implementation, all the cuts that are generated at a subproblem are inherited by their children. We limit the total number of DCCs in a subproblem by 10. As mentioned earlier, an important decision in the implementation is to choose which DCCs and DCyCs to be added to the subproblem. We ranked variables to branch and generate a cut using four different rules. These rules are presented in the following subsection.

5.3.4 Branching and searching

Branching is one of the most important decisions inside the BCC algorithm. Here we list four different rules for selecting the index $\hat{i}$ of the variable that will be used for branching. We use these branching decision rules in our implementation.

Our first branching method is *most fractional* (MF) branching, which is borrowed from the MILO literature. In this method, one branches using the integer variable whose fractional part is closest to half [3]. This branching rule may be formulated as

$$\hat{i} = \underset{i:z_i \notin \mathbb{Z}^+}{\arg\max} \left\{ \min \left\{ \lceil z_i \rceil - z_i, z_i - \lfloor z_i \rfloor \right\} \right\}.$$

Our second branching rule is called *highest cost* (HC) branching, where the variable with the highest return rate (a) is chosen. We propose this branching rule due to our observation that assets with higher return rates have more impact on the solution when they are fixed. The rule is defined as follows

$$\hat{i} = \underset{i:z_i \notin \mathbb{Z}^+}{\arg\max}\, a_i.$$

The next two branching rules were proposed by Bonami and Lejeune [28] for portfolio optimization problems that have integer variables. The third branching rule uses for branching the asset that has the greatest variance Q_{ii}. This rule is called *idiosyncratic risk* branching (IR), and is defined as

$$\hat{i} = \underset{i:z_i \notin \mathbb{Z}^+}{\arg\max}\, Q_{ii}.$$

The fourth rule is called *portfolio risk* (PR) branching. The main idea in this branching rule is to calculate a score for each variable based on the current condition of the problem. Let us simplify the portfolio optimization problem as

$$\begin{array}{rrcl} \text{minimize:} & z^\top \hat{Q} z & & \\ \text{subject to:} & Az & \leq & b \\ & z & \geq & 0, \end{array} \tag{5.17}$$

where the corresponding Lagrangian of (5.17) is

$$\mathcal{L}_\lambda(z) = z^\top \hat{Q} z + \lambda^\top (Az - b).$$

Displacement in z changes the Lagrangian as much as

$$\begin{aligned}\mathcal{L}_\lambda(z+\epsilon) - \mathcal{L}_\lambda(z) &= (z+\epsilon)^\top \hat{Q}(z+\epsilon) + \lambda^\top (A(z+\epsilon) - b) - z^\top \hat{Q} z - \lambda^\top (Az - b) \\ &= \epsilon^\top \hat{Q} \epsilon + \left(2z^\top \hat{Q} + \lambda^\top A\right) \epsilon.\end{aligned}$$

At an optimal solution z^*, Karush-Kuhn-Tucker conditions satisfy

$$\nabla \mathcal{L}_\lambda(z^*) = 2(z^*)^\top \hat{Q} + \lambda^\top A = 0,$$

and hence

$$\mathcal{L}_\lambda(z^* + \epsilon) - \mathcal{L}_\lambda(z^*) = \epsilon^\top \hat{Q} \epsilon.$$

Suppose we have a fractional value for a variable z_i. Branching on z_i creates two branches and for branches with $z_i \leq \lfloor z_i^* \rfloor$ and $z_i \geq \lceil z_i^* \rceil$, the changes in the Lagrangian are

$$\begin{aligned}\delta_j^- &= (z_i^* - \lfloor z_i^* \rfloor) e_i^\top \hat{Q} (z_i^* - \lfloor z_i^* \rfloor) e_i = (z_i^* - \lfloor z_i \rfloor)^2 Q_{ii} \\ \delta_i^+ &= (\lceil z_i^* \rceil - z_i^*) e_i^\top \hat{Q} (\lceil z_i^* \rceil - z_i^*) e_i = (\lceil z_i \rceil - z_i^*)^2 Q_{ii},\end{aligned}$$

respectively, where e_i is the i^{th} unit vector. The final score associated with variable z_i is calculated as the combination of these two values, such as

$$\delta_i = \alpha \min\left\{\delta_i^-, \delta_i^+\right\} + \beta \max\left\{\delta_i^-, \delta_i^+\right\},$$

where α and β are weights of the minimum and the maximum of δ_i^- and δ_i^+, respectively. Finally, we choose the variable with the highest δ:

$$\hat{i} = \underset{i: z_i \notin \mathbb{Z}^+}{\arg\max}\, \delta_i.$$

We use these same rules both for branching and also for choosing a variable to generate DCCs and DCyCs.

The last component we need is the searching rule. In the BCC algorithm, searching refers to choosing which subproblem to be processed next. Searching has also strong implications on the performance of a BCC implementation. We use the following searching rules, which are based on well-known searching strategies: depth-first with a priority to upper bound constraints, depth-first with a priority to lower bound constraints, and best-first.

5.4 Computational results

In this section, we analyze the effects of DCCs on the BCC tree size and the solution time. For this purpose, we present experiments with various settings to identify the benefits of the DCCs.

For experiments, we implemented the method described in Hirschberger et al. [63] to generate random portfolio data sets. These sets have been shown to very close to realistic instances in the original study. We set the expected value of the covariance of assets to $2 \cdot 10^{-3}$, their standard deviation to $4 \cdot 10^{-6}$, and the expected value of the variance of assets to 0.1 as used in the original paper. An open-source script that generates random portfolio data sets are available online [34]. Table 5.1 gives a list of the problem parameters used in these tests.

Data sets	N_500_1, N_500_2, N_500_3, N_500_4, N_500_5, N_500_6, N_500_7, N_500_8, N_500_9, N_500_10
# Assets	25, 50
Capital	50000, 100000
Return	2%, 3%, 4%, 5%, 6%
Cardinality	2, 3

Table 5.1: Problem parameters

The discussion of our results is organized as follows. First, we discuss the effects

of DCCs on the BCC tree size and the lower bound that was obtained in the nodes. Then we compare the various branching, cutting, and searching rules in order to choose the default settings for our BCC solver. Later we compare the cut application strategies. We close this section with a comparison of a pure B&B, our BCC, and a commercial solver in terms of tree size.

5.4.1 The effect of DCCs on the objective value and the BCC tree

Our first set of experiments demonstrates the effects of DCCs on the nodes of the BCC tree. As discussed earlier, DCCs are used to tighten a given MISOCO formulation. Therefore, it is expected that adding DCCs to the nodes of the BCC tree will improve the lower bound in a given node. However, adding cuts to the formulation is an expensive operation. For that reason, it is crucial to look for a systematic way to select which cuts to add to the subproblems. The goal of these experiments is to provide insights on which indicators may be used to identify which DCCs should be added in the BCC tree.

The first indicator we explored is the depth of the DCCs. In the round-lot AAPs, we use the constraint (5.11) to derive our DCCs, which are obtained by introducing an auxiliary variable and moving the objective function to the constraints. In this case, our results show that the depth of the added DCC is positively correlated to the improvement on the optimal objective value, as is shown in Figure 5.5. In particular, Figure shows an almost linear relation between the improvement on the objective value of a subproblem with the depth of the DCC. It is important to notice that this is true provided that the constraint (5.11) is active at the optimal solution.

The second indicator of the DCC performance we explored is the level at which the cut is generated in the BCC tree. Recall from our methodology description in Section 5.3 that we are using IPMs for solving the node relaxations. As a consequence, the addition of a DCC or DCyC may increase the solution time for a subproblem as

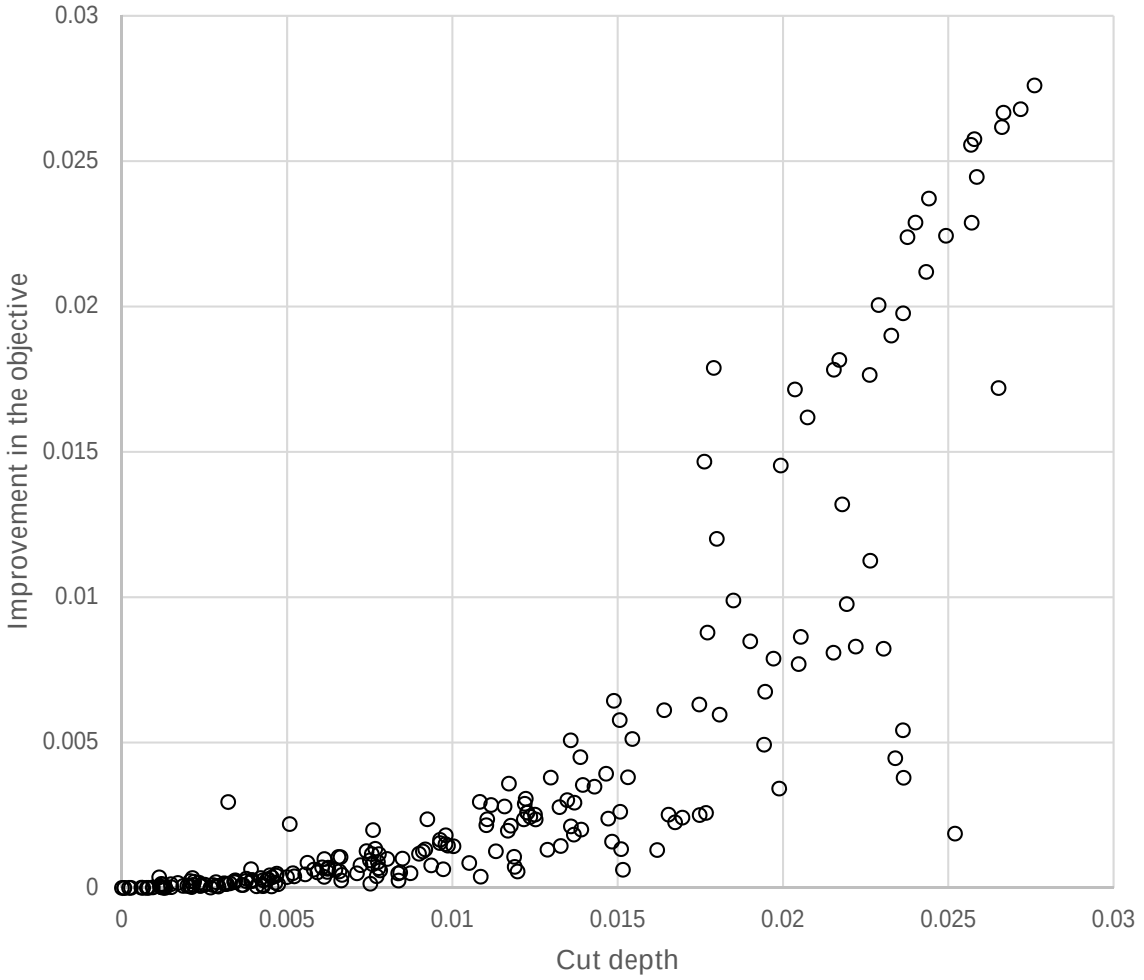


Figure 5.5: Improvement on the objective value versus depth of cuts for round-lot-constrained AAPs.

explained in the following paragraph. This effect can be observed in our experiments. In particular, adding DCCs near the root node increases the solution time for all subproblems, and hence the BCC solution time. Moreover, DCCs added in the lower levels have often better impact on the improvement in the lower bound. Figure 5.6 shows how improvements in the objective value relates to the node level at which a DCC is added. In that figure we observe that DCCs added in the upper levels are to be generally less effective for improving the objective value. This can be explained

by that the solution space is usually larger in upper levels, hence the effect of the cut is usually minimal. For this reason, it is better to add DCCs in the lower levels of the BCC over adding them in the upper levels. On the other hand, adding cuts near the root node often decreases BCC tree sizes significantly. Consequently, our cut generation procedure starts at the root node for the BCC-F method. To lessen the possible increase of the solution time due to new conic constraints, we limited the number of DCCs added to the subproblems.

