Successive Convex Relaxation Methods for Nonconvex Quadratic Optimization Problems

Akiko TAKEDA

Submitted in partial fulfillments of
the requirement for the degree of
DOCTOR OF SCIENCE

Department of Mathematical and Computing Sciences
Tokyo Institute of Technology

March 2001

Acknowledgment

I would like to express my profound gratitude to my adviser Professor Masakazu Kojima for introducing me to the successive convex relaxation method, explaining its theoretical framework and guiding me through the project. Many discussions with him always stimulated further research and finally, led me to the completion of the thesis.

I also would like to thank my co-adviser Professor Yang Dai for her generous guidance. She helped me to write up my first paper on the practical successive convex relaxation method. Our collaboration greatly influenced my mathematical writing style.

I would like to thank all other collaborators. Dr. Katsuki Fujisawa assisted me in parallel implementation of successive convex relaxation methods. Mr. Yusuke Fukaya gave me some idea on our relaxation models suitable for parallel computing. Also, Mr. Mituhiro Fukuda cooperated with me to implement the first version of successive convex relaxation methods. I have had plenty of intriguing discussions with them.

Professor Satoshi Matsuoka and the members of his laboratory offered me to use their advanced PC cluster for my research. Thanks to their kind cooperation, I could develop parallel successive convex relaxation algorithms on their parallel computing system.

I am also deeply indebted to Professor Hisakazu Nishino and Professor Yasushi Masuda of Keio University. While I was in Keio University, I had enjoyed studying mathematical programming through the seminars with them. They had supported me with their continuing encouragement since my graduation from Keio University.

I am grateful to Professor Sunyoung Kim of Ewha Women's University, Korea. She gave me valuable advice on my English conversation and writing, while she was visiting Tokyo Institute of Technology. Her positive support cheered me up, when I had a hard time devoting myself to this thesis.

Special thanks are due to all other members of Professor Kojima's, Professor Takahashi's and Professor Miyoshi's laboratories for their friendship and encouragements. I really had a good time with them. I also would like to express my sincere thanks to technical administrators of the computer systems in the laboratories for their considerable work.

Finally, I would like to thank my parents and my sister for their continuing supports.

Contents

1 Introduction **1**

2 Preliminaries on Successive Convex Relaxation Methods **7**

2.1 Examples of Nonconvex Quadratic Optimization Programs 7

2.2 Two Prototypes of Successive Convex Relaxation Methods 11

2.3 Successive Convex Relaxation Models 14

2.3.1 Conceptual Models 16

2.3.2 Discretized-Localized Models 17

2.4 Related Work to Successive Convex Relaxation Methods 19

2.4.1 The Lift-and-Project Procedure for 0-1 Integer Programs 19

2.4.2 The Reformulation-Linearization Technique for QOPs 23

3 Complexity Analysis for Conceptual Relaxation Methods **26**

3.1 Accuracy Measures of Convex Relaxation 27

3.2 A Spherical-SDP Model 32

3.3 Rank-2 Models 35

3.3.1 Convex Cones of Rank-2 Functions 36

3.3.2 Complexity Analysis 39

3.4 Some Remarks 44

4 Practical Algorithms and Their Implementations **47**

4.1 Practical Successive Convex Relaxation Methods 47

4.1.1 Choices of δ-nets 48

4.1.2 Algorithms 50

4.1.3 Other Versions of the Algorithms 51

4.2 Computational Experiments on QOPs 52
4.2.1 Test Problems 52
4.2.2 Numerical Results 53

5 An Application of Successive Convex Relaxation Methods to Bilevel Quadratic Optimization Problems 61
5.1 Transformation into a QOP Formulation 63
5.2 Special Techniques for BQOPs 65
5.2.1 Scaling Lagrange Multipliers 65
5.2.2 Tightening Upper Bounds for Each Complementary Pair of Variables 67
5.2.3 An Illustrating Example 68
5.3 Computational Experiments on BQOPs 69
5.3.1 Some Implementation Details 69
5.3.2 Numerical Results 70

6 Parallel Implementations of Successive Convex Relaxation Methods 75
6.1 Properties of Previous SCRMs 76
6.2 Parallel Successive Convex Relaxation Methods 79
6.2.1 An Effective Technique for Reducing Inequalities 79
6.2.2 A Parallel Algorithm 82
6.3 Computational Experiments 85
6.3.1 Test Problems 85
6.3.2 Numerical Results 87

7 Conclusions and Future Research 93

Bibliography 94

Chapter 1

Introduction

Quadratic optimization constitutes one of the most important areas of nonlinear programming. The importance of quadratic optimization models is due to several reasons:

(a) Quadratic functions are the simplest nonlinear smooth functions whose derivatives are readily available and easy to manipulate.

(b) Any twice differentiable function can be approximated by a quadratic function in a neighborhood of a given point, so in a sense quadratic models are very natural.

(c) Numerous applications in economics, engineering, and other fields lead to quadratic nonconvex optimization problems. Recent engineering applications include a product design problem from the semiconductor industry [7] and scheduling bottleneck operations [28]. Quadratic constraints often appear in facility location problems [8, 29]. Also, the sensitivity analysis on the efficiency evaluation with Data Envelopment Analysis (DEA) utilizes quadratic optimization models [66].

In addition to these direct applications, quadratic optimization problems (abbreviated by QOPs) cover various important nonconvex mathematical programs such as linear and quadratic 0-1 integer programs [45], linear complementarity problems [11], bilinear matrix inequalities [24, 38], bilevel linear and quadratic programs [70], sum of linear fractional programs [16, 51], and many other programming problems. For instance, a 0-1 constraint of the form $x \in \{0, 1\}$ can be written as the quadratic constraint $x(x-1) = 0$. See Section 2.1 for more detailed transformations from the above-mentioned problems into QOPs.

It is clear that QOPs have great importance both from mathematical and application viewpoints. Thus, QOPs have attracted many researchers since the 1970s, and a large number of approaches have been proposed especially for a linearly constrained QOP. See the references [18, 27, 67, etc.] for those various approaches. The problem consists of a quadratic objective function and a set of linear inequality constraints, as shown below:

$$\left.\begin{array}{ll} \max & f(\boldsymbol{x}) = \gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}. \end{array}\right\} \tag{1.1}$$

Here $\gamma \in R$, $\boldsymbol{q} \in R^n$, $\boldsymbol{b} \in R^m$, $\boldsymbol{A} \in R^{m \times n}$ and $\boldsymbol{Q}$ is an $n \times n$ symmetric matrix. According to the nature of the quadratic matrix $\boldsymbol{Q}$, (1.1) can be classified as convex quadratic problems, bilinear problems, concave quadratic problems, indefinite quadratic problems, etc. For each individual class of (1.1), global optimization algorithms have been developed. When $\boldsymbol{Q}$ is not positive semidefinite, (1.1) becomes a nonconvex problem, known as a tough problem to solve. Pardalos and Vavasis [43] showed that even the simplest QOP whose matrix $\boldsymbol{Q}$ has just one negative eigenvalue:

$$\min\{-x_1^2 + \boldsymbol{c}^T\boldsymbol{x} \ : \ \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \ \boldsymbol{x} \geq \boldsymbol{0}\}$$

is an NP-hard problem. Nevertheless, quite practical algorithms exist which can solve the above problem in a time usually no longer than the time needed for solving a few linear programs (abbreviated by LPs) of the same size (see, e.g., Konno, Thach and Tuy [34]). More additional quadratic constraints to (1.1) complicate the problem significantly.

In this thesis, we consider the most general class of QOPs formulated as follows:

$$\max \ \boldsymbol{c}^T\boldsymbol{x} \quad \text{subject to} \ \ \boldsymbol{x} \in F, \tag{1.2}$$

where

$$\begin{array}{rcl} \boldsymbol{c} & = & \text{a constant column vector in the } n\text{-dimensional Euclidean space } R^n, \\ F & = & \{\boldsymbol{x} \in R^n : p(\boldsymbol{x}) \leq 0 \ \ (\forall p(\cdot) \in \mathcal{P}_F)\}, \\ \mathcal{P}_F & : & \text{a set of finitely or infinitely many quadratic functions on } R^n. \end{array}$$

When a given QOP has a quadratic objective function such as $\gamma_0 + 2\boldsymbol{q}_0^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_0\boldsymbol{x}$, we can transform the QOP into the form (1.2) by replacing the quadratic objective function by a new variable t and adding $\gamma_0 + 2\boldsymbol{q}_0^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_0\boldsymbol{x} = t$ to the set of constraints. Therefore, (1.2) is the general form of QOPs. Throughout this thesis, we assume that (i) F is a compact set, and (ii) a compact convex set C_0 including F is known. Then F of (1.2) can be expressed as

$$F = \{\boldsymbol{x} \in C_0 : \ p(\boldsymbol{x}) \leq 0 \ \ (\forall p(\cdot) \in \mathcal{P}_F)\}. \tag{1.3}$$

Problem (1.2) has received far less attention than (1.1). One of the reasons is theoretical and practical difficulties in the process of solving such a general problem. Even finding a feasible solution in F can be a difficult task. However, as we have shown at the beginning of this section, a general QOP contains various practical applications and covers all the other categories of quadratic problems as special instances. It is therefore well worth dealing with the most general class of QOPs. In this thesis, focusing on the general QOP (1.2), we develop implementable algorithms from the theoretical framework established by Kojima and Tunçel [32, 33].

As practical solution methods proposed for QOPs (1.2) with little additional special structure, branch-and-bound methods are indispensable. A branch-and-bound algorithm consists of two fundamental operations; bounding and branching. Various bounding techniques for QOPs (1.2) are investigated intensively, since tight upper bounds for (1.2) can enhance the fathoming efficiency in the branch-and-bound algorithm. One bounding technique was proposed by Al-Khayyal and Falk [4] for some type of QOPs (1.2), bilinear problems. Their technique utilizes the convex envelope of each bilinear term to generate bounding problems. Such bounding problems were significantly improved by a *Reformulation-Linearization Technique (RLT)* [55] which Sherali and Alameddine designed. The RLT was further extended to the class of continuous polynomial programs [56], quadratic mixed integer programs [54] and linearly constrained quadratic programs [54]. A branch-and-bound algorithm with such a bounding procedure provides increasingly better approximations as the divided feasible regions become smaller. When all divided regions are sufficiently small, the algorithm will generate a point within any desired tolerance of the optimum solution.

Recently, Kojima and Tunçel [32] proposed quite unconventional outer-approximation procedures; successive convex relaxation methods (abbreviated by SCRMs) for solving QOP (1.2), and established a theoretical basis for their methods. We might regard SCRMs as extensions of the lift-and-project procedure, which was proposed independently by Lovász and Schrijver [35] and Sherali and Adams [53] for 0-1 integer programs, to general QOPs.

Theoretically, starting from an initially known compact convex set C_0 containing F, SCRMs successively construct tighter convex relaxations C_k $(k = 1, 2, ...,)$ of F by repeating the lift-and-project procedure with the use of semidefinite programming (abbreviated by SDP) or semi-infinite LP (abbreviated by SILP) relaxation. See [3, 9, 19, 23, 39, 45, 59, 72] for the SDP relaxation and [55, 57] for the semi-infinite LP relaxation. Therefore, maximizing the linear objective function over a convex relaxation C_k $(k = 0, 1, 2, \ldots)$ of F, we successively obtain improved upper bounds $\{\zeta_k \ (k = 0, 1, 2, \ldots, \ )\}$ for the maximum objective function value ζ^*. While these relaxation methods enjoy global convergence such that $\zeta_k \to \zeta^*$ as $k \to \infty$, they involve an infinite number of semi-infinite SDPs (or semi-infinite LPs) to generate a new convex relaxation C_k of F. To resolve this difficulty, Kojima and Tunçel proposed discretized-localized SCRMs [33] by bringing two new techniques, "discretization" and "localization", into the theoretical framework of previous SCRMs [32]. The "discretization" makes it possible to approximate an infinite number of semi-infinite SDPs (or semi-infinite LPs), which need to be solved at each iteration of the previous SCRMs, by a finite number of standard SDPs (or standard LPs, respectively) with a finite number of inequalities. The "localization" allows one to generate a convex relaxation of F which has a better approximation around a neighborhood of the objective direction $\boldsymbol{c}$. The motivations behind are (1) a better convex relaxation of F in that direction may contribute more in finding a better upper bound for the maximum objective function value, and (2) it eliminates some redundant work to make a convex relaxation of F accurate in unnecessary directions. The paper [33] concluded that for any given small precision $\epsilon > 0$, the discretized-localized SCRMs generate ϵ-approximate upper bounds ζ_k, which satisfy $\zeta^* \leq \zeta_k < \zeta^* + \epsilon$ for the maximum objective function value ζ^*, within a finite number k of iterations. However, they are still impractical because as a higher accuracy ϵ is required, not only the number of the SDPs (or LPs) to be solved at each iteration but also their sizes explode quite rapidly. Above all, the details of discretization and localization have not been studied, and the effect of specifically chosen discretization and localization procedures on the efficiency of SCRMs has not been clarified.

This thesis investigates SCRMs in further detail by highlighting the following issues (i)-(iv) with the aim of deriving "better" bounding procedures than existing techniques. Here a "better" procedure means that it attains a more accurate upper bound for QOP (1.2) with less computational time.

(i) Some remaining concerns of the discretization-localization procedures, which lead to practical SCRMs.

(ii) Special SCRMs to bilevel programs.

(iii) Suitable SCRMs to parallel computing.

(iv) Computational complexity analysis in conceptual SCRMs.

We will discuss the issue (i) in Chapter 4, (ii) in Chapter 5, (iii) in Chapter 6, and (iv) in Chapter 3.

Figure 1.1 depicts the structure of the thesis. There are five chapters sandwiched between the introduction and the conclusion. In Chapter 2, after introducing some numerous applications and mathematical programs which can be formulated as QOP (1.2), we explain the concepts of original SCRMs proposed by Kojima and Tunçel [32, 33]. We also review some theoretical results on their methods, paying attention to properties of global convergence, and present one interesting theorem (Theorem 7.6 of [33]) which relates their SCRMs to three types of lift-and-project procedures. In Chapter 2, we also write some notation and definitions utilized throughout this thesis.

Chapter 3 investigates computational complexity of conceptual SCRMs, proposed in the paper [32]. An example given in Section 7 of [32] shows that the convergence of upper bounds $\{\zeta_k \ (k = 0, 1, 2, \dots, \ )\}$ to the maximum objective function value ζ^* is slower than linear in the worst case. In this chapter, we bound the number of iterations k, which the conceptual SCRMs require to generate an approximation of c.hull(F) with a given "accuracy." We extract some quantity and quality from the input data of QOP (1.2), such as a nonconvexity measure and a nonlinearity measure from the set $\mathcal{P}_F$, the diameter of C_0, the diameter of F, and so on. These quantities directly affect the upper bounds that we will derive for the number k of iterations. The upper bound itself might not be so significant, and might be improved by a more sophisticated analysis. It should be emphasized, however, that the upper bound is a polynomial in these quantities, and that this study provides a new way to analyze the computational complexity for other SCRMs.

The discussions beginning in Chapter 4 are based on some remaining issues of the discretized-localized SCRMs [33]. While the research on the SCRM so far was mainly devoted to its theoretical aspect, its practical aspect has not been explored so much. Indeed, global convergence of the discretized-localized SCRMs was already proved in [33], but it is still unknown how to take several parameters necessary to practical implementation

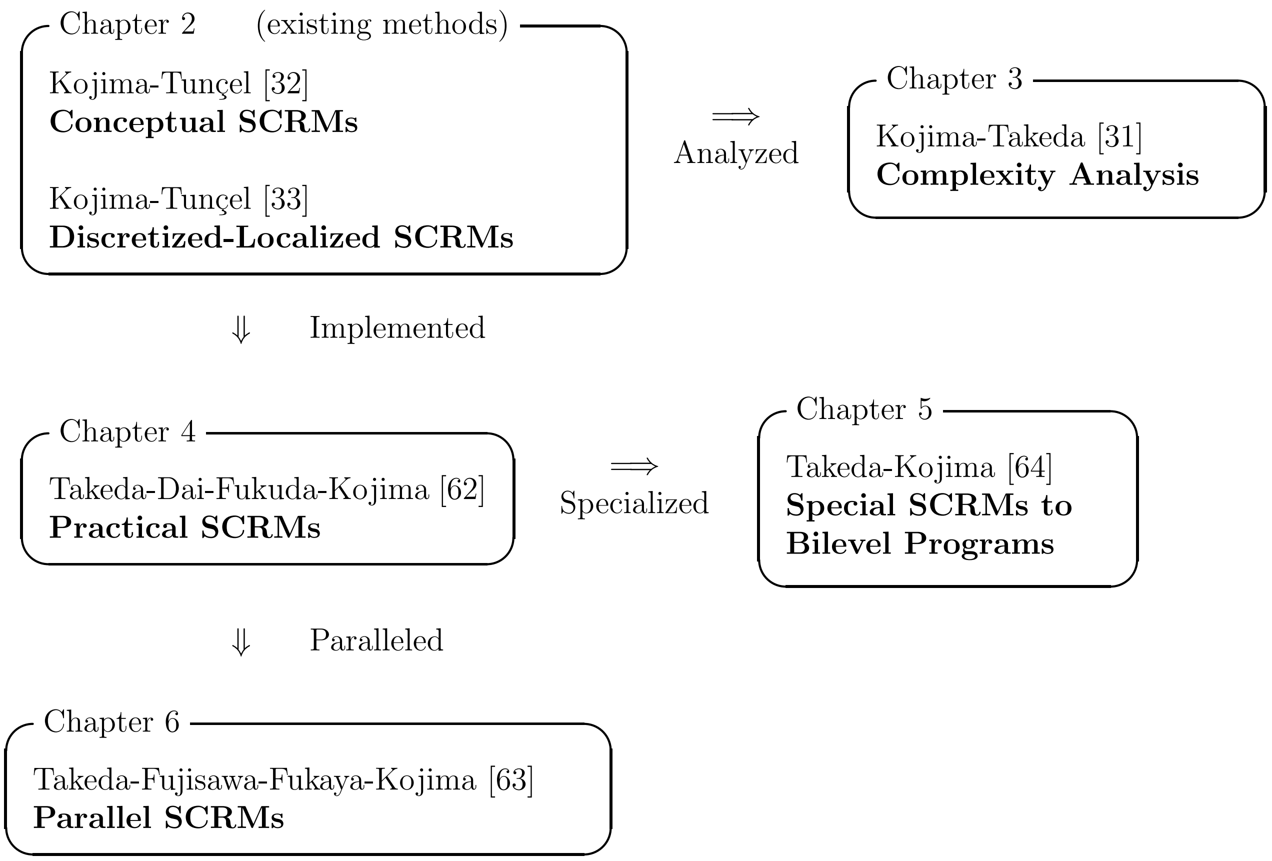


Figure 1.1: Plan of the thesis

of the methods. In Chapter 4, we discuss such practical issues of the discretized-localized SCRMs, and construct practical SCRMs by further slimming down the original discretized-localized SCRMs to overcome the rapid explosion in the sizes of SDPs (or LPs) and in the the number of SDPs (or LPs) to be solved at each iteration. Although these methods have no guarantee to generate ϵ-approximate upper bounds ζ_k $(k = 0, 1, 2, \ldots)$ for a prescribed accuracy $\epsilon > 0$, numerical results look promising.

In Chapter 5, we focus on bilevel quadratic optimization problems (BQOPs) and propose two techniques to transform them into particular cases of QOP (1.2) via the Karush-Kuhn-Tucker (KKT) optimality condition. Moreover, we modify the practical SCRMs proposed in Chapter 4, taking full advantage of a special structure of the transformed QOP (1.2), *i.e.,* complementarity constraints induced from the KKT optimality condition on the lower level problem of a BQOP. An exploitation of the special structure accelerates the SCRMs and generates tighter upper bounds for the maximum objective function value.

As we will show later in Chapter 2, SCRMs solve a large number of SDPs or LPs at every iteration to constitute relaxed convex regions C_k $(k = 1, 2, \ldots)$ of the feasible region F of QOP (1.2). In Chapter 6, we propose parallel versions of the practical SCRMs proposed in Chapter 4, which process some SDPs or LPs at the same time using multiple processors. Also, the SCRMs adopt some technique which decreases the number of constraints included in each SDP or LP considerably, and make it possible to deal with larger dimensional QOPs. We focus our attention on the parallel SCRM with successive SDP relaxations, which obtains accurate upper bounds for QOPs. We discuss parallel execution of an SDP solver called SDPA [20] on a Ninf (network based information library for high performance computing) system. Ninf [50, 52] provides a global network-wide computing infrastructure developed for high-performance numerical computation services.

Finally, Chapter 7 summarizes the results of this thesis, and mentions some future directions of this work.

Chapter 2

Preliminaries on Successive Convex Relaxation Methods

We begin with several examples of QOP (1.2) in this chapter. Though these examples are nonconvex mathematical programs different from QOP (1.2), they can be transformed into (1.2). Thus, Section 2.1 indicates that QOP (1.2) has great importance because of its wide variety of applications. In Sections 2.2 and 2.3, we review previous results on some solution methods of QOP (1.2), known as successive convex relaxation methods (SCRMs) [32, 33]. The properties of these methods are referred in the succeeding chapters, and definitions and notation introduced there are used throughout this thesis. And in Section 2.4, we introduce related work to SCRMs; three types of lift-and-project procedures [9, 35, 53].

2.1 Examples of Nonconvex Quadratic Optimization Programs

A set of test problems for our numerical experiments reported in Chapters 4, 5 and 6 consists of six types of problems.

(a) Minimization of a linear function over linear and quadratic constraints (Chapter 4).

(b) Minimization of a quadratic function with linear constraints (Chapters 4 and 6).

(c) Bilinear programs (Chapter 4).

(d) Mixed quadratic 0-1 integer programs (Chapters 4 and 6).

(e) Sum of linear or quadratic fractional programs (Chapters 4 and 6).

(f) Bilevel quadratic programs (Chapters 4, 5 and 6).

Some problems are from literature, some are randomly generated by publicly available softwares. The sources of test problems or the references of public softwares are specified in the sections of numerical experiments in Chapters 4, 5 and 6. In these chapters, we transform the above problems into QOP (1.2) and apply our solution techniques to the transformed problems. Since the transformation from problems (a)-(c) to QOP (1.2) is straightforward, we only consider the remaining three examples, Examples 2.1.1, 2.1.2 and 2.1.3.