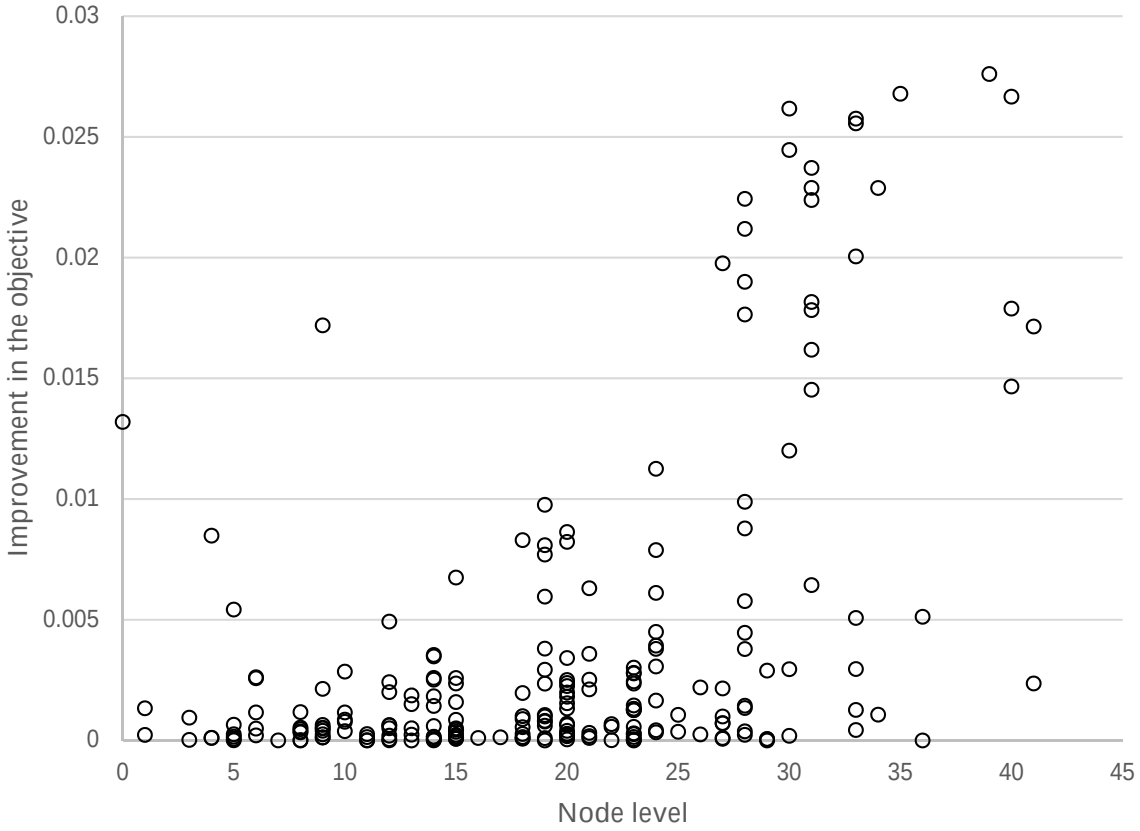


Figure 5.6: Improvement on the objective value versus node level of generated cuts for roundlot constrained AAPs.

The third indicator we need to consider is the dimension of the DCCs. As discussed earlier, some of the assets in the cardinality-constrained AAPs can be left linear for the

quadratic-cardinality constraint. Notice that DCCs for cardinality-constrained AAPs are full in size; hence the number of quadratic terms in the cardinality constraint gives the dimension of the DCC, as well. We varied the number of assets that are left linear in the quadratic-cardinality constraint, which provided us DCyCs with various dimensions. Figure 5.7 shows that the total solution time increases with the number of quadratic terms in the cardinality constraint. Although we have a single type of DCCs for this setting, one must aim to choose lower-dimensional DCCs to minimize its effects on solution time. As mentioned by Pólik and Terlaky [90], iteration complexity of IPMs is independent of the dimension of the cones. However, cost per iteration is significantly affected by the dimension. Although adding a small cone versus adding a big cone has the same complexity in theory, adding a big cone leads to longer arithmetic operations, which eventually increases solution time in practice. Since our approach focuses on practice, we were careful in terms of adding a high number of large DCCs in the BCC tree. On the other hand, adding small DCCs is generally a good practice that does not affect IPM solution time significantly.

We may summarize our conclusions from the results presented in this subsection as follows. First, the depth of a DCC is an important indicator for predicting its effect on the lower bound for problems where the objective is used for generating DCCs. Second, DCCs are more effective when they are added at the lower levels of the BCC tree. However, they often result in smaller tree sizes when they are added closer to the root node. Hence, to maximize the benefit from the DCCs, one should add them closer to the root node of a BCC tree when possible. Third, the dimension of a DCC is an important indicator of its effect on solution time. It is imperative to be more selective and reluctant when adding big DCCs inside the BCC tree.

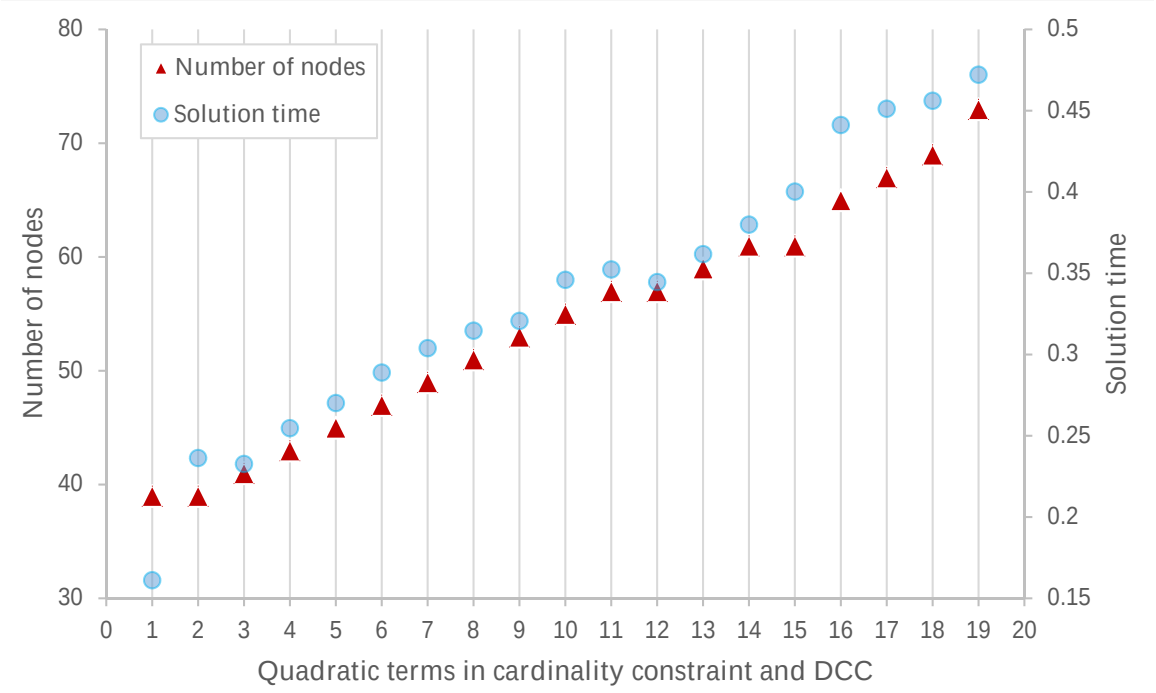


Figure 5.7: Change in solution time and number of nodes for various dimensions of DCCs.

5.4.2 The effect of branching, cutting, and searching rules on the BCC tree

For any given subproblem, one can either branch or generate a cut based on an integer infeasible variable. In the solution of a subproblem, we often get multiple integer infeasible variables. Hence, we need to decide which variable to use for branching or cutting. Here we used the variable selection rules presented in Section 5.3.4 and we are interested on exploring their effects on the solution process.

Before comparing the branching and cutting strategies, we start with identfying the most effective searching rule for the AAPs. Table 5.2 shows the average tree size among four different cut generation strategies. Best-first dominates other searcing rules in terms of average BCC tree size both for B&B and BCCs with various settings.

Since depth-first searching rule with lower bound ($\geq$) direction highlights the impact of DCCs more, we use it as the default searching rule unless otherwise noted.

	B&B	BCC-F	BCC-I	BCC-D
Best-first	7421.00	7385.50	6919.75	6911.25
Breadth-first	8364.00	7508.06	7234.38	7217.25
Depth-first ($\leq$ first)	10453.00	9305.63	10755.06	13058.00
Depth-first ($\geq$ first)	14129.50	7626.00	11797.31	7662.75
Depth-first (closest first)	8304.00	7245.75	8698.06	7232.25

Table 5.2: Average BCC tree size with various searching strategies over cut generation strategies for round-lot AAPs.

We are particularly interested in how branch rules and cut ordering rules affect the BCC when solving AAPs. In Table 5.3, we compare average BCC tree sizes for all combinations of branching and cut ordering for round-lot AAPs. For PR branching, we chose $\alpha = 1/3$ and $\beta = 2/3$ in our implementation. This selection has an empirical basis and was based on our practical observations. From Table 5.3, we observe that ordering branching variables with the MF rule and cuts with the HC rule gives us the minimum tree size in most of the cases for BCC-I strategy.

Based on these results, we use MF as the default branching rule, HC as the default cut ordering rule, and depth-first as the default searching rule. We use these settings for the rest of the experiments, unless otherwise noted.

5.4.3 Comparison of cut application strategies

To identify which cut application strategy performs better, we tested our BCC framework on all data sets for all AAP types. Our aim is to identify the best DCC application strategy for different problem and DCC structures. For this purpose, we compare performance of each strategy to a pure B&B. We ran our experiments by

Cutting	Strategy	Branching			
		MF	HC	PR	IR
MF	BCC-F	3466.63	3373.77	3283.63	1140.45
	BCC-I	1849.31	1092.64	3228.60	4599.07
HC	BCC-F	4984.29	3375.92	3285.63	1140.64
	BCC-I	1859.00	1023.18	1691.20	1156.23
PR	BCC-F	3405.50	6692.33	2993.40	6036.86
	BCC-I	1821.86	1131.17	1139.17	1081.17
IR	BCC-F	3465.50	5079.86	1689.00	3313.50
	BCC-I	1849.31	1125.33	1373.31	1119.83

Table 5.3: Comparison of average BCC tree size for branch and cut ordering rules.

using the default options (MF branching, HC cutting, and depth-first searching) and tested the five cut application strategies presented in Section 5.3.3.

The results for round-lot-constrained AAPs are summarized in Table A.1, where we show the number of nodes in the B&B and BCC trees of various strategies. The results show that in general the BCC methods outperform the B&B methods in terms of the number of nodes. Notice that BCC-I and BCC-F produce similar results for many problems. This is due to fact that BCC-I stops if there is no sufficient improvement in the objective value. For round-lot-constrained AAPs, BCC-I and BCC-F produce the best performance in general. The number of DCCs generated in these methods is higher than in BCC-D. However, these extra DCCs make a great difference in terms of tree size. For a better comparison, performance profiles of number of nodes and solution time are presented in Figure 5.8 and 5.9, respectively. Our expectation about DCCs reducing the BCC tree size when compared with a pure B&B is achieved by all cut-management strategies in many of the cases. On the other hand, an increase in the solution time is often a result of bigger subproblems in the

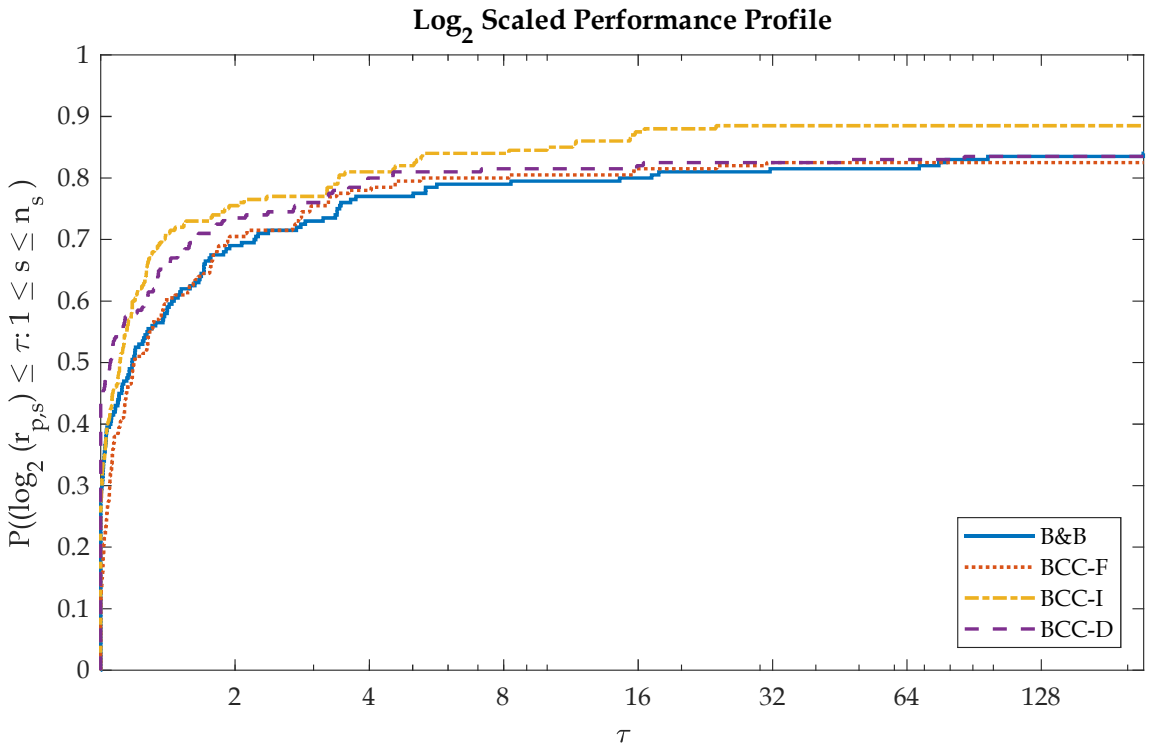


Figure 5.8: Performance profile of number of nodes of cut management strategies on round-lot-constrained AAP.