Example 2.1.4 presents a transformation technique from a general nonlinear program to QOP (1.2), according to the paper [30]. We see from this example that QOP (1.2) covers a wide class of mathematical programs. It is, however, difficult to give an explicit expression for such transformation, so that SCRMs can hardly deal with such a nonlinear program in a general way.

The last two examples show interesting applications of QOP (1.2) in the field where nonconvex quadratic optimization models are unfamiliar; Data Envelopment Analysis (Example 2.1.5) and the root finding problem for a polynomial equation system (Example 2.1.6). The paper [66] proposed the quadratic formulation (2.5) of Example 2.1.5 and applied another global optimization technique fully utilizing the structure of (2.5). Example 2.1.6 describes a combinatorial problem induced from the root finding problem, based on the paper [65]. We will show a quadratic formulation for the combinatorial problem in Example 2.1.6.

Example 2.1.1. A general form for *mixed quadratic integer programs* is

$$\begin{array}{ll} \max & \gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \leq \boldsymbol{0}, \\ & x_i \in \{0,1\} \text{ for } i \in \widetilde{\mathcal{N}}, \end{array}$$

where $\boldsymbol{x} \in R^n$, $\gamma \in R$, $\boldsymbol{q} \in R^n$, $\boldsymbol{b} \in R^m$, $\boldsymbol{A} \in R^{m\times n}$ and $\boldsymbol{Q}$ is an $n \times n$ symmetric matrix. Let $\widetilde{\mathcal{N}}$ be a subset of $\{1, 2, \dots, n\}$. Using a new variable x_{n+1} for the objective function and replacing 0-1 integer constraints by quadratic constraints, we rewrite the above problem as

$$\left.\begin{array}{ll} \max & x_{n+1} \\ \text{subject to} & \gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} - x_{n+1} = 0, \\ & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \leq \boldsymbol{0}, \\ & x_i(x_i - 1) = 0 \text{ for } i \in \widetilde{\mathcal{N}}. \end{array}\right\} \tag{2.1}$$

Thus, mixed quadratic integer programs can be easily transformed into QOP (1.2).

Example 2.1.2. Consider a *sum of quadratic fractional programming problem* with the following form

$$\begin{array}{ll} \max & \displaystyle\sum_{\ell=1}^{k} \frac{a_{0\ell} + \boldsymbol{a}_\ell{}^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_\ell\boldsymbol{x}}{b_{0\ell} + \boldsymbol{b}_\ell{}^T\boldsymbol{x}} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \leq \boldsymbol{0}, \end{array}$$

where k is some positive integer, $\boldsymbol{Q}_\ell$ ($\ell = 1, 2, \ldots, k$) are $n \times n$ matrices, $\boldsymbol{a}_\ell$, $\boldsymbol{b}_\ell$ ($\ell = 1, 2, \ldots, k$) are n-dimensional vectors, and $a_{0\ell}, b_{0\ell}$ ($\ell = 1, 2, \ldots, k$) are real values. We assume that $b_{0\ell} + {\boldsymbol{b}_\ell}^T \boldsymbol{x} > 0$ ($\ell = 1, 2, \ldots, k$) for any feasible $\boldsymbol{x}$. By introducing a new variable $x_{n+\ell}$ ($\ell = 1, 2, \ldots, k$) for each fractional term, we can replace the problem with the following equivalent one.

$$\left.\begin{array}{ll} \max & \displaystyle\sum_{\ell=1}^{k} x_{n+\ell} \\ \text{subject to} & a_{0\ell} + {\boldsymbol{a}_\ell}^T \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{Q}_\ell \boldsymbol{x} - b_{0\ell} x_{n+\ell} - (\boldsymbol{b}_\ell^T \boldsymbol{x}) x_{n+\ell} = 0, \quad \ell = 1, 2, \ldots, k \\ & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \leq \boldsymbol{0}. \end{array}\right\} \tag{2.2}$$

Example 2.1.3. A *bilevel quadratic programming problem* can be formulated as

$$\begin{array}{ll} \displaystyle\min_{\boldsymbol{x}} & F(\boldsymbol{x}, \boldsymbol{y}) \\ \text{subject to} & \boldsymbol{y} \in \left\{ \begin{array}{ll} \displaystyle\min_{\boldsymbol{y}} & f(\boldsymbol{x}, \boldsymbol{y}) \\ \text{subject to} & A_1 \boldsymbol{x} + B_1 \boldsymbol{y} \leq \boldsymbol{b}_1 \end{array} \right\}, \\ & A_2 \boldsymbol{x} + B_2 \boldsymbol{y} \leq \boldsymbol{b}_2. \end{array}$$

Here $F(\boldsymbol{x}, \boldsymbol{y})$ is a quadratic function of $\boldsymbol{x}$ and $\boldsymbol{y}$, and $f(\boldsymbol{x}, \boldsymbol{y})$ is a convex quadratic function of $\boldsymbol{y}$ when $\boldsymbol{x}$ is fixed. Then, the Karush-Kuhn-Tucker optimality condition on the inner problem is both necessary and sufficient for the inner optimality, and reduces the above problem to the following one.

$$\left.\begin{array}{ll} \displaystyle\max_{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}} & -F(\boldsymbol{x}, \boldsymbol{y}) \\ \text{subject to} & A_1 \boldsymbol{x} + B_1 \boldsymbol{y} \leq \boldsymbol{b}_1, \\ & \nabla_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y}) + B_1^T \boldsymbol{u} = \boldsymbol{0}, \\ & u_i (A_1 \boldsymbol{x} + B_1 \boldsymbol{y} - \boldsymbol{b}_1)_i = 0, \ \boldsymbol{u} \geq \boldsymbol{0}, \\ & A_2 \boldsymbol{x} + B_2 \boldsymbol{y} \leq \boldsymbol{b}_2. \end{array}\right\} \tag{2.3}$$

The resultant problem involves linear and bilinear constraints, and it can be considered as QOP (1.2) after an appropriate transformation.

Example 2.1.4. Kojima, Matsumoto and Shida [30] pointed out that *a wide class of nonlinear programs* can be reduced to a nonconvex quadratic program of QOP (1.2). More generally, it is known that any closed subset $G \subset R^n$ can be represented as

$$G = \{\boldsymbol{x} \in R^n : g(\boldsymbol{x}) - \|\boldsymbol{x}\|^2 \leq 0\}$$

using some convex function $g(\cdot) : R^n \to R$. See, for example, Corollary 3.5 of [67]. Thus, given any closed subset G of R^n and any compact convex subset C_0 of R^n, we can rewrite maximization of a linear function $\boldsymbol{c}^T \boldsymbol{x}$ over a compact set $G \cap C_0$ as

$$\max \ \boldsymbol{c}^T \boldsymbol{x} \text{ subject to } (\boldsymbol{x}, t) \in \widetilde{F},$$

where

$$\begin{array}{rcl} \widetilde{F} & = & \{(\boldsymbol{x}, t) \in \widetilde{C_0} : \ t - \boldsymbol{x}^T\boldsymbol{x} \leq 0\}, \\ \widetilde{C_0} & = & \{(\boldsymbol{x}, t) \in R^{n+1} : \ g(\boldsymbol{x}) - t \leq 0, \ 0 \leq t \leq \bar{t}, \ \boldsymbol{x} \in C_0\}, \\ g(\cdot) & : & \text{a convex function on } R^n. \end{array}$$

Then, $\widetilde{C_0}$ turns out to be compact and convex, and the resultant problem is a special case of the general nonconvex QOP (1.2). Although this construction is not implementable because an explicit algebraic representation of such a convex function $g(\cdot)$ is usually impossible, it certainly shows theoretical potential of the SCRMs for quite general nonlinear programs.

Example 2.1.5. Takeda and Nishino [66] gave some nonconvex programming formulation for the *sensitivity analysis of the efficiency evaluation* with Data Envelopment Analysis (DEA), and presented the transformation from the formulation to QOP (1.2).

DEA is a useful method to evaluate relative efficiency of multi-input and multi-output units based on observed data. Suppose that we have r DMUs (Decision Making Units); $\text{DMU}_1, \ldots,$ DMU_r with input-output vectors $\boldsymbol{z}_i^T \in R^n$ $(i = 1, \ldots, r)$. The *reference set* denotes a polyhedral set P_0 such as

$$P_0 = \left\{ \boldsymbol{z} \in R^n \ : \ \boldsymbol{z} = \boldsymbol{D}\boldsymbol{\mu}, \quad \boldsymbol{\mu} \geq \boldsymbol{0}, \quad \boldsymbol{\mu} \in R^{(r+n)} \right\}.$$

The matrix $\boldsymbol{D} \in R^{n \times (r+n)}$ is induced from all input-output vectors $\boldsymbol{z}_i \in R^n$ $(i = 1, \ldots, r)$, and unit coordinate vectors $\boldsymbol{e}_i \in R^n$ $(i = 1, \ldots, n)$.

Here we suppose that DMU_0 $(0 \in \{1, \cdots, r\})$ is evaluated as inefficient DMU by the conventional CCR model of DEA. Takeda and Nishino [66] proposed a new technique to assess the sensitivity for the inefficiency classification of DMU_0. Their sensitivity technique formulates the nonconvex quadratic program;

$$\left.\begin{array}{ll} \min & f(\boldsymbol{z}) = (\boldsymbol{z} - \boldsymbol{z}_0)^T \boldsymbol{Q} (\boldsymbol{z} - \boldsymbol{z}_0) \\ \text{subject to} & \boldsymbol{z} \in \overline{R^n \setminus P_0}, \end{array}\right\} \tag{2.4}$$

where $\boldsymbol{Q}$ is a positive definite matrix, $\overline{X}$ denotes the closure of a set X and $R^n \setminus P$ denotes the complement of a convex set P. The inefficiency of DMU_0 ensures that $\boldsymbol{z}_0$ exists in the interior of the region P_0. Using the optimum solution $\boldsymbol{z}^*$ of (2.4), we compute $\sqrt{f(\boldsymbol{z}^*)}$ for the sensitivity measure of DMU_0. As the value of $\sqrt{f(\boldsymbol{z}^*)}$ becomes larger, we treat the inefficiency evaluation of $\boldsymbol{z}_0$ more stable.

Theorem 3.2 of [66] showed that the program (2.4) is equivalent to the following nonconvex quadratic program:

$$\left.\begin{array}{ll} \min & g(\boldsymbol{x}) = -\boldsymbol{z}_0^T \boldsymbol{x} \\ \text{subject to} & \boldsymbol{D}^T \boldsymbol{x} \leq \boldsymbol{0}, \quad \boldsymbol{x}^T \boldsymbol{Q}^{-1} \boldsymbol{x} \geq 1, \\ & \boldsymbol{x} \in R^n. \end{array}\right\} \tag{2.5}$$

We briefly summarize the correspondence between two problems;

$$\begin{array}{ll} \text{(optimum value)} & \sqrt{f(\boldsymbol{z}^*)} = g(\boldsymbol{x}^*), \\ \text{(optimum solution)} & \boldsymbol{z}^* = \boldsymbol{z}_0 + g(\boldsymbol{x}^*) \boldsymbol{Q}^{-1} \boldsymbol{x}^*, \end{array}$$

using two optimum solutions; $\boldsymbol{z}^*$ of (2.4) and $\boldsymbol{x}^*$ of (2.5). Note that (2.5) is formulated as QOP (1.2).

Example 2.1.6. Takeda, Kojima and Fujisawa [65] presented an interesting combinatorial (enumeration) problem, which arises in the initial phase of the polyhedral homotopy continuation method for computing all solutions of a polynomial equation system in complex variables. We show below that QOP (1.2) also covers such a combinatorial problem.

Let $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})) = \boldsymbol{0}$ be a system of n polynomial equations in n complex unknowns $x_i \in C$ $(i = 1, 2, \ldots, n)$, where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in C^n$. Each polynomial $f_i(\boldsymbol{x})$ consists of m_i terms (including a constant term). We denote each monomial as $\boldsymbol{x}^a = x_1^{a^{(1)}} x_2^{a^{(2)}} \ldots x_n^{a^{(n)}}$, and identify it with a lattice point $\boldsymbol{a} = (a^{(1)}, a^{(2)}, \ldots, a^{(n)}) \in Z_+^n \equiv \{0, 1, 2, \ldots\}^n$. Denoting each term of the ith equation as $c_{ij}\ \boldsymbol{x}^{a_{ij}}$ $(j = 1, 2, \ldots, m_i)$ with a coefficient $c_{ij} \in C$ and $\boldsymbol{a}_{ij} \in Z_+^n$, and letting ω_{ij} be a real number chosen generically, we present the combinatorial problem of [65]:

Find a solution $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\alpha_1, \alpha_2, \ldots, \alpha_n, \beta_1, \beta_2, \ldots, \beta_n) \in R^{2n}$ which satisfies

$$\beta_i - \boldsymbol{a}_{ij}^T \boldsymbol{\alpha} \leq \omega_{ij} \quad (i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m_i), \tag{2.6}$$

with 2 equalities for each $i \in \{1, 2, \ldots, n\}$.

This problem can be formulated as the following mathematical program with additional variables $\boldsymbol{s} = (s_{ij} : i = 1, \ldots, n,\ j = 1, \ldots, m_i)$ and $\boldsymbol{x} = (x_{ij} : i = 1, \ldots, n,\ j = 1, \ldots, m_i)$:

$$\left.\begin{array}{lll}
\max & \boldsymbol{d}_1^T \boldsymbol{\alpha} + \boldsymbol{d}_2^T \boldsymbol{\beta} + \boldsymbol{d}_3^T \boldsymbol{s} + \boldsymbol{d}_4^T \boldsymbol{x} & \\
\text{subject to} & \beta_i - \boldsymbol{a}_{ij}^T \boldsymbol{\alpha} + s_{ij} = \omega_{ij} & (i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m_i), \\
& 0 \leq s_{ij} \leq M(1 - x_{ij}),\ x_{ij} \in \{0, 1\} & (i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m_i), \\
& \sum_{j=1}^{m_i} x_{ij} = 2 & (i = 1, 2, \ldots, n),
\end{array}\right\} \tag{2.7}$$

where $M \in R$ is a sufficiently large number. Also, $\boldsymbol{d}_1 \in R^n$, $\boldsymbol{d}_2 \in R^n$, $\boldsymbol{d}_3 \in R^{\sum_{i=1}^n m_i}$ and $\boldsymbol{d}_4 \in R^{\sum_{i=1}^n m_i}$ are arbitrary vectors. Example 2.1.1 ensures that a 0-1 integer constraint $x_{ij} \in \{0, 1\}$ of the problem (2.7) can be expressed by a quadratic constraint, and hence, the combinatorial problem arising in the polyhedral homotopy continuation method can be formulated as QOP (1.2).

2.2 Two Prototypes of Successive Convex Relaxation Methods

At first, we introduce some notation and conditions necessary to describe basic algorithms of SCRMs. Let $\mathcal{S}^n$ and $\mathcal{S}_+^n$ denote the set of $n \times n$ symmetric matrices and the set of $n \times n$ symmetric positive semidefinite matrices, respectively. Also, for two matrices $\forall \boldsymbol{A} \in \mathcal{S}^n$

and $\forall \boldsymbol{B} \in \mathcal{S}^n$, we define a matrix operation such that $\boldsymbol{A} \bullet \boldsymbol{B} \equiv$ the inner product of two symmetric matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, *i.e.*, $\boldsymbol{A} \bullet \boldsymbol{B} \equiv \sum_{i=1}^n \sum_{j=1}^n A_{ij} B_{ij}$.

We often use the notation $qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q})$ to designate the constant term γ, the linear term $2\boldsymbol{q}^T\boldsymbol{x}$ and the quadratic term $\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}$ involved in a quadratic function $p(\cdot)$ defined on R^n;

$$p(\boldsymbol{x}) = qf(\boldsymbol{x}; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \equiv \gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} \ (\forall \boldsymbol{x} \in R^n),$$

where $\gamma \in R$, $\boldsymbol{q} \in R^n$ and $\boldsymbol{Q} \in \mathcal{S}^n$. With this convention, we can write the set $\mathcal{Q}$ of quadratic functions on R^n, the set $\mathcal{Q}_+$ of convex quadratic functions on R^n and the set $\mathcal{L}$ of linear functions on R^n as

$$\begin{aligned} \mathcal{Q} &\equiv \{qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q}) : \gamma \in R, \ \boldsymbol{q} \in R^n, \ \boldsymbol{Q} \in \mathcal{S}^n\}, \\ \mathcal{Q}_+ &\equiv \{qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q}) : \gamma \in R, \ \boldsymbol{q} \in R^n, \ \boldsymbol{Q} \in \mathcal{S}^n_+\}, \\ \mathcal{L} &\equiv \{qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q}) : \gamma \in R, \ \boldsymbol{q} \in R^n, \ \boldsymbol{Q} = \boldsymbol{O}\}, \end{aligned}$$

respectively. Throughout the paper, we assume the following conditions on QOP (1.2).

Condition 2.2.1.

(i) F is compact.

(ii) A compact convex set $C_0 \subseteq R^n$ such that

$$\begin{aligned} F &\subseteq C_0, \\ \eta_0 &\equiv \max\{\|\boldsymbol{x}' - \boldsymbol{x}\| : \boldsymbol{x}', \ \boldsymbol{x} \in C_0\} > 0 \\ &\quad \text{(the diameter of } C_0) \end{aligned}$$

is known in advance.

(iii) $\|\boldsymbol{c}\| = 1$.

Note that if η_0 was zero in (ii) then F would consist of at most one point; hence QOP (1.2) would be trivial. Also (iii) is not essential because we may assume it without loss of generality whenever $\boldsymbol{c} \neq \boldsymbol{0}$. In addition to Condition 2.2.1 above, we need to assume the following condition to implement SCRMs on a computer.

Condition 2.2.2.

(iv) The quadratic inequality representation $\mathcal{P}_F$ of F is finite.

(v) The compact convex set C_0 containing F (see (ii) above) has a finite convex quadratic (or linear) inequality representation $\widetilde{\mathcal{P}}_0$:

$$\left.\begin{array}{l} F \subseteq C_0 \equiv \{\boldsymbol{x} \in R^n : p(\boldsymbol{x}) \leq 0 \ (\forall p(\cdot) \in \widetilde{\mathcal{P}}_0)\} \\ \text{for } \exists \text{ finite } \widetilde{\mathcal{P}}_0 \subseteq \mathcal{Q}_+ \quad (\text{or } \exists \text{ finite } \widetilde{\mathcal{P}}_0 \subseteq \mathcal{L}) \ . \end{array}\right\} \tag{2.8}$$

Let $\mathcal{P}$ be a nonempty subset of quadratic functions, *i.e.*, $\emptyset \neq \mathcal{P} \subset \mathcal{Q}$. (In the algorithms below, we will take $\mathcal{P}$ to be the union of $\mathcal{P}_F$ and $\mathcal{P}_k$ of quadratic functions which induce quadratic valid inequalities for the kth iterate C_k). We use the notation $\widehat{F}(C_0, \mathcal{P})$ for the SDP (semidefinite programming) relaxation, and the notation $\widehat{F}^L(C_0, \mathcal{P})$ for the semi-infinite LP (linear programming) relaxation applied to the set $\{\boldsymbol{x} \in C_0 : qf(\boldsymbol{x}; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \leq 0, \ \forall qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \in \mathcal{P}\}$;

$$\left.\begin{array}{rcl}
\widehat{F}(C_0,\mathcal{P}) & \equiv & \left\{ \boldsymbol{x} \in C_0 : \begin{array}{l} \exists \boldsymbol{X} \in \mathcal{S}^n \text{ such that } \begin{pmatrix} 1 & \boldsymbol{x}^T \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} \in \mathcal{S}^{1+n}_+ \text{ and} \\ \gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{Q} \bullet \boldsymbol{X} \leq 0 \ (\forall qf(\cdot;\gamma,\boldsymbol{q},\boldsymbol{Q}) \in \mathcal{P}) \end{array} \right\} \\
 & \equiv & \text{an SDP relaxation of} \\
 & & \{\boldsymbol{x} \in C_0 : qf(\boldsymbol{x};\gamma,\boldsymbol{q},\boldsymbol{Q}) \leq 0 \ (\forall qf(\cdot;\gamma,\boldsymbol{q},\boldsymbol{Q}) \in \mathcal{P})\}, \\
\widehat{F}^L(C_0,\mathcal{P}) & \equiv & \left\{ \boldsymbol{x} \in C_0 : \begin{array}{l} \exists \boldsymbol{X} \in \mathcal{S}^n \text{ such that} \\ \gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{Q} \bullet \boldsymbol{X} \leq 0 \ (\forall qf(\cdot;\gamma,\boldsymbol{q},\boldsymbol{Q}) \in \mathcal{P}) \end{array} \right\} \\
 & \equiv & \text{a semi-infinite LP relaxation of} \\
 & & \{\boldsymbol{x} \in C_0 : qf(\boldsymbol{x};\gamma,\boldsymbol{q},\boldsymbol{Q}) \leq 0 \ (\forall qf(\cdot\gamma,\boldsymbol{q},\boldsymbol{Q}) \in \mathcal{P})\}.
\end{array}\right\} \quad (2.9)$$

The lemma below provides a fundamental characterization of $\widehat{F}(C_0, \mathcal{P})$ and $\widehat{F}^L(C_0, \mathcal{P})$, which plays an essential role in global convergence analysis of the SCRMs. Our complexity analysis shown in Chapter 3 is based on the lemma. Here, let c.cone($\mathcal{P}$) denote the convex cone generated by $\mathcal{P} \subset \mathcal{Q}$; $\text{c.cone}(\mathcal{P}) \equiv \{\sum_{i=1}^{\ell} \lambda_i p_i(\cdot) : \lambda_i \geq 0, \ p_i(\cdot) \in \mathcal{P} \ (i = 1, 2, \ldots, \ell), \ \ell \geq 0\}$.

Lemma 2.2.3. *([32]) Let $\emptyset \neq \mathcal{P} \subset \mathcal{Q}$.*

(i) $\widehat{F}(C_0, \mathcal{P}) = \{\boldsymbol{x} \in C_0 : p(\boldsymbol{x}) \leq 0 \ (\forall p(\cdot) \in \text{c.cone}(\mathcal{P}) \cap \mathcal{Q}_+)\}$.

(ii) $\widehat{F}^L(C_0, \mathcal{P}) = \{\boldsymbol{x} \in C_0 : p(\boldsymbol{x}) \leq 0 \ (\forall p(\cdot) \in \text{c.cone}(\mathcal{P}) \cap \mathcal{L})\}$.