BCC tree nodes. Note that our BCC solver does not benefit from preprocessing and warm-starting between iterative solutions. Hence we expect the difference between solution times to be smaller when a full BCC framework is implemented.

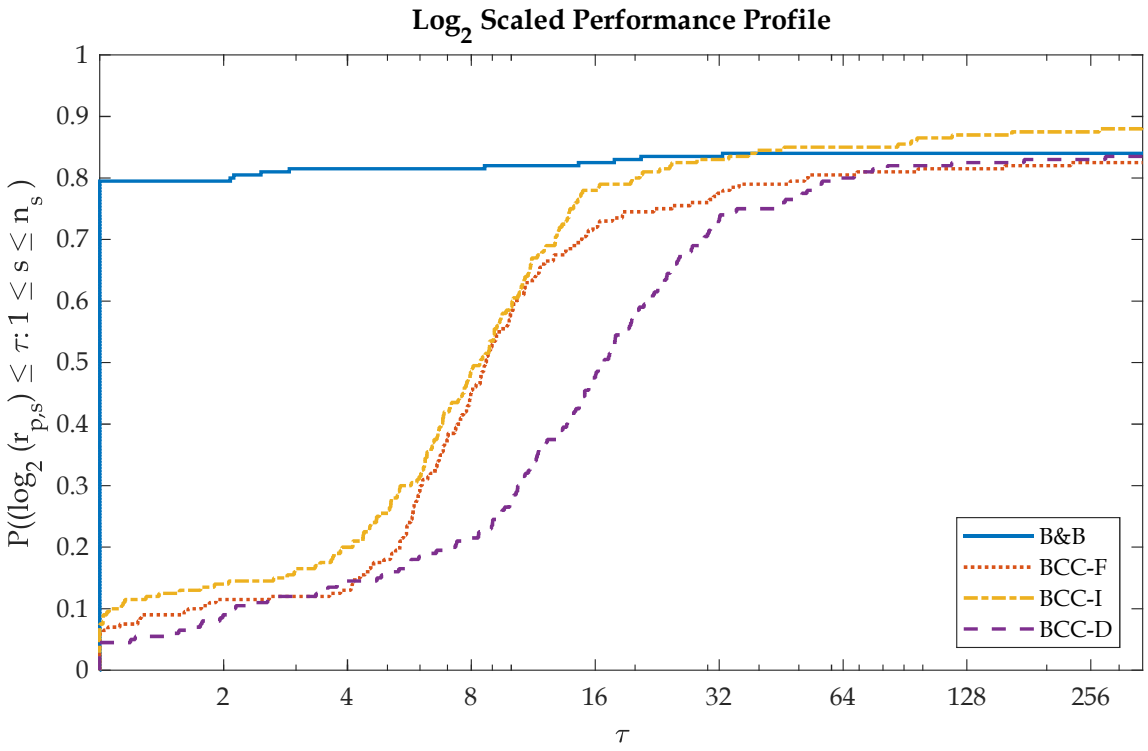


Figure 5.9: Performance profile of solution time of cut management strategies on round-lot-constrained AAP.

We provide the number of nodes in Table B.1 for each data instance with all cut management strategies. Performance profiles for cardinality-constrained AAPs are shown in Figure 5.10 and 5.11 over two different cardinality parameters, $k = 2, 3$ for two different number of assets 10 and 20. The results show that BCC-R, adding all possible DCCs at the root node, dominates all other methods in terms of number of nodes in the BCC tree. On the other hand, BCC methods often result worse solution times. The main reason why solution times increase significantly compared to B&B can be explained with the increasing number of variables when the quadratically constrained optimization problems are converted to SOCO inside the solver. Despite worse solution times of BCC methods compared to B&B, BCC-R works better in

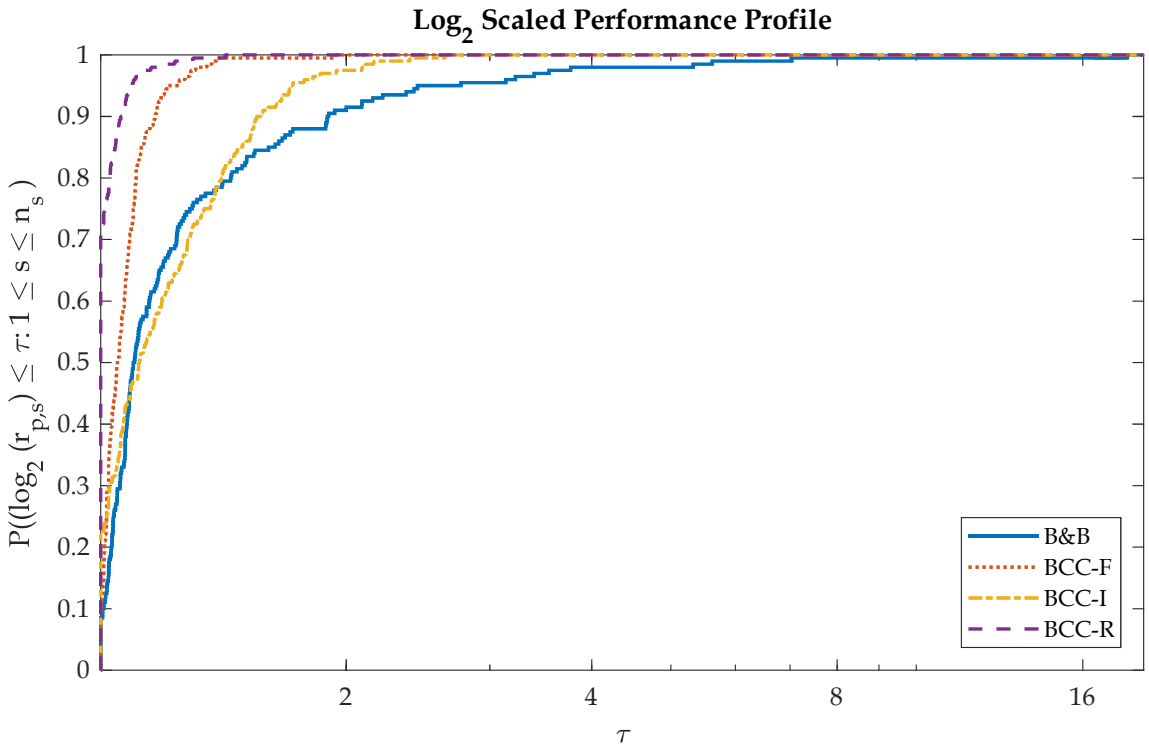


Figure 5.10: Performance profile of number of nodes of cut management strategies on cardinality-constrained AAP.

terms of both the number of nodes and the solution time compared to other BCC cut management methods. Since we have a limited number of variables and all of them are binary, adding these cuts at the root node decreases the tree size significantly. The success of BCC-R matches the preliminary results of Góez [54] on constrained layout problems. Note that DCCs generated for cardinality-constrained AAPs are full in sizes, indicating that the performance of BCC-R is significant. We conclude that it is better to add DCCs generated for binary variables at the root of the BCC tree. The nature of the binary variables enables us even to drop the original constraint after adding the corresponding DCCs.

We may summarize the conclusion of these results as follows. For decreasing

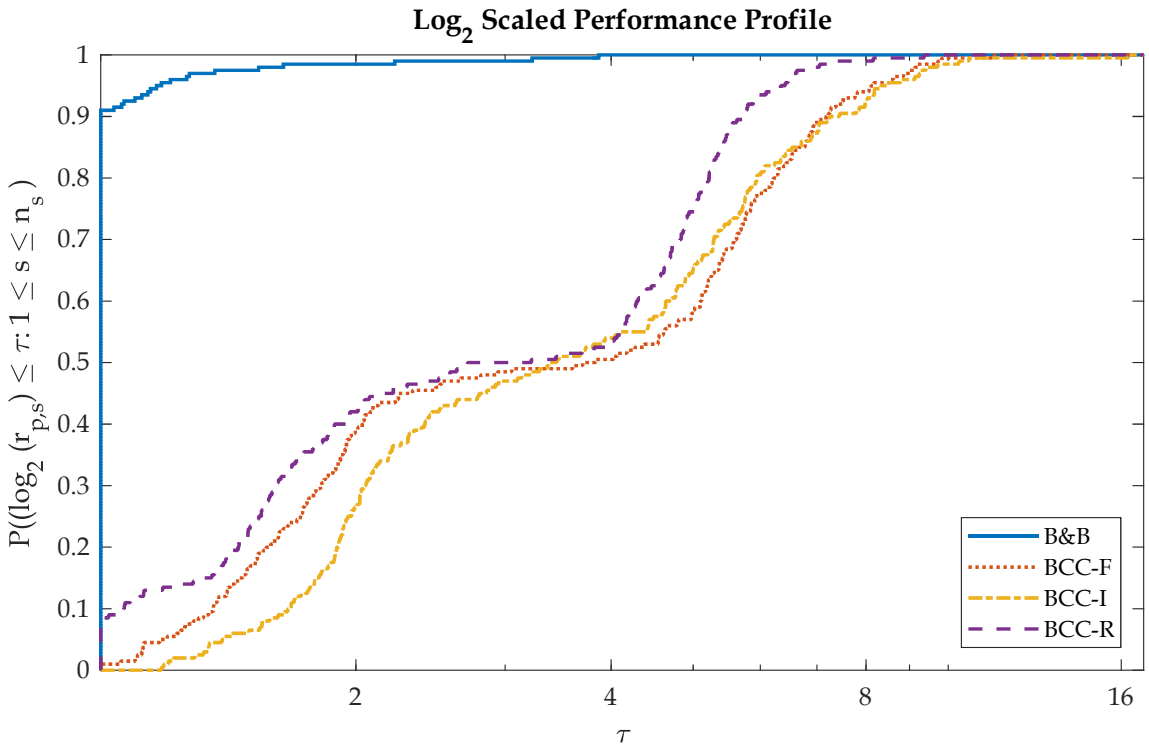


Figure 5.11: Performance profile of solution time of cut management strategies on cardinality-constrained AAP.

the number of nodes, the BCC-I works best for constraints that have general integer variables and BCC-R works best for constraints with binary variables. In the following subsection, we compare the suggested methodology against B&B and a commercial solver.

5.4.4 Comparison of solution approaches

To verify our observations from the previous experiments, we tested our suggested BCC framework against a pure B&B and against a commercial solver, MOSEK. Our aim is to show how much we can save in comparison to other methods. We also implemented a new method, called MOSEK-R, in which DCCs are added as a

preprocessing operation for MOSEK. Since MOSEK does not allow conic cuts to be added in the tree, we add these cuts at the root node. We apply this method only for quadratic-cardinality and quadratic-bound-constrained AAPs, because of the binary variables in the formulation.

For quadratic-cardinality-constrained AAPs, we compare B&B, BCC-R, MOSEK and MOSEK-R, where the latter uses MOSEK after all possible cuts are added at the root node. Figure 5.12 shows the tree size performance profile for cardinality-constrained AAPs. These results show that DCCs reduce BCC tree size significantly for cardinality-constrained AAPs when they are added at the root node for both our implementation and MOSEK. Comparisons both between B&B and BCC-R and between MOSEK and MOSEK-R reinforce this claim. BCC-R dominates other rules in terms of number of nodes and similarly MOSEK-R dominates MOSEK in terms of number of nodes. This example clearly illustrates the power of DCCs in terms of decreasing BCC tree size. In terms of solution time, adding DCCs at the root node results a much bigger problem for both our implementation and MOSEK. Figure 5.13 shows the performance profile of solution times. We observe that despite smaller BCC tree sizes, adding these cuts increase solution times significantly. We can conclude that even though adding DCCs at the root node has a positive impact on the number of nodes, more research is needed for a full implementation of BCC algorithms to match the time performance of the implementations to those we obtained from number of nodes.