Proof: See Theorem 2.4 and Corollary 2.5 of [32]. ▮

Kojima and Tunçel [32] presented two types of SCRMs for approximating the convex hull of F (abbreviated by c.hull(F)) with the use of SDP or semi-infinite LP relaxation. We will call the first method the SSDP (Successive Semidefinite Programming) Relaxation Method, and the second the SSILP (Successive Semi-Infinite Linear Programming) Relaxation Method.

Algorithm 2.2.4. (SSDP relaxation method)

Step 0: Let $k = 0$.

Step 1: If $C_k = \emptyset$ then stop. Compute $\zeta_k = \sup\{\boldsymbol{c}^T\boldsymbol{x} \ : \ \boldsymbol{x} \in C_k\}$.

```
Step 2: Choose a set $\mathcal{P}_k \subset \mathcal{Q}$ that induces quadratic valid inequalities for $C_k$.
Step 3: Let $C_{k+1} = \widehat{F}(C_0, \mathcal{P}_F \cup \mathcal{P}_k)$.
Step 4: Let $k = k + 1$ and go to Step 1.
```

Algorithm 2.2.5. (SSILP relaxation method)

```
Step 0: Let $k = 0$.
Step 1: If $C_k = \emptyset$ then stop. Compute $\zeta_k = \sup\{\boldsymbol{c}^T\boldsymbol{x} \ : \ \boldsymbol{x} \in C_k\}$.
Step 2: Choose a set $\mathcal{P}_k \subset \mathcal{Q}$ that induces quadratic valid inequalities for $C_k$.
Step 3: Let $C_{k+1} = \widehat{F}^L(C_0, \mathcal{P}_F \cup \mathcal{P}_k)$.
Step 4: Let $k = k + 1$ and go to Step 1.
```

At each iteration of these methods, we first generate a set $\mathcal{P}_k = \{p(\cdot)\}$ of finitely or infinitely many quadratic functions such that each $p(\boldsymbol{x}) \leq 0$ forms a valid inequality for the kth iterate C_k. Since C_k was chosen to include F at the previous iteration, each $p(\boldsymbol{x}) \leq 0$ serves as a (redundant) valid inequality for F; hence F is represented as

$$F = \{\boldsymbol{x} \in C_0 : p(\boldsymbol{x}) \leq 0 \ \ (\forall p(\cdot) \in \mathcal{P}_F \cup \mathcal{P}_k)\}.$$

In order to generate the next iterate C_{k+1}, we then apply the SDP relaxation or the semi-infinite LP relaxation to the set F with the above representation in terms of the set $\mathcal{P}_F \cup \mathcal{P}_k$ of quadratic functions. (The semi-infinite LP relaxation is also called the Reformulation-Linearization Technique (RLT) in the literature [55, 56, 57]). Apparently, the SDP relaxation is at least as accurate as the semi-infinite LP relaxation, *i.e.*, $\widehat{F}(C_0, \mathcal{P}_F \cup \mathcal{P}_k) \subseteq \widehat{F}^L(C_0, \mathcal{P}_F \cup \mathcal{P}_k)$.

To implement the above algorithms, we need to provide precise description for $\mathcal{P}_k \subset \mathcal{Q}$. The difference among the existing SCRM models lies in the definition of $\mathcal{P}_k$. The next section shows several candidates for $\mathcal{P}_k$, which the papers [32, 33] proposed and studied.

2.3 Successive Convex Relaxation Models

Throughout the paper, we mainly focus our attention on the following two models of SCRMs with the use of the SDP relaxation or the use of the semi-infinite LP relaxation.

- Rank-2-SDP Model: We take $\mathcal{P}_k$ to be a set of rank-2 quadratic functions in Algorithm 2.2.4, and we perform the SDP relaxation.

- Rank-2-SILP Model: We take $\mathcal{P}_k$ to be a set of rank-2 quadratic functions in Algorithm 2.2.5, and we perform the semi-infinite LP relaxation.

Now, we introduce some notation and symbols to define the set $\mathcal{P}_k$ of functions. Let

$$\left.\begin{array}{rcl} \overline{D} & \equiv & \{\boldsymbol{d} \in R^n : \|\boldsymbol{d}\| = 1\} \text{ (the set of unit vectors in } R^n), \\ \pm I & \equiv & \{\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n, -\boldsymbol{e}_1, -\boldsymbol{e}_2, \ldots, -\boldsymbol{e}_n\}, \\ \boldsymbol{e}_i & \equiv & \text{the } i\text{th unit coordinate vector } (i = 1, 2, \ldots, n). \end{array}\right\} \quad (2.10)$$

Let $C' \subset C$ and let C be a compact subset of R^n. For $\forall \boldsymbol{d}$, $\forall \boldsymbol{d}_1$, $\forall \boldsymbol{d}_2 \in \overline{D}$, $\forall \boldsymbol{\xi} \in R^n$, $\forall \rho > 0$ and $\forall \boldsymbol{x} \in R^n$, define

$$\begin{array}{rcl} \alpha(C, \boldsymbol{d}) & \equiv & \sup\{\boldsymbol{d}^T\boldsymbol{x} \ : \ \boldsymbol{x} \in C\}, \\ \ell sf(\boldsymbol{x}; C, \boldsymbol{d}) & \equiv & \boldsymbol{d}^T\boldsymbol{x} - \alpha(C, \boldsymbol{d}), \\ r2sf(\boldsymbol{x}; C, \boldsymbol{d}_1, C', \boldsymbol{d}_2) & \equiv & -(\boldsymbol{d}_1^T\boldsymbol{x} - \alpha(C, \boldsymbol{d}_1))(\boldsymbol{d}_2^T\boldsymbol{x} - \alpha(C', \boldsymbol{d}_2)), \\ r1sf(\boldsymbol{x}; C, \boldsymbol{d}) & \equiv & r2sf(\boldsymbol{x}; C, \boldsymbol{d}, C, -\boldsymbol{d}) \\ & = & -(\boldsymbol{d}^T\boldsymbol{x} - \alpha(C, \boldsymbol{d}))(-\boldsymbol{d}^T\boldsymbol{x} - \alpha(C, -\boldsymbol{d})), \\ \rho(C, \boldsymbol{\xi}) & = & \sup\{\|\boldsymbol{x} - \boldsymbol{\xi}\| : \boldsymbol{x} \in C\}, \\ sf(\boldsymbol{x}; \boldsymbol{\xi}, \rho) & = & (\boldsymbol{x} - \boldsymbol{\xi})^T(\boldsymbol{x} - \boldsymbol{\xi}) - \rho^2. \end{array}$$

The optimum value $\rho(C, \boldsymbol{\xi})$ corresponds to the radius of the minimum ball with a center $\boldsymbol{\xi} \in R^n$ that contains $C \subset R^n$. We call $\ell sf(\cdot; C, \boldsymbol{d})$ *a linear supporting function for* C *in a direction* $\boldsymbol{d} \in \overline{D}$, $r2sf(\cdot; C, \boldsymbol{d}_1, C', \boldsymbol{d}_2)$ *a rank-2 (quadratic) supporting function for* C' *in a pair of directions* $\boldsymbol{d}_1, \boldsymbol{d}_2 \in \overline{D}$, $r1sf(\cdot; C, \boldsymbol{d})$ *a rank-1 (quadratic) supporting function for* C *in a direction* $\boldsymbol{d} \in \overline{D}$, and $sf(\boldsymbol{x}; \boldsymbol{\xi}, \rho)$ *a spherical function with a center* $\boldsymbol{\xi} \in R^n$ *and a radius* $\rho > 0$. Let

$$\begin{array}{rcl} \mathcal{P}^L(C, D) & \equiv & \{\ell sf(\cdot; C, \boldsymbol{d}) : \boldsymbol{d} \in D\} \ \ (\forall D \subseteq \overline{D}), \\ \mathcal{P}^2(C, D_1; C', D_2) & \equiv & \{r2sf(\cdot; C, \boldsymbol{d}_1, C', \boldsymbol{d}_2) : \boldsymbol{d}_1 \in D_1, \ \boldsymbol{d}_2 \in D_2\} \quad (\forall D_1, \ D_2 \subseteq \overline{D}), \\ \mathcal{P}^1(C, D) & \equiv & \{r1sf(\cdot; C, \boldsymbol{d}) : \boldsymbol{d} \in D\} \ \ (\forall D \subseteq \overline{D}), \\ \mathcal{P}^S(C) & \equiv & \{sf(\cdot; \boldsymbol{\xi}, \rho) : \boldsymbol{\xi} \in R^n, \ \rho \geq \rho(C, \boldsymbol{\xi})\}. \end{array}$$

Note that the construction of $\mathcal{P}^L(C, D)$ requires the optimum solutions of the following programs:

$$\alpha(C, \boldsymbol{d}) = \sup\{\boldsymbol{d}^T\boldsymbol{x} \ : \ \boldsymbol{x} \in C\} \qquad \text{for } \forall \boldsymbol{d} \in D.$$

Moreover, to define $\mathcal{P}^2(C, D_1; C', D_2)$, we need to solve

$$\alpha(C, \boldsymbol{d}_1) \ = \ \sup\{\boldsymbol{d}_1^T\boldsymbol{x} \ : \ \boldsymbol{x} \in C\} \qquad \text{for } \forall \boldsymbol{d}_1 \in D_1$$

and

$$\alpha(C', \boldsymbol{d}_2) \ = \ \sup\{\boldsymbol{d}_2^T\boldsymbol{x} \ : \ \boldsymbol{x} \in C'\} \qquad \text{for } \forall \boldsymbol{d}_2 \in D_2.$$

In the following two subsections (Sections 2.3.1 and 2.3.2), we will show several different definitions for the set $\mathcal{P}_k$ constructed in Step 2 of Algorithm 2.2.4 or 2.2.5.

2.3.1 Conceptual Models

According to the paper [32], we present the definitions of the set $\mathcal{P}_k$ for the conceptual versions of Rank-2-SDP and Rank-2-SILP Models, and for one additional model (Spherical-SDP Model).

- Spherical-SDP Model: $\mathcal{P}_k = \mathcal{P}^S(C_k)$ in Algorithm 2.2.4
- Rank-2-SDP Model: $\mathcal{P}_k = \mathcal{P}^2(C_0, D_1; C_k, D_2)$ † with $D_1 = \pm I, D_2 = \overline{D}$ in Algorithm 2.2.4
- Rank-2-SILP Model: $\mathcal{P}_k = \mathcal{P}^2(C_0, D_1; C_k, D_2)$ with $D_1 = \pm I, D_2 = \overline{D}$ in Algorithm 2.2.5

The complexity analysis of the Spherical-SDP Model is much simpler than that of the latter two models, and the former analysis helps an easier understanding of the latter two models, which are of more practical interest.

Note that $\mathcal{P}_k = \mathcal{P}^S(C_k)$ of Spherical-SDP Model and $\mathcal{P}_k = \mathcal{P}^2(C_0, D_1; C_k, D_2)$ of Rank-2 Models provide a quadratic inequality representation of C_k in Algorithm 2.2.4 or 2.2.5. For above three models, we can derive the convergence of the sequence $\{C_k \ (k = 0, 1, 2, \dots, \ )\}$ to the convex hull, c.hull(F), and the convergence of the sequence $\{\zeta_k \ (k = 0, 1, 2, \dots, \ )\}$ to the optimum value ζ^* of QOP (1.2) as Theorem 2.3.1 shows below. However, we should mention that these three models are conceptual, because

- Step 2 of Algorithms 2.2.4 and 2.2.5 requires to compute a continuum of scalars

$$\rho(C_k, \boldsymbol{\xi}) = \sup\{\|\boldsymbol{x} - \boldsymbol{\xi}\| : \boldsymbol{x} \in C_k\} \ \ (\forall \boldsymbol{\xi} \in R^n),$$

  or a continuum of

$$\alpha(C_k, \boldsymbol{d}) = \sup\{\boldsymbol{d}^T\boldsymbol{x} : \boldsymbol{x} \in C_k\} \ \ (\forall \boldsymbol{d} \in \overline{D})$$

  for the construction of $\mathcal{P}_k$, and moreover,

- the feasible region C_k of the above problems includes a continuum of inequalities;

$$\gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{Q} \bullet \boldsymbol{X} \leq 0 \ \ (\forall qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \in \mathcal{P}_{k-1}).$$

Thus, Algorithms 2.2.4 and 2.2.5 of the three models are called conceptual SCRMs. To resolve these difficulties in implementing Rank-2-SDP and Rank-2-SILP Models on a computer, we need to choose a finite number of vectors for D_2. However, with an arbitrary set $D_2 \subseteq \overline{D}$, $\mathcal{P}_k = \mathcal{P}^2(C_0, D_1; C_k, D_2)$ of Rank-2 Models do not form quadratic inequality

†For simplicity of notation, we sometimes use $\mathcal{P}^2(C_k, D_1, D_2)$ instead of $\mathcal{P}^2(C_0, D_1; C_k, D_2)$ in some succeeding chapters.

representations of C_k, *i.e.*, C_k can be a proper subset of $\{\boldsymbol{x} \in R^n : p(\boldsymbol{x}) \leq 0 \ (\forall p(\cdot) \in \mathcal{P}_k)\}$. In Spherical-SDP Model, even if a vector $\boldsymbol{\xi}$ and a bounded set C are given, it is difficult to compute $\rho(C, \boldsymbol{\xi})$ in general.

Here, we summarize the properties of the conceptual SCRMs including Spherical-SDP, Rank-2-SDP and Rank-2-SILP Models.

Theorem 2.3.1. *([32]) Suppose that Condition 2.2.1 holds. In conceptual SCRMs,* $\{C_k \ (k = 0, 1, 2, \dots,)\}$ *and* $\{\zeta_k \ (k = 0, 1, 2, \dots,)\}$ *generated by Algorithm 2.2.4 (or Algorithm 2.2.5) satisfy the following properties:*

(a) $\text{c.hull}(F) \subseteq C_{k+1} \subseteq C_k$ *and* $\zeta^* \leq \zeta_{k+1} \leq \zeta_k \ (\forall k = 0, 1, 2, \dots,)$ *(monotonicity),*

(b) if $F = \emptyset$*, then* $C_k = \emptyset$ *for some finite number* k *(detecting-infeasibility),*

(c) if $F \neq \emptyset$*, then* $\bigcap_{k=1}^{\infty} C_k = \text{c.hull}(F)$ *and* $\zeta_k \to \zeta^*$ *as* $k \to \infty$ *(asymptotic convergence).*

Remark 2.3.2. Kojima and Tunçel [32] proved Theorem 2.3.1 for their conceptual Rank-2 Models slightly different from the conceptual models of this section. Kojima-Tunçel's Rank-2 Models adopted $\mathcal{P}_k = \mathcal{P}^2(C_k, D_1; C_k, D_2)$ instead of $\mathcal{P}_k = \mathcal{P}^2(C_0, D_1; C_k, D_2)$. Here, note that $\mathcal{P}^2(C_k, D_1; C_k, D_2)$ generates tighter relaxed region C_{k+1} than $\mathcal{P}^2(C_0, D_1; C_k, D_2)$ in Algorithms 2.2.4 and 2.2.5, but we can easily show Theorem 2.3.1 for the conceptual SCRM models with $\mathcal{P}_k = \mathcal{P}^2(C_0, D_1; C_k, D_2)$ using the proof of Theorem 2.3.1. See the proof of Theorem 3.4 in [32]. The revised models with $\mathcal{P}_k = \mathcal{P}^2(C_0, D_1; C_k, D_2)$ need less computation than Kojima-Tunçel's models in Algorithms 2.2.4 and 2.2.5, because at the kth iteration with $k > 0$, the computation of $\alpha(C_k, \boldsymbol{d})$ for $\boldsymbol{d} \in D_1$ is avoidable and the number of SDPs (or LPs) to be solved is reduced from $|D_1 \cup D_2|$ to $|D_2|$.

2.3.2 Discretized-Localized Models

To make Rank-2-SDP and Rank-2-SILP Models implementable, we need to take a finite number of vectors for the direction set $D_2 \subset \overline{D}$. Kojima and Tunçel [33] defined a *δ-net* of $\overline{D}$, a finite discretization of $\overline{D}$ which uniformly approximates $\overline{D}$, and employed it for D_2. Even with this compromise, we can maintain the monotonicity property (a) and the detecting-infeasibility property (b). But we can not expect the asymptotic convergence property (c) any more.

Let $D \subseteq \overline{D}$. For some $\delta \geq 0$, a subset D' of $\overline{D}$ is *a δ-net of D* if

$$\text{for } \forall \boldsymbol{d} \in D, \text{ there exists a } \boldsymbol{d}' \in D' \text{ such that } \|\boldsymbol{d}' - \boldsymbol{d}\| \leq \delta.$$

By definition, if $\delta > \delta' \geq 0$ and D' is a δ'-net of D, then D' is a δ-net of D. Particularly, D itself is a δ-net of D for any $\delta \geq 0$. It should be noted that we can take a finite δ-net D' of D whenever $\delta > 0$. Thus a δ-net D' of D with $\delta > 0$ leads us to a finite discretization of D.

In addition to the above *discretization technique*, *a localization technique* was introduced in [33]. The localization technique generates a sequence $\{C_k\ (k = 0, 1, 2, \dots,\ )\}$ of convex relaxations of a feasible region F, which becomes accurate only in certain directions in a neighborhood of the objective direction $\boldsymbol{c}$, and cuts off redundant work to make the convex relaxations accurate in unnecessary directions. If we are only interested in a good approximate solution and a good bound for the optimum value of QOP (1.2) but not in an approximation of the entire c.hull(F), the SCRMs with discretization and localization techniques are sufficiently effective. Therefore, we define the subset $D(\boldsymbol{c}, \kappa)$ of $\overline{D}$ as

$$D(\boldsymbol{c}, \kappa) \equiv \{\boldsymbol{d} \in \overline{D}\ :\ \|\boldsymbol{d} - \boldsymbol{c}\| \leq \kappa\}$$

for given $\kappa > 0$, and consider a finite discretization δ-net of $D(\boldsymbol{c}, \kappa)$ with some $\delta > 0$. Letting D_2 be a δ-net of $D(\boldsymbol{c}, \kappa)$, Kojima and Tunçel [33] proposed the following two models.

- Rank-2-SDP Model: $\mathcal{P}_k = \mathcal{P}^L(C_0, D_1) \cup \mathcal{P}^2(C_0, D_1; C_k, D_2)$ with $D_1 = \pm I, D_2 =$ (a δ-net of $D(\boldsymbol{c}, \kappa)$) in Algorithm 2.2.4.
- Rank-2-SILP Model: $\mathcal{P}_k = \mathcal{P}^L(C_0, D_1) \cup \mathcal{P}^2(C_0, D_1; C_k, D_2)$ with $D_1 = \pm I, D_2 =$ (a δ-net of $D(\boldsymbol{c}, \kappa)$) in Algorithm 2.2.5.

Algorithm 2.2.4 of Rank-2-SDP Model is called a discretized-localized SSDP (abbreviated by DLSSDP) relaxation method, and Algorithm 2.2.5 of Rank-2-SILP Model called a discretized-localized SSILP (abbreviated by DLSSILP) relaxation method. Each discretized-localized version generates a sequence $\{C_k\ (k = 0, 1, 2, \dots,\ )\}$ of convex subsets of C_0 and a sequence $\{\zeta_k\ (k = 0, 1, 2, \dots,\ )\}$ of real numbers. We note here that C_k can be a proper subset of $\{\boldsymbol{x} \in R^n : p(\boldsymbol{x}) \leq 0\ (\forall p(\cdot) \in \mathcal{P}_k)\}$ unless $D_2 = \bar{D}$. Nevertheless, both $\{C_k\ (k = 0, 1, 2, \dots,\ )\}$ and $\{\zeta_k\ (k = 0, 1, 2, \dots,\ )\}$ converge globally as shown in the following theorem.

Theorem 2.3.3. *([33]) Suppose that Condition 2.2.1 holds. Let $\kappa > 0$ and $\epsilon > 0$. In discretized-localized SCRMs, there exists a $\delta > 0$ such that if we take a δ-net D_2 of $D(\boldsymbol{c}, \kappa)$, then $\{C_k\ (k = 0, 1, 2, \dots\ )\}$ and $\{\zeta_k\ (k = 0, 1, 2, \dots\ )\}$ generated by Algorithm 2.2.4 (or Algorithm 2.2.5) satisfy the above properties (a), (b) and the following property (c)':*

(c)' if $F \neq \emptyset$, then $\zeta^* \leq \zeta_k < \zeta^* + \epsilon$ for some finite number k.

We call ζ_k satisfying $\zeta_k < \zeta^* + \epsilon$ an *ϵ-approximation* of the maximum objective function value of QOP (1.2). We should remark that Theorem 2.3.3 remains valid even when Condition 2.2.2 is not satisfied, *i.e.*, the quadratic representations $\mathcal{P}_F$ and $\widetilde{\mathcal{P}}_0$ are infinite. But Condition 2.2.2, together with a δ-net, enables C_k to be defined by finitely many inequality constraints. Namely, at the kth $(k = 0, 1, 2, \dots)$ iteration of Algorithm DLSSDP (or Algorithm DLSSILP), a finite number of SDPs (or LPs) with finitely many inequality constraints are generated in order to construct a function set $\mathcal{P}_k$. Therefore, Condition 2.2.2 and the device of δ-net are unavoidable for implementable Algorithms 2.2.4 and 2.2.5.

Remark 2.3.4. The original versions [33] for the discretized-localized Rank-2 Models adopt $\mathcal{P}_k = \mathcal{P}^L(C_k, D_1) \cup \mathcal{P}^2(C_k, D_1; C_k, D_2)$. Theorem 2.3.3 was proved for those original models. See the proof of Theorem 3.5 in [33]. However, from the proof, we find that the claim of Theorem 2.3.3 holds even if we take $\mathcal{P}_k = \mathcal{P}^L(C_0, D_1) \cup \mathcal{P}^2(C_0, D_1; C_k, D_2)$. When we consider the computational tasks necessary for Algorithms 2.2.4 and 2.2.5, it is better to use the revised discretized-localized Rank-2 Models.

Remark 2.3.5. In Rank-2-SDP Model, we can substitute $\mathcal{P}^1(C_0, D_1)$ for $\mathcal{P}^L(C_0, D_1)$ in the definition of $\mathcal{P}_k$, *i.e.*, $\mathcal{P}_k = \mathcal{P}^1(C_0, D_1) \cup \mathcal{P}^2(C_0, D_1; C_k, D_2)$. The set $\mathcal{P}^1(C_0, D_1)$ of convex quadratic functions makes the relaxation C_{k+1} tighter than $\mathcal{P}^L(C_0, D_1)$ does (see Remark 3.8 of [33]), though $\mathcal{P}^1(C_0, D_1)$ has half as many functions as $\mathcal{P}^L(C_0, D_1)$ does.