5.4.5 Effects of cuts as a preprocessing step

DCCs can be added to MISOCO problems as a preprocessing step. As mentioned earlier, an example of this approach can be seen in Góez [54] on constrained layout problems. These problems include quadratic constraints that have a single binary variable. Replacing those quadratic constraints by the generated DCCs based on the binary variable disjunctions is shown to be effective for these problems. DCCs tighten

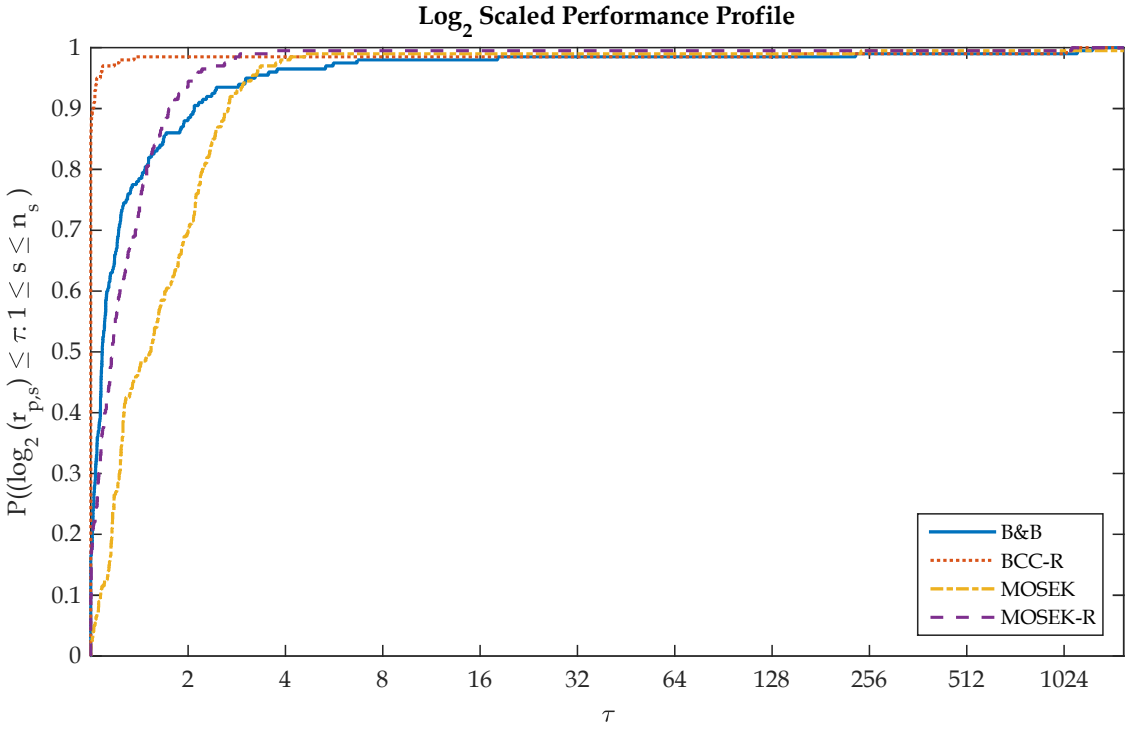


Figure 5.12: Performance profile of number of nodes on quadratic-cardinality-constrained AAPs.

the formulation and reduce solution time significantly.

As an illustration, consider the oversimplified example of a quadratic-bound constraint

$$x_j^2 \leq z_j.$$

Generating a DCC for this constraint brings back the original linear constraint $x_j \leq z_j$ into formulation, which provides better numerical accuracy. To compare accuracy, we solve the instance N_500_2 with B&B and BCC-R for $k = 1$ for 20 assets. We set our integer feasibility tolerance to $\epsilon = 10^{-6}$ and solve the continuous relaxations within BCC with MOSEK in both strategies. Table 5.4 shows that the total investment into assets with $z_j < 10^{-6}$ sums up to 0.43% of the total investment when solved with B&B. This leads to the reported optimal objective value differing by 0.84% of

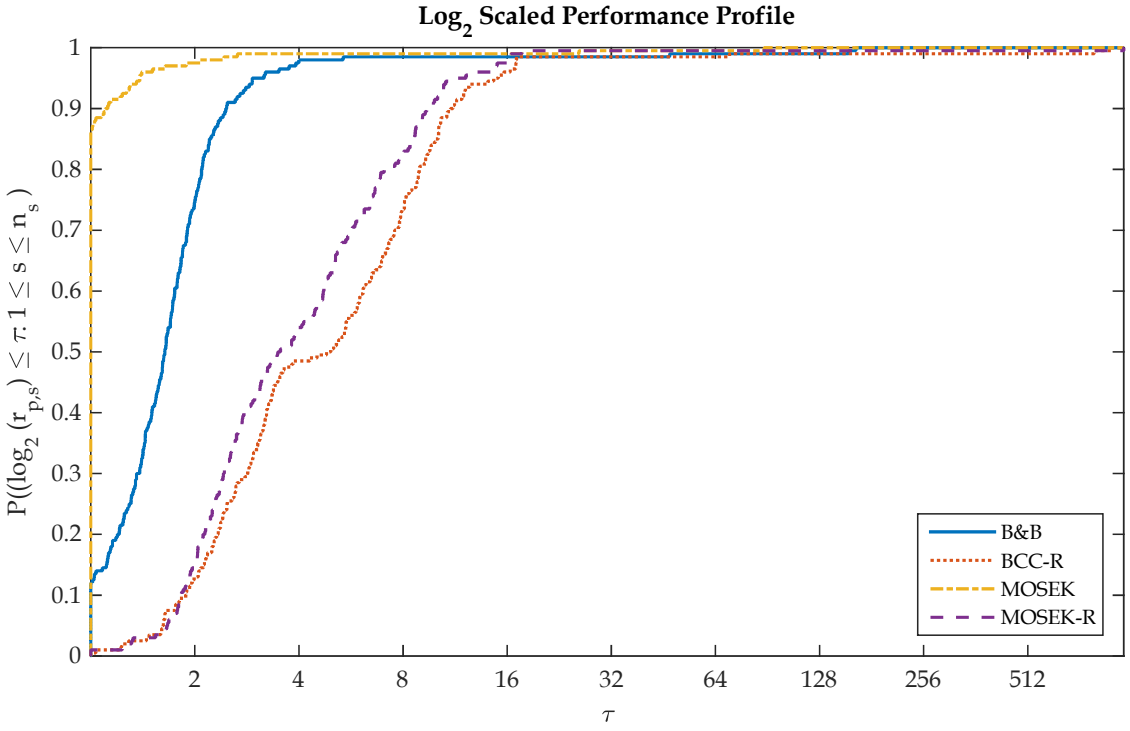


Figure 5.13: Performance profile of solution time on quadratic-cardinality-constrained AAPs.

the true optimal value. Adding DCCs as a preprocessing restores numerical accuracy and allows solver to produce the true optimal solution when solving with BCC-R. In general, we can replace the quadratic constraint that has a single binary variable with the corresponding DCC by using binary disjunction. This preprocessing step at the root node often decreases the number of nodes in BCC tree significantly.

	B&B		BCC-R	
Asset	z	x	z	x
1	1.0000	$9.9566 \cdot 10^{-1}$	1.0000	1.0000
2	$7.3701 \cdot 10^{-8}$	$2.2997 \cdot 10^{-4}$	$9.4848 \cdot 10^{-11}$	$7.2813 \cdot 10^{-11}$
3	$7.1394 \cdot 10^{-8}$	$2.2454 \cdot 10^{-4}$	$9.5056 \cdot 10^{-11}$	$7.3291 \cdot 10^{-11}$
4	$7.2651 \cdot 10^{-8}$	$2.2757 \cdot 10^{-4}$	$9.5115 \cdot 10^{-11}$	$7.2972 \cdot 10^{-11}$
5	$7.3406 \cdot 10^{-8}$	$2.2918 \cdot 10^{-4}$	$9.2110 \cdot 10^{-11}$	$7.2070 \cdot 10^{-11}$
6	$7.2862 \cdot 10^{-8}$	$2.2798 \cdot 10^{-4}$	$9.0413 \cdot 10^{-11}$	$7.1294 \cdot 10^{-11}$
7	$7.7673 \cdot 10^{-8}$	$2.3813 \cdot 10^{-4}$	$9.0883 \cdot 10^{-11}$	$7.1274 \cdot 10^{-11}$
8	$7.3302 \cdot 10^{-8}$	$2.2897 \cdot 10^{-4}$	$9.4299 \cdot 10^{-11}$	$7.3213 \cdot 10^{-11}$
9	$7.4398 \cdot 10^{-8}$	$2.3137 \cdot 10^{-4}$	$9.3464 \cdot 10^{-11}$	$7.2511 \cdot 10^{-11}$
10	$7.3573 \cdot 10^{-8}$	$2.2960 \cdot 10^{-4}$	$9.4311 \cdot 10^{-11}$	$7.3104 \cdot 10^{-11}$
11	$7.3541 \cdot 10^{-8}$	$2.2961 \cdot 10^{-4}$	$9.4877 \cdot 10^{-11}$	$7.2862 \cdot 10^{-11}$
12	$7.3655 \cdot 10^{-8}$	$2.2987 \cdot 10^{-4}$	$9.4890 \cdot 10^{-11}$	$7.2793 \cdot 10^{-11}$
13	$7.3313 \cdot 10^{-8}$	$2.2908 \cdot 10^{-4}$	$9.4824 \cdot 10^{-11}$	$7.2996 \cdot 10^{-11}$
14	$7.3843 \cdot 10^{-8}$	$2.3011 \cdot 10^{-4}$	$8.8534 \cdot 10^{-11}$	$7.0139 \cdot 10^{-11}$
15	$7.3600 \cdot 10^{-8}$	$2.2963 \cdot 10^{-4}$	$9.3668 \cdot 10^{-11}$	$7.2993 \cdot 10^{-11}$
16	$7.3150 \cdot 10^{-8}$	$2.2873 \cdot 10^{-4}$	$9.4997 \cdot 10^{-11}$	$7.2913 \cdot 10^{-11}$
17	$7.2638 \cdot 10^{-8}$	$2.2748 \cdot 10^{-4}$	$9.1394 \cdot 10^{-11}$	$7.1981 \cdot 10^{-11}$
18	$6.5921 \cdot 10^{-8}$	$2.1212 \cdot 10^{-4}$	$9.5289 \cdot 10^{-11}$	$7.3302 \cdot 10^{-11}$
19	$7.4275 \cdot 10^{-8}$	$2.3120 \cdot 10^{-4}$	$9.4550 \cdot 10^{-11}$	$7.2737 \cdot 10^{-11}$
20	$7.2710 \cdot 10^{-8}$	$2.2769 \cdot 10^{-4}$	$9.4969 \cdot 10^{-11}$	$7.3088 \cdot 10^{-11}$
Objective	$2.8764 \cdot 10^{1}$		$2.9007 \cdot 10^{1}$	

Table 5.4: Comparison of numerical accuracy of solution without DCCs (B&B) versus when DCCs are added in the preprocessing (BCC-R).

5.5 Conclusions and future work

In this chapter, we show the effect of DCCs and DCyCs on solving AAPs. We focused on AAP problems where integer variables appear; more specifically we consider round-lot-constrained and cardinality-constrained AAPs. We present all steps of cut generation for these problems, as well as the details on how we implement DCCs in a complete BCC framework.

Our purpose for studying these problems was to illustrate the positive effect of conic cuts for an important real-world problem. A key contribution of this chapter is that it moves the recently developed theory of DCCs from theory to computational practice. We illustrate how the recent theoretical development of DCCs can be used in general-purpose MISOCO solvers. We developed a BCC framework around the recently developed DCCs and DCyCs. The proposed software was able to solve instances of the AAP in a reasonable amount of time and enables us to compare various strategies in terms of managing conic and cylindrical cuts within a BCC framework.