Remark 2.3.6. The linear functions of $\mathcal{P}^L(C_0, D_1)$ play an important role in the region C_{k+1}. Whenever $\mathcal{P}_k$ includes the set $\mathcal{P}^L(C_0, D_1)$, any feasible vector $\boldsymbol{x}$ of C_{k+1} satisfies

$$\ell sf(\boldsymbol{x}; C_k, \boldsymbol{d}) = \boldsymbol{d}^T\boldsymbol{x} - \alpha(C_k, \boldsymbol{d}) \leq 0 \quad (\forall \boldsymbol{d} \in D_2),$$

though these linear inequalities are not included in the set of constraints of C_{k+1}. Even in DLSSDP model with $\mathcal{P}_k = \mathcal{P}^1(C_0, D_1) \cup \mathcal{P}^2(C_0, D_1; C_k, D_2)$, the convex quadratic functions of $\mathcal{P}^1(C_0, D_1)$ induce the above linear constraints $\ell sf(\boldsymbol{x}; C_k, \boldsymbol{d}) \leq 0$ for $\forall \boldsymbol{d} \in D_2$.

2.4 Related Work to Successive Convex Relaxation Methods

Lovász-Schrijver [35] and Sherali-Adams [53] independently developed the lift-and-project procedure for 0-1 integer problems. We can regard the conceptual SCRMs as the extensions of the lift-and-project procedure proposed for 0-1 integer programs, to QOP (1.2). Sherali-Adams' method, often called the Reformulation-Linearization Technique (RLT), can handle not only 0-1 integer linear programs but also linearly constrained nonconvex quadratic programs [57].

2.4.1 The Lift-and-Project Procedure for 0-1 Integer Programs

In this section, we compare these three types of lift-and-project procedures:

- L. Lovász and A. Schrijver [35]
- E. Balas, S. Ceria and G. Cornuéjols [9]
- H. Sherali and W. Adams [53]

and introduce the theorem shown by Kojima and Tunçel [33], which relates conceptual SCRMs to the Lovász and Schrijver construction.

Here we consider a mixed 0-1 integer program:

$$\max \ \boldsymbol{c}^T \boldsymbol{x} \quad \text{subject to} \quad \boldsymbol{x} \in F, \tag{2.11}$$

where

$$\begin{array}{rcl} F & = & \{\boldsymbol{x} \in C_0 : \ x_j \in \{0,1\}, \ j = 1, \ldots, p\}, \\ C_0 & \equiv & \{\boldsymbol{x} \in R^n : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \ 0 \leq x_j \leq 1, \ j = 1, \ldots, p\} \\ & \equiv & \{\boldsymbol{x} \in R^n : \widetilde{\boldsymbol{A}}\boldsymbol{x} \leq \widetilde{\boldsymbol{b}}\}. \end{array}$$

F is the feasible set of a mixed integer programming problem with n variables, p of which are 0-1 integer constrained. C_0 is the standard LP relaxation. Let us combine bound-factors and constraint-factors in a single set, as the last expression of C_0 shows. Suppose that $\widetilde{\boldsymbol{A}}$ is an $m \times n$ matrix. We denote the ith constraint of $\widetilde{\boldsymbol{A}}\boldsymbol{x} \leq \widetilde{\boldsymbol{b}}$ by $\widetilde{A}_i \boldsymbol{x} \leq \widetilde{b}_i$ $(i = 1, \ldots, m)$. In this section, we consider the procedures for finding the convex hull of F (denoted by c.hull(F)).

The Lovász-Schrijver construction

One lift-and-project procedure was proposed by Lovász and Schrijver [35] for 0-1 integer programs (2.11). The procedure consists of following two steps.

Lifting : Multiplying every inequality by every 0-1 integer variable and its complement in turn, and then linearizing the resulting system of quadratic inequalities, we obtain a linear-constraint system of higher dimensional variables.

Projecting : We project back the system onto the original space.

Repeating the lift-and-project steps p times (the number of original 0-1 variables) yields the convex hull of the feasible region F, while each lifting step involves a squaring of the number of variables and an even steeper increase in the number of constraints. The lift-and-project procedure of the Lovász-Schrijver construction is as follows.

Step 1. Multiply $\widetilde{\boldsymbol{A}}\boldsymbol{x} \leq \widetilde{\boldsymbol{b}}$ with x_i and $1 - x_i$ $(i = 1, \ldots, p)$ to obtain the set of nonlinear quadratic functions

$$\mathcal{Q}(C_0) \equiv \left\{ \begin{array}{c} (1 - x_i)(\widetilde{A}_j \boldsymbol{x} - \widetilde{b}_j), \ x_i(\widetilde{A}_j \boldsymbol{x} - \widetilde{b}_j) \\ i = 1, \ldots, p, \ j = 1, \ldots, m \end{array} \right\}. \tag{2.12}$$

Step 2. Linearize (2.12) by substituting X_{ij} for $x_i x_j$, and setting $X_{ij} = X_{ji}$ ($i = 1, \ldots, n$, $j = 1, \ldots, p$). Also, set $x_j = X_{jj}$ ($j = 1, \ldots, p$), since $x_j = x_j^2$ holds for the both cases of $x_j = 0$ and $x_j = 1$. Then, let $M(C_0)$ be the polyhedron consisting of such linearized functions, *i.e.*,

$$M(C_0) = \left\{ \begin{pmatrix} 1 & \boldsymbol{x}^T \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} \in \mathcal{S}^{1+n} : \begin{array}{l} \gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{Q} \bullet \boldsymbol{X} \leq 0 \ \ (\forall qf(\cdot;\gamma,\boldsymbol{q},\boldsymbol{Q}) \in \mathcal{Q}(C_0)) \\ X_{jj} = x_j, \ \ (j = 1, \ldots, p) \end{array} \right\}.$$

Step 3. Project $M(C_0)$ onto the $\boldsymbol{x}$-space. Let $N(C_0) \subset R^n$ denote the projected polyhedron *i.e.*,

$$N(C_0) = \left\{ \boldsymbol{x} \in R^n : \begin{array}{c} \exists \boldsymbol{X} \in \mathcal{S}^n \text{ such that} \\ \begin{pmatrix} 1 & \boldsymbol{x}^T \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} \in M(C_0) \end{array} \right\}.$$

Let $N^0(C_0) = C_0$ and let $N^t(C_0) = N(N^{t-1}(C_0))$ for $t \geq 1$. Lovász-Schrijver has shown that $N(C_0)$ and $N^p(C_0)$ have the following properties:

Theorem 2.4.1. *([35]) $N(C_0) \subseteq c.hull(C_0 \cap \{\boldsymbol{x} \in R^n : \ x_j \in \{0,1\}\})$ for $j = 1, \ldots, p$.*

Theorem 2.4.2. *([35]) $N^p(C_0) = c.hull(F)$.*

Theorem 2.4.2 means that iterating the above procedure p times yields the integer hull. The paper [35] also proposed a tighter relaxation (denoted by $M_+(C_0)$) than $M(C_0)$ by adding one condition such that

$$\begin{pmatrix} 1 & \boldsymbol{x}^T \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} \in \mathcal{S}^{1+n}_+ \qquad \text{(positive semidefinite)}$$

to $M(C_0)$. Also, $N_+(C_0) \subseteq R^n$ denotes the convex set projected from $M_+(C_0) \subseteq \mathcal{S}^{1+n}_+$ onto the $\boldsymbol{x}$-space. Theorems 2.4.1 and 2.4.2 hold not only for $N(C_0)$ but for $N_+(C_0)$.

In the papers [32, 33], Kojima and Tunçel connected Algorithm 2.2.4 (the SSDP relaxation method) and Algorithm 2.2.5 (the SSILP relaxation method) to the Lovász-Schrijver's lift-and-project procedure. Now let us see how to apply Algorithms 2.2.4 and 2.2.5 to a 0-1 integer program (2.11). Let $\mathcal{P}_F$ denote the set of quadratic functions:

$$x_i(x_i - 1) \ \text{ and } \ -x_i(x_i - 1) \ \ (\forall i = 1, 2, \ldots, p). \tag{2.13}$$

Then we can rewrite the 0-1 integer program under consideration as

$$\left. \begin{array}{ll} \max & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} & \boldsymbol{x} \in F \equiv \{\boldsymbol{x} \in C_0 : p(\boldsymbol{x}) \leq 0 \ (\forall p(\cdot) \in \mathcal{P}_F)\}. \end{array} \right\} \tag{2.14}$$

Theorem 2.4.3 below shows that the conceptual SCRMs, introduced in Section 2.3.1, work similar to N_+ and N procedures.

Theorem 2.4.3. *([33]) Apply the conceptual SCRMs with* $D_1 = \pm I$, $D_2 = \overline{D}$ *and* $\mathcal{P}_k = \mathcal{P}^2(C_k, D_1, D_2)$ *to the* 0-1 *integer program (2.14). Then,*

$$\begin{aligned} N_+^k(C_0) &= \widehat{F}(C_0, \mathcal{P}_F \cup \mathcal{P}_k) \text{ when we use Algorithm 2.2.4,} \\ N^k(C_0) &= \widehat{F}^L(C_0, \mathcal{P}_F \cup \mathcal{P}_k) \text{ when we use Algorithm 2.2.5,} \end{aligned}$$

$(k = 0, 1, 2, \dots)$.

The Balas, Ceria and Cornuéjols construction

Balas-Ceria-Cornuéjols [9] proposed another lift-and-project procedure where the original constraints $\widetilde{\boldsymbol{A}}\boldsymbol{x} \le \widetilde{\boldsymbol{b}}$ are multiplied by a single 0-1 variable and its complement before projecting back onto the original space.

Step 0. Select an index $j \in \{1, \dots, p\}$.

Step 1. Multiply $\widetilde{A}_i\boldsymbol{x} \le \widetilde{b}_i$ $(i = 1, \dots, m)$ with $1 - x_j$ and x_j to obtain the nonlinear system;

$$(1 - x_j)(\widetilde{A}_i\boldsymbol{x} - \widetilde{b}_i) \le 0, \;\; x_j(\widetilde{A}_i\boldsymbol{x} - \widetilde{b}_i) \le 0, \;\; i = 1, \dots, m. \tag{2.15}$$

Step 2. Linearize (2.15) by substituting y_i for x_ix_j $(i = 1, \dots, n, \; i \ne j)$ and x_j for x_j^2. Then, construct a polyhedron from such linearized constraints with variables $\boldsymbol{x} \in R^n$ and $\boldsymbol{y} \in R^{n-1}$. We designate the polyhedron as $M_j(K)$.

Step 3. Project $M_j(K)$ onto the $\boldsymbol{x}$-space. Let $P_j(K)$ denote the resulting polyhedron.

If the region C_0 of (2.11) has m constraints and n variables, the polyhedron $M_j(K)$ consists of $2m$ constraints and $2n - 1$ variables. For $t \ge 2$, define

$$P_{i_1,\dots,i_t}(C_0) = P_{i_t}(P_{i_{t-1}} \dots (P_{i_1}(C_0)) \dots).$$

Balas, Ceria and Cornuéjols have shown the following two theorems. Note that Theorem 2.4.4 implies the claim of Theorem 2.4.1, since $N(C_0) \subseteq P_j(C_0)$ for $j \in \{1, \dots, p\}$.