We solved a real-world data set and several randomly generated data sets. By experimenting with different sized conic and cylindrical cuts, we were able to observe their effects. We show that adding DCCs and DCyCs help reduce the BCC tree size significantly for the majority of the experiments, although solution time may increase in many instances because of the increasing size of the problems. We tested several strategies for choosing and adding DCCs and DCyCs within a BCC tree. Our experiments show that BCC-I was the best method for generating DCCs for MISOCO formulations with general integer variables. On the other hand, BCC-R provides significant reduction in tree size for MISOCO formulations with binary variables. We saw significant improvements in the BCC tree size by applying DCCs when they are ordered with the HC rule. It should be emphasized that solution time can be greatly reduced within a full BCC framework with warm-starting, preprocessing, and cut management.

Many important questions remain for future research, such as cut pooling, cut removal, and cut recovery. Note that we have not used any DCCs or DCyCs based on general disjunctions. The development of warm-start methods after conic cuts are added and preprocessing techniques also remain for future research. As these crucial elements are developed, it is possible to develop a comprehensive, efficient, general-purpose BCC methodology to solve MISOCO problems by integrating DCCs and DCyCs.

Chapter 6

Conclusions and future work

This thesis investigated the computational approaches for MISOCO. We presented:

- A novel warm-starting method specifically for SOCO
- Novel heuristics to obtain and improve MISOCO solutions
- Conic cut-management strategies on AAPs

Both theoretical and practical aspects of these topics are discussed in this thesis.

After presenting the basics and state of the art, we introduced a novel warm-starting method for IPMs for SOCO problems in Chapter 3. Recent development on generating valid cuts for MISOCO problems has demonstrated the need for an efficient warm-starting method. We presented detailed numerical results for the warm-starting method on the CBLIB test set. Our warm-starting method decreased the number of IPM iterations needed by 20 to 50% based on variable type, showing that this method is more efficient than available methods in the literature. Fine-tuning the warm-start parameters can further increase the benefit gained from the method. This warm-starting method for IPMs will increase competitiveness of IPMs for MISOCO problems inside BCC frameworks against outer-approximation methods. Although this thesis describes only warm-starting

after branching, it is straightforward to extend the warm-starting method after linear and conic cuts are introduced to MISOCO problems.

Chapter 4 presented novel rounding heuristics for MISOCO problems. These heuristics exploit Jordan algebra and provide a means of obtaining integer feasible solutions in an inexpensive way; they are developed specifically for SOCO and are the first of their kinds in the literature. Since heuristics are vital parts of any B&B algorithm, our work here provided an important component of a full BCC framework. Numerical results showed that not only are our heuristics successful at finding feasible solutions for MISOCO problems, but they also provide a small gap to an optimal solution. Within a reasonable time limit, at least one of the heuristics provided a feasible solution for 1327 out of 1328 test problems that we considered. These are significant results for a heuristic method, even for LO. We also showed that the method can be easily translated to MIQO, and we provided the derivations. Our work here proved that the developments on conic optimization are helpful for future research on QO as well. Benefits obtained from these heuristics can be further increased by allocating the budget between primal and dual heuristics based on the problem structure.

Chapter 5 presented our investigation of applying DCCs and DCyCs for AAPs. We managed to show that these powerful cuts can be helpful when solving real-world problems. As expected, we conclude that the benefit obtained from these cuts depends heavily on the problem and variable structures. We presented results that show DCCs and DCyCs significantly reducing the number of nodes in BCC search trees for AAPs. This work highlights the importance of the work on generating valid inequalities for MISOCO problems. Although a specific problem type is studied here to focus on cut-management strategies at a micro level, more work is needed to manage such cuts for general-purpose solvers.

Due to the encouraging results presented in this thesis, we believe that developing a full BCC framework that includes the methods and strategies presented in this thesis

is the next step for future research. Another future research direction is extending the warm-starting and heuristics methods to SDO. However, more work is needed to extend these methods to SDO efficiently, since obtaining eigenvectors for large and dense matrices is expensive for SDO. Regarding the DCCs and DCyCs, many important elements for developing a full BCC framework remain for future research; such as cut pooling, cut removal, and cut recovery.

Appendix A

Round-lot-constrained AAP experiments

Asset	Dataset	Capital	Return	B&B	BCC-F	BCC-I	BCC-D
25	N_500_1	50000	0.02	143	143	143	143
25	N_500_1	50000	0.03	29	37	37	35
25	N_500_1	50000	0.04	2029	2029	2029	2017
25	N_500_1	50000	0.05	441	499	443	495
25	N_500_1	50000	0.06	203	209	207	209
25	N_500_1	100000	0.02	2029	2033	2029	2007
25	N_500_1	100000	0.03	203	205	205	213
25	N_500_1	100000	0.04	1215	1281	1277	1297
25	N_500_1	100000	0.05	77	81	79	69
25	N_500_1	100000	0.06	531	561	529	557
25	N_500_2	50000	0.02	4539	5757	5739	5957
25	N_500_2	50000	0.03	4289	3395	3577	1927
25	N_500_2	50000	0.04	Limit	Limit	Limit	22421
25	N_500_2	50000	0.05	3483	2331	1791	2215
25	N_500_2	50000	0.06	Limit	Limit	Limit	Limit
25	N_500_2	100000	0.02	11985	Limit	24421	22427
25	N_500_2	100000	0.03	119755	Limit	Limit	Limit
25	N_500_2	100000	0.04	71651	Limit	41781	Limit
25	N_500_2	100000	0.05	Limit	11725	11403	11043
25	N_500_2	100000	0.06	8707	2821	2799	2539

Asset	Dataset	Capital	Return	B&B	BCC-F	BCC-I	BCC-D
25	N_500_3	50000	0.02	2439	3031	3043	2733
25	N_500_3	50000	0.03	1795	1765	1755	1777
25	N_500_3	50000	0.04	625	2855	2855	2823
25	N_500_3	50000	0.05	491	145	149	1033
25	N_500_3	50000	0.06	155	151	29	29
25	N_500_3	100000	0.02	625	2855	2821	2825
25	N_500_3	100000	0.03	155	33	31	29
25	N_500_3	100000	0.04	2409	2445	2413	2139
25	N_500_3	100000	0.05	4941	3977	985	3907
25	N_500_3	100000	0.06	955	2017	2003	1301
25	N_500_4	50000	0.02	1193	1045	1059	999
25	N_500_4	50000	0.03	733	929	729	931
25	N_500_4	50000	0.04	813	711	651	739
25	N_500_4	50000	0.05	887	17	15	13
25	N_500_4	50000	0.06	943	59	55	873
25	N_500_4	100000	0.02	813	655	657	739
25	N_500_4	100000	0.03	943	103	53	871
25	N_500_4	100000	0.04	2329	2489	2527	2479
25	N_500_4	100000	0.05	1939	1645	983	937
25	N_500_4	100000	0.06	293	957	941	971
25	N_500_5	50000	0.02	355	377	377	381
25	N_500_5	50000	0.03	297	309	311	297
25	N_500_5	50000	0.04	985	873	771	819
25	N_500_5	50000	0.05	Limit	1035	1029	1005
25	N_500_5	50000	0.06	Limit	1457	1457	1491
25	N_500_5	100000	0.02	985	833	783	819
25	N_500_5	100000	0.03	1567	1453	1601	1491
25	N_500_5	100000	0.04	6093	5419	5469	5291
25	N_500_5	100000	0.05	6617	4559	5897	4401
25	N_500_5	100000	0.06	3461	6107	4947	4929
25	N_500_6	50000	0.02	2463	1143	1141	1093
25	N_500_6	50000	0.03	6705	69	69	5939
25	N_500_6	50000	0.04	Limit	35099	33471	Limit
25	N_500_6	50000	0.05	Limit	2637	3613	4261
25	N_500_6	50000	0.06	19	19	63	19
25	N_500_6	100000	0.02	35191	35139	35173	Limit
25	N_500_6	100000	0.03	19	19	19	19
25	N_500_6	100000	0.04	8823	2717	2699	2631

Asset	Dataset	Capital	Return	B&B	BCC-F	BCC-I	BCC-D
25	N_500_6	100000	0.05	Limit	17063	22531	22651
25	N_500_6	100000	0.06	271	441	343	403
25	N_500_7	50000	0.02	465	457	457	431
25	N_500_7	50000	0.03	621	717	719	601
25	N_500_7	50000	0.04	77	217	77	77
25	N_500_7	50000	0.05	163	447	161	441
25	N_500_7	50000	0.06	373	479	485	669
25	N_500_7	100000	0.02	77	217	77	77
25	N_500_7	100000	0.03	373	477	281	665
25	N_500_7	100000	0.04	317	317	313	311
25	N_500_7	100000	0.05	185	295	185	183
25	N_500_7	100000	0.06	511	491	427	575
25	N_500_8	50000	0.02	203	177	175	171
25	N_500_8	50000	0.03	1645	1725	939	1705
25	N_500_8	50000	0.04	2007	2593	2589	1457
25	N_500_8	50000	0.05	1915	1573	1701	1327
25	N_500_8	50000	0.06	1531	945	907	899
25	N_500_8	100000	0.02	2007	2591	1845	1451
25	N_500_8	100000	0.03	1531	913	919	899
25	N_500_8	100000	0.04	1043	859	857	1229
25	N_500_8	100000	0.05	67	205	57	205
25	N_500_8	100000	0.06	4667	683	3725	321
25	N_500_9	50000	0.02	21	21	21	21
25	N_500_9	50000	0.03	2289	2391	2291	2229
25	N_500_9	50000	0.04	93	111	111	95
25	N_500_9	50000	0.05	Limit	1591	3081	1597
25	N_500_9	50000	0.06	1143	1805	1147	1807
25	N_500_9	100000	0.02	93	93	93	99
25	N_500_9	100000	0.03	1143	1805	1145	1801
25	N_500_9	100000	0.04	377	349	377	345
25	N_500_9	100000	0.05	1735	1749	1737	1939
25	N_500_9	100000	0.06	1093	1093	1073	1135
25	N_500_10	50000	0.02	645	691	235	445
25	N_500_10	50000	0.03	399	505	501	473
25	N_500_10	50000	0.04	341	347	331	337
25	N_500_10	50000	0.05	421	277	267	259
25	N_500_10	50000	0.06	133	133	133	113
25	N_500_10	100000	0.02	341	343	341	335

Asset	Dataset	Capital	Return	B&B	BCC-F	BCC-I	BCC-D
25	N_500_10	100000	0.03	133	133	133	113
25	N_500_10	100000	0.04	71	69	69	273
25	N_500_10	100000	0.05	733	415	579	687
25	N_500_10	100000	0.06	1483	1191	1185	935
50	N_500_1	50000	0.02	51709	Limit	Limit	13861
50	N_500_1	50000	0.03	16797	17051	16949	Limit
50	N_500_1	50000	0.04	2637	3603	3571	3501
50	N_500_1	50000	0.05	707	753	557	743
50	N_500_1	50000	0.06	Limit	Limit	547	Limit
50	N_500_1	100000	0.02	2637	3605	2599	3501
50	N_500_1	100000	0.03	43987	Limit	203	9548
50	N_500_1	100000	0.04	85689	Limit	Limit	10322
50	N_500_1	100000	0.05	2885	3971	4233	3637
50	N_500_1	100000	0.06	1323	10867	15117	Limit
50	N_500_2	50000	0.02	70923	Limit	3072	904
50	N_500_2	50000	0.03	4313	3429	4358	1274
50	N_500_2	50000	0.04	1753	Limit	4742	930
50	N_500_2	50000	0.05	Limit	Limit	3222	914
50	N_500_2	50000	0.06	Limit	Limit	17216	1121
50	N_500_2	100000	0.02	1753	Limit	4789	918
50	N_500_2	100000	0.03	Limit	Limit	18643	1129
50	N_500_2	100000	0.04	29615	29103	4933	937
50	N_500_2	100000	0.05	70371	Limit	3187	933
50	N_500_2	100000	0.06	7615	4545	3165	1345
50	N_500_3	50000	0.02	7735	12113	11989	11821
50	N_500_3	50000	0.03	2263	2339	2323	2117
50	N_500_3	50000	0.04	1113	1827	671	1813
50	N_500_3	50000	0.05	177	163	199	161
50	N_500_3	50000	0.06	14953	14809	14581	Limit
50	N_500_3	100000	0.02	1113	1829	667	1813
50	N_500_3	100000	0.03	14953	14827	14013	10750
50	N_500_3	100000	0.04	14677	16069	17219	Limit
50	N_500_3	100000	0.05	421	383	383	809
50	N_500_3	100000	0.06	5315	4125	4107	3637
50	N_500_4	50000	0.02	1219	1225	10011	1745
50	N_500_4	50000	0.03	Limit	2573	2521	2505
50	N_500_4	50000	0.04	17001	Limit	16133	Limit
50	N_500_4	50000	0.05	9151	15117	14049	Limit

Asset	Dataset	Capital	Return	B&B	BCC-F	BCC-I	BCC-D
50	N_500_4	50000	0.06	Limit	19577	19557	Limit
50	N_500_4	100000	0.02	17001	14857	16239	Limit
50	N_500_4	100000	0.03	10313	19653	19547	Limit
50	N_500_4	100000	0.04	31089	Limit	Limit	Limit
50	N_500_4	100000	0.05	20805	18625	18617	Limit
50	N_500_4	100000	0.06	21863	16899	16893	14429
50	N_500_5	50000	0.02	195	175	175	179
50	N_500_5	50000	0.03	1379	1495	1445	1263
50	N_500_5	50000	0.04	3119	3041	3029	2691
50	N_500_5	50000	0.05	Limit	6443	6329	5725
50	N_500_5	50000	0.06	Limit	12453	12345	11891
50	N_500_5	100000	0.02	3119	3041	2983	2689
50	N_500_5	100000	0.03	37447	12431	Limit	11891
50	N_500_5	100000	0.04	Limit	Limit	Limit	13836
50	N_500_5	100000	0.05	Limit	Limit	Limit	Limit
50	N_500_5	100000	0.06	Limit	Limit	Limit	9753
50	N_500_6	50000	0.02	705	541	633	541
50	N_500_6	50000	0.03	1173	1719	1299	1573
50	N_500_6	50000	0.04	Limit	901	689	879
50	N_500_6	50000	0.05	2981	1351	1631	1415
50	N_500_6	50000	0.06	3175	2473	2483	2253
50	N_500_6	100000	0.02	1985	909	691	879
50	N_500_6	100000	0.03	3175	2491	2463	2251
50	N_500_6	100000	0.04	3319	3879	3871	3289
50	N_500_6	100000	0.05	53321	Limit	22447	Limit
50	N_500_6	100000	0.06	46389	13173	13115	12779
50	N_500_7	50000	0.02	5871	10409	10279	9563
50	N_500_7	50000	0.03	7897	7611	7585	6617
50	N_500_7	50000	0.04	6015	6021	6013	5691
50	N_500_7	50000	0.05	4641	5297	4919	4943
50	N_500_7	50000	0.06	Limit	4189	4175	2987
50	N_500_7	100000	0.02	6015	6017	6893	5693
50	N_500_7	100000	0.03	4447	4197	3373	2985
50	N_500_7	100000	0.04	2635	1469	763	1385
50	N_500_7	100000	0.05	753	535	529	599
50	N_500_7	100000	0.06	1141	1127	1095	1091
50	N_500_8	50000	0.02	20193	Limit	Limit	12018
50	N_500_8	50000	0.03	Limit	Limit	Limit	Limit

Asset	Dataset	Capital	Return	B&B	BCC-F	BCC-I	BCC-D
50	N_500_8	50000	0.04	Limit	Limit	Limit	Limit
50	N_500_8	50000	0.05	6227	6207	6543	6139
50	N_500_8	50000	0.06	Limit	16893	9273	Limit
50	N_500_8	100000	0.02	51047	Limit	Limit	Limit
50	N_500_8	100000	0.03	15827	16903	9293	14337
50	N_500_8	100000	0.04	2657	7871	13369	10451
50	N_500_8	100000	0.05	Limit	Limit	Limit	11551
50	N_500_8	100000	0.06	Limit	15805	17689	Limit
50	N_500_9	50000	0.02	15127	14959	14701	Limit
50	N_500_9	50000	0.03	Limit	Limit	Limit	Limit
50	N_500_9	50000	0.04	Limit	Limit	Limit	10806
50	N_500_9	50000	0.05	Limit	Limit	Limit	8995
50	N_500_9	50000	0.06	15997	16007	15971	Limit
50	N_500_9	100000	0.02	30041	Limit	Limit	10703
50	N_500_9	100000	0.03	15997	16031	17929	Limit
50	N_500_9	100000	0.04	56017	Limit	Limit	Limit
50	N_500_9	100000	0.05	35437	Limit	Limit	Limit
50	N_500_9	100000	0.06	Limit	Limit	Limit	Limit
50	N_500_10	50000	0.02	89	2087	2079	87
50	N_500_10	50000	0.03	1583	1605	1593	1715
50	N_500_10	50000	0.04	1059	3427	3407	3375
50	N_500_10	50000	0.05	Limit	12929	12767	10114
50	N_500_10	50000	0.06	1011	15815	15929	Limit
50	N_500_10	100000	0.02	1059	3429	3403	3381
50	N_500_10	100000	0.03	1011	15829	15519	Limit
50	N_500_10	100000	0.04	2721	Limit	Limit	8809
50	N_500_10	100000	0.05	2237	2335	2577	2793
50	N_500_10	100000	0.06	2017	2095	19987	1997

Table A.1: Performance of cut application strategies on round-lot-constrained AAPs.

Appendix B

Cardinality-constrained AAP experiments

Asset	Dataset	Cardinality	Return	B&B	BCC-F	BCC-I	BCC-R
10	N_500_1	2	0.02	91	93	85	87
10	N_500_1	2	0.03	89	87	79	83
10	N_500_1	2	0.04	69	53	69	49
10	N_500_1	2	0.05	53	37	45	33
10	N_500_1	2	0.06	27	23	29	21
10	N_500_1	3	0.02	272	235	243	233
10	N_500_1	3	0.03	277	235	211	227
10	N_500_1	3	0.04	234	151	173	143
10	N_500_1	3	0.05	201	89	133	83
10	N_500_1	3	0.06	147	51	71	39
10	N_500_2	2	0.02	99	91	91	89
10	N_500_2	2	0.03	91	91	83	89
10	N_500_2	2	0.04	87	89	81	87
10	N_500_2	2	0.05	83	71	75	67
10	N_500_2	2	0.06	69	43	53	41
10	N_500_2	3	0.02	272	239	247	239
10	N_500_2	3	0.03	292	239	235	239
10	N_500_2	3	0.04	280	221	233	227
10	N_500_2	3	0.05	239	169	199	165
10	N_500_2	3	0.06	217	127	147	115

Asset	Dataset	Cardinality	Return	B&B	BCC-F	BCC-I	BCC-R
10	N_500_3	2	0.02	89	91	85	89
10	N_500_3	2	0.03	83	67	83	73
10	N_500_3	2	0.04	59	49	73	47
10	N_500_3	2	0.05	57	39	63	37
10	N_500_3	2	0.06	51	17	23	15
10	N_500_3	3	0.02	249	241	229	239
10	N_500_3	3	0.03	237	183	203	193
10	N_500_3	3	0.04	233	135	157	123
10	N_500_3	3	0.05	216	77	111	67
10	N_500_3	3	0.06	190	31	57	27
10	N_500_4	2	0.02	91	87	91	85
10	N_500_4	2	0.03	81	71	75	69
10	N_500_4	2	0.04	73	65	69	59
10	N_500_4	2	0.05	65	51	67	49
10	N_500_4	2	0.06	51	29	45	27
10	N_500_4	3	0.02	260	239	235	237
10	N_500_4	3	0.03	251	209	223	207
10	N_500_4	3	0.04	230	167	191	166
10	N_500_4	3	0.05	206	103	157	103
10	N_500_4	3	0.06	195	61	115	55
10	N_500_5	2	0.02	95	91	89	89
10	N_500_5	2	0.03	89	91	87	89
10	N_500_5	2	0.04	93	85	79	89
10	N_500_5	2	0.05	87	69	79	67
10	N_500_5	2	0.06	81	51	57	49
10	N_500_5	3	0.02	258	241	253	239
10	N_500_5	3	0.03	283	241	237	239
10	N_500_5	3	0.04	257	237	237	239
10	N_500_5	3	0.05	269	179	219	179
10	N_500_5	3	0.06	242	125	167	125
10	N_500_6	2	0.02	89	93	95	89
10	N_500_6	2	0.03	89	91	91	89
10	N_500_6	2	0.04	89	87	91	89
10	N_500_6	2	0.05	93	79	81	81
10	N_500_6	2	0.06	79	55	63	53
10	N_500_6	3	0.02	260	241	249	239
10	N_500_6	3	0.03	256	241	251	239
10	N_500_6	3	0.04	275	239	245	239

Asset	Dataset	Cardinality	Return	B&B	BCC-F	BCC-I	BCC-R
10	N_500_6	3	0.05	267	215	225	223
10	N_500_6	3	0.06	251	155	187	147
10	N_500_7	2	0.02	89	91	91	89
10	N_500_7	2	0.03	89	91	89	89
10	N_500_7	2	0.04	82	73	79	61
10	N_500_7	2	0.05	97	53	77	45
10	N_500_7	2	0.06	43	27	41	25
10	N_500_7	3	0.02	272	239	243	239
10	N_500_7	3	0.03	271	237	233	237
10	N_500_7	3	0.04	257	176	203	175
10	N_500_7	3	0.05	232	119	171	111
10	N_500_7	3	0.06	229	51	103	43
10	N_500_8	2	0.02	89	91	87	89
10	N_500_8	2	0.03	87	91	83	89
10	N_500_8	2	0.04	88	63	63	61
10	N_500_8	2	0.05	71	51	65	47
10	N_500_8	2	0.06	59	33	49	31
10	N_500_8	3	0.02	274	241	241	239
10	N_500_8	3	0.03	265	241	215	239
10	N_500_8	3	0.04	234	165	195	155
10	N_500_8	3	0.05	218	115	157	115
10	N_500_8	3	0.06	204	89	97	65
10	N_500_9	2	0.02	89	91	97	89
10	N_500_9	2	0.03	89	91	91	89
10	N_500_9	2	0.04	77	63	77	61
10	N_500_9	2	0.05	65	47	65	45
10	N_500_9	2	0.06	51	35	51	33
10	N_500_9	3	0.02	285	241	245	239
10	N_500_9	3	0.03	266	241	227	239
10	N_500_9	3	0.04	249	183	205	181
10	N_500_9	3	0.05	242	147	185	145
10	N_500_9	3	0.06	207	107	141	99
10	N_500_10	2	0.02	29	31	41	29
10	N_500_10	2	0.03	19	21	27	19
10	N_500_10	2	0.04	19	21	29	19
10	N_500_10	2	0.05	45	21	25	19
10	N_500_10	2	0.06	47	33	45	17
10	N_500_10	3	0.02	203	91	127	83

Asset	Dataset	Cardinality	Return	B&B	BCC-F	BCC-I	BCC-R
10	N_500_10	3	0.03	103	91	97	73
10	N_500_10	3	0.04	153	89	97	69
10	N_500_10	3	0.05	208	41	80	37
10	N_500_10	3	0.06	127	9	13	7
20	N_500_1	2	0.02	371	365	345	363
20	N_500_1	2	0.03	231	247	289	233
20	N_500_1	2	0.04	173	177	177	163
20	N_500_1	2	0.05	115	119	143	111
20	N_500_1	2	0.06	77	77	99	71
20	N_500_1	3	0.02	2211	2189	1981	2167
20	N_500_1	3	0.03	1613	1465	1551	1495
20	N_500_1	3	0.04	1035	1025	1057	999
20	N_500_1	3	0.05	667	607	683	597
20	N_500_1	3	0.06	303	287	359	279
20	N_500_2	2	0.02	381	389	401	379
20	N_500_2	2	0.03	373	383	377	373
20	N_500_2	2	0.04	321	353	359	321
20	N_500_2	2	0.05	261	275	303	255
20	N_500_2	2	0.06	191	195	243	185
20	N_500_2	3	0.02	2289	2285	2261	2279
20	N_500_2	3	0.03	2257	2251	2173	2233
20	N_500_2	3	0.04	2107	1955	1955	1937
20	N_500_2	3	0.05	1827	1551	1669	1541
20	N_500_2	3	0.06	1125	1057	1209	1039
20	N_500_3	2	0.02	367	367	341	363
20	N_500_3	2	0.03	243	281	289	241
20	N_500_3	2	0.04	171	185	207	171
20	N_500_3	2	0.05	133	149	187	127
20	N_500_3	2	0.06	65	85	99	63
20	N_500_3	3	0.02	2175	2143	2019	2151
20	N_500_3	3	0.03	1647	1497	1593	1531
20	N_500_3	3	0.04	1143	1077	1143	1067
20	N_500_3	3	0.05	615	597	675	587
20	N_500_3	3	0.06	237	227	327	217
20	N_500_4	2	0.02	379	391	377	379
20	N_500_4	2	0.03	353	375	343	363
20	N_500_4	2	0.04	279	283	299	275
20	N_500_4	2	0.05	219	249	247	219

Asset	Dataset	Cardinality	Return	B&B	BCC-F	BCC-I	BCC-R
20	N_500_4	2	0.06	145	181	171	145
20	N_500_4	3	0.02	2267	2277	2189	2275
20	N_500_4	3	0.03	2175	2149	2057	2141
20	N_500_4	3	0.04	1921	1739	1757	1727
20	N_500_4	3	0.05	1415	1305	1327	1291
20	N_500_4	3	0.06	785	789	819	771
20	N_500_5	2	0.02	381	377	379	371
20	N_500_5	2	0.03	369	357	349	353
20	N_500_5	2	0.04	321	317	291	315
20	N_500_5	2	0.05	215	221	217	195
20	N_500_5	2	0.06	149	163	171	139
20	N_500_5	3	0.02	2281	2265	2161	2257
20	N_500_5	3	0.03	2211	2127	1929	2129
20	N_500_5	3	0.04	1943	1567	1671	1853
20	N_500_5	3	0.05	1317	1201	1263	1183
20	N_500_5	3	0.06	783	793	785	733
20	N_500_6	2	0.02	381	383	399	379
20	N_500_6	2	0.03	377	385	363	379
20	N_500_6	2	0.04	333	347	333	315
20	N_500_6	2	0.05	241	251	291	233
20	N_500_6	2	0.06	191	193	239	173
20	N_500_6	3	0.02	2255	2291	2217	2279
20	N_500_6	3	0.03	2191	2277	2117	2267
20	N_500_6	3	0.04	2059	1883	1891	1955
20	N_500_6	3	0.05	1829	1489	1539	1465
20	N_500_6	3	0.06	1129	943	1031	927
20	N_500_7	2	0.02	379	379	387	379
20	N_500_7	2	0.03	379	373	367	373
20	N_500_7	2	0.04	357	365	331	341
20	N_500_7	2	0.05	207	223	295	191
20	N_500_7	2	0.06	129	143	199	119
20	N_500_7	3	0.02	2281	2287	2237	2277
20	N_500_7	3	0.03	2265	2247	2071	2237
20	N_500_7	3	0.04	2057	1615	1797	1991
20	N_500_7	3	0.05	1255	1111	1241	1089
20	N_500_7	3	0.06	697	621	777	581
20	N_500_8	2	0.02	379	371	401	359
20	N_500_8	2	0.03	279	327	335	291

Asset	Dataset	Cardinality	Return	B&B	BCC-F	BCC-I	BCC-R
20	N_500_8	2	0.04	245	235	287	221
20	N_500_8	2	0.05	167	189	317	163
20	N_500_8	2	0.06	131	151	215	125
20	N_500_8	3	0.02	2275	2237	1999	2193
20	N_500_8	3	0.03	1877	1873	1877	1845
20	N_500_8	3	0.04	1425	1373	1555	1367
20	N_500_8	3	0.05	1105	1001	1271	995
20	N_500_8	3	0.06	633	609	859	589
20	N_500_9	2	0.02	379	387	367	379
20	N_500_9	2	0.03	231	245	323	229
20	N_500_9	2	0.04	135	147	183	131
20	N_500_9	2	0.05	105	115	165	97
20	N_500_9	2	0.06	69	85	119	67
20	N_500_9	3	0.02	2265	2231	1973	2273
20	N_500_9	3	0.03	1595	1431	1503	1423
20	N_500_9	3	0.04	936	895	967	883
20	N_500_9	3	0.05	675	629	735	619
20	N_500_9	3	0.06	381	385	435	373
20	N_500_10	2	0.02	243	233	293	331
20	N_500_10	2	0.03	181	187	217	177
20	N_500_10	2	0.04	151	157	215	149
20	N_500_10	2	0.05	119	129	163	111
20	N_500_10	2	0.06	81	93	73	67
20	N_500_10	3	0.02	1587	1481	1585	1921
20	N_500_10	3	0.03	1257	1243	1347	1213
20	N_500_10	3	0.04	887	859	915	823
20	N_500_10	3	0.05	479	479	505	447
20	N_500_10	3	0.06	263	243	307	223

Table B.1: Performance of cut application strategies on quadratic-cardinality-constrained AAPs.

Bibliography

[1] Tobias Achterberg. *Constraint Integer Programming.* PhD thesis, Technische Universität Berlin, 2008.

[2] Tobias Achterberg and Timo Berthold. Improving the feasibility pump. *Discrete Optimization*, 4(1):77–86, 2007.

[3] Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005.

[4] Tobias Achterberg, Timo Berthold, Thorsten Koch, and Kati Wolter. Constraint integer programming: A new approach to integrate CP and MIP. In Laurent Perron and Michael A. Trick, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 6–20. Springer, 2008.

[5] M. Selim Aktürk, Alper Atamtürk, and Sinan Gürel. A strong conic quadratic reformulation for machine-job assignment with controllable processing times. *Operations Research Letters*, 37(3):187–191, 2009.

[6] Farid Alizadeh and Donald Goldfarb. Second-order cone programming. *Mathematical Programming*, 95(1):3–51, 2003.

[7] Erling D. Andersen and Knud D. Andersen. The MOSEK optimization software. *EKA Consulting ApS, Denmark*, 2000.

[8] Erling D. Andersen, Cornelis Roos, and Tamás Terlaky. On implementing a primal-dual interior-point method for conic quadratic optimization. *Mathematical Programming*, 95(2):249–277, 2003.

[9] Kent Andersen and Anders Nedergaard Jensen. Intersection cuts for mixed integer conic quadratic sets. In Michel Goemans and José Correa, editors, *Integer Programming and Combinatorial Optimization*, pages 37–48. Springer, 2013.

[10] Alper Atamtürk and Vishnu Narayanan. Conic mixed-integer rounding cuts. *Mathematical Programming*, 122(1):1–20, 2010.

[11] Alper Atamtürk and Vishnu Narayanan. Lifting for conic mixed-integer programming. *Mathematical Programming*, 126(2):351–363, 2011.

[12] Alper Atamtürk, Laurent Flindt Muller, and David Pisinger. Separation and extension of cover inequalities for secondorder conic knapsack constraints with generalized upper bounds. Technical report, Technical report, Department of Management Engineering, Technical University of Denmark, Denmark, 2011.

[13] Alper Atamtürk, Gemma Berenguer, and Zuo-Jun Shen. A conic integer programming approach to stochastic joint location-inventory problems. *Operations Research*, 60(2):366–381, 2012.

[14] Egon Balas. Intersection cuts—a new type of cutting planes for integer programming. *Operations Research*, 19(1):19–39, 1971.

[15] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical programming*, 58(1-3): 295–324, 1993.

[16] Pietro Belotti, Julio C. Góez, Imre Pólik, Ted K. Ralphs, and Tamás

Terlaky. On families of quadratic surfaces having fixed intersections with two hyperplanes. *Discrete Applied Mathematics*, 161(16):2778–2793, 2013.

[17] Pietro Belotti, Julio C. Góez, Imre Pólik, Ted K. Ralphs, and Tamás Terlaky. A conic representation of the convex hull of disjunctive sets and conic cuts for integer second order cone optimization. In Mehiddin Al-Baali, Lucio Grandinetti, and Anton Purnama, editors, *Numerical Analysis and Optimization*, pages 1–35. Springer, 2015.

[18] Pietro Belotti, Julio C. Góez, Imre Pólik, Ted K. Ralphs, and Tamás Terlaky. A complete characterization of disjunctive conic cuts for mixed integer second order cone optimization. *Discrete Optimization*, 2016. ISSN 1572-5286. doi: http://dx.doi.org/10.1016/j.disopt.2016.10.001.

[19] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. SIAM, 2001.

[20] Hande Y. Benson and Ümit Sağlam. Mixed-integer second-order cone programming: A survey. In Huseyin Topaloglu, J. Cole Smith, and Harvey J. Greenberg, editors, *Theory Driven by Influential Applications*, pages 13–36. INFORMS, 2013.

[21] Hande Y. Benson and David F. Shanno. An exact primal–dual penalty method approach to warmstarting interior-point methods for linear programming. *Computational Optimization and Applications*, 38(3):371–399, 2007.

[22] Livio Bertacco, Matteo Fischetti, and Andrea Lodi. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization*, 4(1):63–76, 2007.

[23] Timo Berthold. Primal Heuristics for Mixed Integer Programs. Master's thesis, Technische Universität Berlin, 2006.

[24] Timo Berthold. *Heuristic Algorithms in Global MINLP Solvers*. PhD thesis, Technische Universität Berlin, 2014.

[25] Timo Berthold, Stefan Heinz, and Stefan Vigerske. Extending a CIP framework to solve MIQCPs. In Jon Lee and Sven Leyffer, editors, *Mixed Integer Nonlinear Programming*, pages 427–444. Springer, 2012.

[26] Daniel Bienstock. Computational study of a family of mixed-integer quadratic programming problems. *Mathematical Programming*, 74(2):121–140, 1996.

[27] Pierre Bonami and João P.M. Gonçalves. Heuristics for convex mixed integer nonlinear programs. *Computational Optimization and Applications*, 51(2):729–747, 2012.

[28] Pierre Bonami and Miguel A. Lejeune. An exact solution approach for portfolio optimization problems under stochastic and integer constraints. *Operations Research*, 57(3):650–670, 2009.

[29] Pierre Bonami, Lorenz T. Biegler, Andrew R. Conn, Gérard Cornuéjols, Ignacio E. Grossmann, Carl D. Laird, Jon Lee, Andrea Lodi, François Margot, Nicolas Sawaya, and Andreas Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204, 2008.

[30] Pierre Bonami, Mustafa Kilinç, and Jeff Linderoth. Algorithms and software for convex mixed integer nonlinear programs. In Jon Lee and Sven Leyffer, editors, *Mixed Integer Nonlinear Programming*, pages 1–39. Springer, 2012.

[31] Aykut Bulut. *Computational Approaches to Mixed Integer Second Order Cone Optimization*. PhD thesis, Lehigh University, 2017.

[32] Sertalp B. Çay, Julio C. Góez, and Tamás Terlaky. Effects of disjunctive conic cuts within a branch and conic cut algorithm to solve asset allocation problems. Technical Report 16T-005, Lehigh University, 2016.

[33] Sertalp B. Çay, Imre Pólik, and Tamás Terlaky. Warm-start of interior point methods for second order cone optimization via rounding over optimal Jordan frames. Technical Report 17T-006, Lehigh University, 2017.

[34] Sertalp B. Çay. Random portfolio dataset generator. http://sertalpbilal.github.io/randomportfolio/, 2016. URL http://dx.doi.org/10.5281/zenodo.53204. Accessed: 04/25/2016.

[35] Mehmet Tolga Çezik and Garud Iyengar. Cuts for mixed 0-1 conic programming. *Mathematical Programming*, 104(1):179–202, 2005.

[36] John W. Chinneck. Practical optimization: a gentle introduction. *Systems and Computer Engineering), Carleton University, Ottawa. http://www. sce. carleton. ca/faculty/chinneck/po. html*, 2006.

[37] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Polyhedral Approaches to Mixed Integer Linear Programming*. Springer, 2010.

[38] Gerard Cornuejols and Reha H. Tütüncü. *Optimization Methods in Finance*, volume 5. Cambridge University Press, 2006.

[39] ILOG CPLEX. High-performance software for mathematical programming and optimization, 2005.

[40] Etienne De Klerk. *Aspects of Semidefinite Programming: Interior Point Algorithms and Selected Applications*, volume 65. Springer Science & Business Media, 2006.

[41] Sarah Drewes. *Mixed Integer Second Order Cone Programming*. PhD thesis, Technische Universität Darmstadt, 2009.

[42] Marco A. Duran and Ignacio E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36(3):307–339, 1986.

[43] Alexander Engau. Recent progress in interior-point methods: Cutting-plane algorithms and warm starts. In Miguel F. Anjos and Jean B. Lasserre, editors, *Handbook on Semidefinite, Conic and Polynomial Optimization*, pages 471–498. Springer, 2012.

[44] Alexander Engau, Miguel F. Anjos, and Anthony Vannelli. On interior-point warmstarts for linear and combinatorial optimization. *SIAM Journal on Optimization*, 20(4):1828–1861, 2010.

[45] Jacques Faraut and Adam Korányi. *Analysis on Symmetric Cones*. Oxford Science Publications, 1994.

[46] FICO™ Xpress Optimization Suite. *Xpress-Optimizer, Reference manual*, Fair Isaac Corporation, 2009.

[47] Jonathan E. Fieldsend, John Matatko, and Ming Peng. Cardinality constrained portfolio optimisation. In Zheng Rong Yang, Hujun Yin, and Richard M. Everson, editors, *IDEAL*, volume 4, pages 788–793. Springer, 2004.

[48] Matteo Fischetti, Fred Glover, and Andrea Lodi. The feasibility pump. *Mathematical Programming*, 104(1):91–104, 2005.

[49] Robert M. Freund. A potential-function reduction algorithm for solving a linear program directly from an infeasible "warm start". *Mathematical Programming*, 52(1-3):441–466, 1991.

[50] Henrik A. Friberg. CBLIB 2014: a benchmark library for conic mixed-integer and continuous optimization. *Mathematical Programming Computation*, 8(2): 191–214, 2016.

[51] Fabio Furini, Emiliano Traversi, Pietro Belotti, Antonio Frangioni, Ambros Gleixner, Nick Gould, Leo Liberti, Andrea Lodi, Ruth Misener, Hans Mittelmann, Nikolaos Sahinidis, Stefan Vigerske, and Angelika Wiegele.

QPLIB: A library of quadratic programming instances. Technical report, February 2017. URL http://www.optimization-online.org/DB_HTML/2017/02/5846.html.

[52] Arthur M. Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, 1972.

[53] Kimia Ghobadi, Hamid R. Ghaffari, Dionne M. Aleman, David A. Jaffray, and Mark Ruschin. Automated treatment planning for a dedicated multi-source intracranial radiosurgery treatment unit using projected gradient and grassfire algorithms. *Medical Physics*, 39(6):3134–3141, 2012.

[54] Julio C. Góez. *Mixed Integer Second Order Cone Optimization - Disjunctive Conic Cuts: Theory and Experiments*. PhD thesis, Department of Industrial and Systems Engineering, Lehigh University, 2013.

[55] Jacek Gondzio. Warm start of the primal-dual method applied in the cutting-plane scheme. *Mathematical Programming*, 83(1):125–144, 1998.

[56] Jacek Gondzio and Pablo González-Brevis. A new warmstarting strategy for the primal-dual column generation method. *Mathematical Programming*, pages 1–34, 2012.

[57] Jacek Gondzio and Andreas Grothey. Reoptimization with the primal-dual interior point method. *SIAM Journal on Optimization*, 13(3):842–864, 2002.

[58] Jacek Gondzio and Andreas Grothey. A new unblocking technique to warmstart interior point methods based on sensitivity analysis. *SIAM Journal on Optimization*, 19(3):1184–1210, 2008.

[59] Jacek Gondzio and Tamás Terlaky. A computational view of interior point methods. *Advances in Linear and Integer Programming, Oxford University Press: Oxford*, pages 103–144, 1996.

[60] Jacek Gondzio, Pablo González-Brevis, and Pedro Munari. New developments in the primal–dual column generation technique. *European Journal of Operational Research*, 224(1):41–51, 2013.

[61] Zonghao Gu, Edward Rothberg, and Robert E. Bixby. Gurobi optimizer reference manual, version 6.0. *Gurobi Optimization Inc., Houston, USA*, 2014.

[62] Hassan Hijazi, Pierre Bonami, and Adam Ouorou. Robust delay-constrained routing in telecommunications. *Annals of Operations Research*, 206(1):163–181, 2013.

[63] Markus Hirschberger, Yue Qi, and Ralph E. Steuer. Randomly generating portfolio-selection covariance matrices with specified distributional characteristics. *European Journal of Operational Research*, 177(3):1610–1625, 2007.

[64] IBM ILOG CPLEX. V12. 1: Users manual for CPLEX. *International Business Machines Corporation*, 46(53):157, 2009.

[65] Elizabeth John and E. Alper Yıldırım. Implementation of warm-start strategies in interior-point methods for linear programming in fixed dimension. *Computational Optimization and Applications*, 41(2):151–183, 2008.

[66] Miroslav Karamanov and Gérard Cornuéjols. Branching on general disjunctions. *Mathematical Programming*, 128(1-2):403–436, 2011.

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

[68] Fatma Kılınç-Karzan. On minimal valid inequalities for mixed integer conic programs. *Mathematics of Operations Research*, 41(2):477–510, 2015.

[69] Fatma Kılınç-Karzan and Sercan Yıldız. Two-term disjunctions on the second-order cone. In Jon Lee and Jens Vygen, editors, *Integer Programming and Combinatorial Optimization*, pages 345–356. Springer, 2014.

[70] Eva K. Lee and John E. Mitchell. Integer programming: branch and bound methods integer programming: Branch and bound methods. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 1634–1643. Springer, 2009.

[71] Jeff T. Linderoth and Martin W. P. Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11(2):173–187, 1999.

[72] Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. Applications of second-order cone programming. *Linear Algebra and Its Applications*, 284(1):193–228, 1998.

[73] Irvin J. Lustig, Roy E. Marsten, and David F. Shanno. Interior point methods for linear programming: Computational state of the art. *ORSA Journal on Computing*, 6(1):1–14, 1994.

[74] Renata Mansini and Maria Grazia Speranza. Heuristic algorithms for the portfolio selection problem with minimum transaction lots. *European Journal of Operational Research*, 114(2):219–233, 1999.

[75] Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[76] Alexander Martin. General mixed integer programming: Computational issues for branch-and-cut algorithms. In Michael Jünger and Denis Naddef, editors, *Computational Combinatorial Optimization*, pages 1–25. Springer, 2001.

[77] Sanjay Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[78] John E. Mitchell and Michael J. Todd. Solving combinatorial optimization problems using Karmarkar's algorithm. *Mathematical Programming*, 56(1-3): 245–284, 1992.

[79] Sina Modaresi, Mustafa R. Kılınç, and Juan Pablo Vielma. Split cuts and extended formulations for mixed integer conic quadratic programming. *Operations Research Letters*, 43(1):10–15, 2015.

[80] Sina Modaresi, Mustafa R. Kılınç, and Juan Pablo Vielma. Intersection cuts for nonlinear integer programming: Convexification techniques for structured sets. *Mathematical Programming*, 155(1-2):575–611, 2016.

[81] APS MOSEK. The MOSEK optimization software, 2010.

[82] Pedro Munari and Jacek Gondzio. Using the primal-dual interior point algorithm within the branch-price-and-cut method. *Computers & Operations Research*, 40(8):2026–2036, 2013.

[83] Masakazu Muramatsu and Tsunehiro Suzuki. A new second-order cone programming relaxation for max-cut problems. *Journal of Operations Research of Japan*, 43:164–177, 2003.

[84] Vishnu Bhama Narayanan. *Branch-and-Cut Algorithms for Conic Mixed-Integer Programming*. PhD thesis, University of California, Berkeley, 2008.

[85] Yurii E. Nesterov and Michael J. Todd. Self-scaled barriers and interior-point methods for convex programming. *Mathematics of Operations Research*, 22(1): 1–42, 1997.

[86] Mohammad R. Oskoorouchi and John E. Mitchell. A second-order cone cutting surface method: complexity and application. *Computational Optimization and Applications*, 43(3):379–409, 2009.

[87] Mohammad R. Oskoorouchi, Hamid R. Ghaffari, Tamás Terlaky, and Dionne M. Aleman. An interior point constraint generation algorithm for semi-infinite optimization with health-care application. *Operations Research*, 59(5):1184–1197, 2011.

[88] Mustafa Ç. Pınar. Mixed-integer second-order cone programming for lower hedging of American contingent claims in incomplete markets. *Optimization Letters*, 7(1):63–78, 2013.

[89] Imre Pólik and Julio C. Góez. Rounding solutions in SOCP. ICCOPT, Lisbon, Portugal, 2013.

[90] Imre Pólik and Tamás Terlaky. Interior point methods for nonlinear optimization. In Gianni Di Pillo and Fabio Schoen, editors, *Nonlinear Optimization*, pages 215–276. Springer, 2010.

[91] Roman Polyak. Modified barrier functions (theory and methods). *Mathematical Programming*, 54(1-3):177–222, 1992.

[92] Cornelis Roos, Tamás Terlaky, and Jean-Philiipe Vial. *Interior Point Methods for Linear Optimization*. Springer, 2006.

[93] Ümit Saglam. *Advanced Optimization and Statistical Methods in Portfolio Optimization and Supply Chain Management*. PhD thesis, Drexel University, 2014.

[94] Kartik Krishnan Sivaramakrishnan, Gema Plaza, and Tamás Terlaky. A conic interior point decomposition approach for large scale semidefinite programming. Technical report, Technical report, Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205, 2005.

[95] Anders Skajaa. *The Homogeneous Interior-Point Algorithm: Nonsymmetric*

Cones, Warmstarting, and Applications. PhD thesis, Technical University of Denmark, Department of Informatics and Mathematical Modeling, 2013.

[96] Anders Skajaa, Erling D. Andersen, and Yinyu Ye. Warmstarting the homogeneous and self-dual interior point method for linear and conic quadratic problems. *Mathematical Programming Computation*, 5(1):1–25, 2013.

[97] Robert A. Stubbs and Sanjay Mehrotra. A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming*, 86(3):515–532, 1999.

[98] Jos F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1-4):625–653, 1999.

[99] Jos F. Sturm. Implementation of interior point methods for mixed semidefinite and second order cone optimization problems. *Optimization Methods and Software*, 17(6):1105–1154, 2002.

[100] Tamás Terlaky, editor. *Interior Point Methods of Mathematical Programming*, volume 5. Springer Science & Business Media, 1996.

[101] Tamás Terlaky and Imre Pólik. Parametric second order cone optimization and its applications: Challenges and perspectives. NSF Grant Proposal, 2010. Lehigh University.

[102] Tamás Terlaky and Zhouhong Wang. On the identification of the optimal partition of second order cone optimization problems. *SIAM Journal on Optimization*, 24(1):385–414, 2014.

[103] Kim-Chuan Toh, Michael J. Todd, and Reha H. Tütüncü. SDPT3 – a MATLAB software package for semidefinite programming. *Optimization Methods and Software*, 11(1-4):545–581, 1999.

[104] Juan Pablo Vielma, Shabbir Ahmed, and George L. Nemhauser. A lifted linear programming branch-and-bound algorithm for mixed-integer conic quadratic programs. *INFORMS Journal on Computing*, 20(3):438–450, 2008.

[105] Tapio Westerlund and Frank Pettersson. An extended cutting plane method for solving convex MINLP problems. *Computers & Chemical Engineering*, 19: 131–136, 1995.

[106] Stephen J. Wright. *Primal-Dual Interior-Point Methods*, volume 54. SIAM, 1997.

[107] Yu Xia. A Newton's method for perturbed second-order cone programs. *Computational Optimization and Applications*, 37(3):371–408, 2007.

[108] Yinyu Ye. *Interior-Point Algorithm: Theory and Analysis*. Wiley, 1997.

[109] E. Alper Yildirim and Stephen J. Wright. Warm-start strategies in interior-point methods for linear programming. *SIAM Journal on Optimization*, 12(3): 782–810, 2002.

[110] Sercan Yıldız and Gérard Cornuéjols. Disjunctive cuts for cross-sections of the second-order cone. *Operations Research Letters*, 43(4):432–437, 2015.

[111] Kazuo Yonekura and Yoshihiro Kanno. Second-order cone programming with warm start for elastoplastic analysis with von Mises yield criterion. *Optimization and Engineering*, 13(2):181–218, 2012.

Vita

Sertalp B. Çay was born in Ankara, Turkey in 1988 to Mustafa Çay and Fatma Çay. He received his Bachelors degree in Industrial Engineering from Bilkent University, Turkey in 2010. He received his Masters degree in Industrial Engineering from Bilkent University, Turkey in 2013. He obtained high honor standings for his successes in both B.S. and M.S studies. He entered the Ph.D. program at Lehigh University in 2012. During his Ph.D., he focused on developing cutting-edge optimization algorithms and software, optimization modeling and algebraic languages. From 2014 to 2018, he worked as summer and year-round intern positions at SAS Institute under Advanced Analytics R&D division. He will be joining SAS Institute as an Operations Research Specialist after his Ph.D.