Theorem 2.4.4. *([9])* $P_j(C_0) = c.hull(C_0 \cap \{\boldsymbol{x} \in R^n : x_j \in \{0, 1\}\})$ *for* $j = 1, \dots, p$.

Corollary 2.4.5. *([9])* $P_{i_1,\dots,i_t}(C_0) = c.hull(C_0 \cap \{\boldsymbol{x} \in R^n : x_j \in \{0, 1\}, \; j = i_1, \dots, i_t\})$.

Theorem 2.4.6. *([9])* $P_{1,\dots,p}(C_0) = c.hull(F)$.

The Sherali and Adams construction

Somewhat earlier than Lovász-Schrijver, Sherali-Adams [53] has proposed a similar procedure, which obtains the integer hull of the feasible region F in a noniterative fashion as follows:

Step 0. Let $t \in \{1, \ldots, p\}$.

Step 1. Multiply $\widetilde{\boldsymbol{A}}\boldsymbol{x} \leq \widetilde{\boldsymbol{b}}$ with every product of the form $\{\Pi_{j \in J_1} x_j\}\{\Pi_{j \in J_2}(1-x_j)\}$, where J_1 and J_2 are disjoint subsets of $\{1, \ldots, p\}$ such that $|J_1 \cup J_2| = t$. We designate the resulting nonlinear system as (NL_t).

Step 2. Linearize (NL_t) by (i) substituting x_j for x_j^2; and (ii) substituting a variable w_J for every product $\Pi_{j \in J} x_j$, where $J \subseteq \{1, \ldots, p\}$, and v_{Jk} for every product $x_k \Pi_{j \in J} x_j$ where $J \subseteq \{1, \ldots, p\}$ and $k \subseteq \{p+1, \ldots, n\}$. Let X_t be the polyhedron defined by the resulting linear functions.

Step 3. Project X_t onto the $\boldsymbol{x}$-space. Let C_t denote the resulting polyhedron.

It is easy to see that $F \subseteq C_p \subseteq \ldots \subseteq C_1 \subseteq C_0$. In addition, Sherali and Adams proved the following theorem.

Theorem 2.4.7. *([53]) $C_p = c.hull(F)$.*

Balas, Ceria and Cornuéjols [9] have shown that Theorem 2.4.7 follows from Corollary 2.4.5. The theorem also follows from Theorem 2.4.2, together with the proposition shown in [35] such that $C_t \subseteq N^t(C_0)$ for $t = 1, 2, \ldots, p$.

2.4.2 The Reformulation-Linearization Technique for QOPs

The Sherali-Adams' method, Reformulation-Linearization Technique (RLT), covers not only 0-1 integer linear programs but also 0-1 mixed-integer polynomial programs [54]. The extension of the RLT to continuous polynomial programs, which include the nonconvex quadratic programs as a special case, was first developed by Sherali and Tuncbilek [56]. Recently, Sherali and Tuncbilek [57] investigated a special RLT for a linearly constrained nonconvex quadratic program;

$$\max \ \beta + 2\,\boldsymbol{p}^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} \quad \text{subject to} \quad \boldsymbol{x} \in F, \tag{2.16}$$

where

$$\begin{aligned} F \ &\equiv \ \left\{ \boldsymbol{x} \in R^n : \begin{array}{l} \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ 0 \leq l_k \leq x_k \leq u_k < \infty \ \ k = 1, \ldots, n \end{array} \right\} \\ &\equiv \ \left\{ \boldsymbol{x} \in R^n : \widetilde{\boldsymbol{A}}\boldsymbol{x} \leq \widetilde{\boldsymbol{b}} \right\}. \end{aligned} \tag{2.17}$$

For brevity of presentation, let us combine bound-factors such that $l_k \leq x_k \leq u_k$ $(k = 1, \ldots, n)$ and constraint-factors such that $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$, as the last expression of (2.17) shows. Supposing that $\widetilde{\boldsymbol{A}}$ is an $m \times n$ matrix, we denote the ith constraint of (2.17) as $\widetilde{A}_i\boldsymbol{x} \leq \widetilde{b}_i$ $(i = 1, \ldots, m)$.

The procedure of the RLT consists of two steps in general terms; the reformulation step and the linearization step. At the reformulation step, we take all possible pairwise products among $\widetilde{A}_i\boldsymbol{x} \leq \widetilde{b}_i$ $(i = 1, \ldots, m)$, including self-products, and generate the following nonlinear implied constraints:

$$-(\widetilde{A}_i\boldsymbol{x} - \widetilde{b}_i)(\widetilde{A}_j\boldsymbol{x} - \widetilde{b}_j) \leq 0, \;\; 1 \leq \forall i \leq \forall j \leq m.$$

Here, we define the set of quadratic functions as $\mathcal{P} \equiv \{-(\widetilde{A}_i\boldsymbol{x} - \widetilde{b}_i)(\widetilde{A}_j\boldsymbol{x} - \widetilde{b}_j), \;\; 1 \leq \forall i \leq \forall j \leq m\}$, and then linearize quadratic functions of $\mathcal{P}$ by substituting

$$X_{ij} = x_i x_j, \;\; 1 \leq \forall i \leq \forall j \leq n.$$

Using the notation introduced in Section 2.2, we obtain the following upper bounding linear program:

$$\text{(RLT-LP)} \quad \left| \begin{array}{ll} \max & \beta + 2\,\boldsymbol{p}^T\boldsymbol{x} + \boldsymbol{P} \bullet \boldsymbol{X} \\ \text{subject to} & \gamma + 2\,\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{Q} \bullet \boldsymbol{X} \leq 0 \quad (\forall qf(\cdot;\gamma,\boldsymbol{q},\boldsymbol{Q}) \in \mathcal{P}). \end{array} \right. \tag{2.18}$$

The set $\mathcal{P}$ has the pairwise products among a lower-bound constraint $(\ell_i \leq x_i)$ and an upper-bound constraint $(x_i \leq u_i)$ of a variable x_i, so that (RLT-LP) includes the following linear constraints.

$$\left. \begin{array}{l} X_{ii} \leq (u_i + l_i)x_i - u_i l_i, \\ 2l_i\, x_i - l_i^2 \leq X_{ii}, \;\; 2u_i\, x_i - u_i^2 \leq X_{ii}. \end{array} \right\} \tag{2.19}$$

These inequalities approximate the relationship $X_{ii} = x_i^2$ over the interval $l_i \leq x_i \leq u_i$.

As Reformulation-Convexification Approach (RCA), the paper [57] proposed another technique which replaces (2.19) by the nonlinear constraints

$$x_i^2 \leq X_{ii} \leq (u_i + l_i)x_i - u_i l_i. \tag{2.20}$$

Notice that the upper bounding linear function of (2.20) is coincident with the first constraint of (2.19). The last two constraints of (2.19) merely approximate the function $X_{ii} = x_i^2$ from below via tangential supports at the points l_i and u_i. On the other hand, since (2.20) produces the exact lower envelope, it is equivalent to having an additional tangential support at the optimum point. Therefore, the upper bounding problem which includes nonlinear constraints (2.20) instead of linear constraints (2.19) generates a tighter approximate solution.

If we apply the SDP relaxation instead of the LP relaxation after the reformulation step of the RLT, we obtain the following semidefinite program:

$$\text{(RLT-SDP)} \quad \left| \begin{array}{ll} \max & \beta + 2\,\boldsymbol{p}^T\boldsymbol{x} + \boldsymbol{P} \bullet \boldsymbol{X} \\ \text{subject to} & \gamma + 2\,\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{Q} \bullet \boldsymbol{X} \leq 0, \quad (\forall qf(\cdot;\gamma,\boldsymbol{q},\boldsymbol{Q}) \in \mathcal{P}), \\ & \begin{pmatrix} 1 & \boldsymbol{x}^T \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} \in \mathcal{S}_+^{1+n}. \end{array} \right. \tag{2.21}$$

The positive semidefinite condition of SDP (2.21) indicates that $x_i^2 \leq X_{ii}$ for $\forall i$. Therefore, (2.21) can provide tighter upper bounds for the problem (2.16) than the RCA.

Here, consider the SILP relaxation $\widehat{F}^L(C_0, \mathcal{P}_F \cup \mathcal{P}_0)$ with $D_1 = D_2 = \pm I$ and $\mathcal{P}_0 = \mathcal{P}^L(C_0, D_1) \cup \mathcal{P}^2(C_0, D_1, D_2)$. Then all constraints of $\widehat{F}^L(C_0, \mathcal{P}_F \cup \mathcal{P}_0)$ are included in RLT-LP (2.18). Similarly, all constraints of the SDP relaxation $\widehat{F}(C_0, \mathcal{P}_F \cup \mathcal{P}_0)$ with the above D_1, D_2 and $\mathcal{P}_0$ are included in RLT-SDP (2.21). Thus, RLT-LP and RLT-SDP generate tighter upper bounds for a optimum objective function value of (2.16) than one applications of the SILP relaxation and the SDP relaxation, respectively. Also, RLT-LP and RLT-SDP include more constraints than $\widehat{F}^L(C_0, \mathcal{P}_F \cup \mathcal{P}_0)$ and $\widehat{F}(C_0, \mathcal{P}_F \cup \mathcal{P}_0)$, respectively. Indeed, the RLT generates quadratic constraints by not only (bound-factor $\times$ bound-factor), but (bound-factor $\times$ constraint-factor) and (constraint-factor $\times$ constraint-factor).

Chapter 3

Complexity Analysis for Conceptual Relaxation Methods

This chapter investigates computational complexity of the conceptual SCRMs (successive convex relaxation methods) given in Section 2.3.1. Theorem 2.4.3 indicates that when applied to 0-1 integer programs, the conceptual SCRMs terminate in p iterations as the lift-and-project procedure [35] does. Here p denotes the number of 0-1 variables. Therefore, we can regard the conceptual SCRMs as the extensions of the lift-and-project procedure proposed for 0-1 integer programs, to QOPs (1.2). In general cases where $\mathcal{P}_F$ consists of arbitrary quadratic functions, the convergence of $\{C_k \ (k = 0, 1, 2, \ldots)\}$ to c.hull(F) is slower than linear in the worst case. (See an example in Section 8.3 of [32]). In the current chapter, we bound the number k of iterations which Algorithm 2.2.4 or 2.2.5 requires to generate an approximation of c.hull(F) with a given "accuracy." See Section 2.2 for the description of Algorithms 2.2.4 and 2.2.5.

We focus our attention on the three models of conceptual SCRMs presented in Section 2.3.1; Spherical-SDP Model, Rank-2-SDP Model and Rank-2-SILP Model. At each iteration, we need to prepare a set $\mathcal{P}_k$ of infinitely many quadratic functions before performing the SDP (semidefinite programming) or SILP (semi-infinite LP) relaxation. Here we assume that such a $\mathcal{P}_k$ is available.

Our complexity analysis of the Spherical-SDP Model is much simpler than that of Rank-2-SDP Model or Rank-2-SILP Model, and the former analysis helps an easier understanding of the latter two models. Therefore, in Section 3.2, we consider the Spherical-SDP Model by taking a set of spherical functions for $\mathcal{P}_k$. And then, in Section 3.3, we deal with the Rank-2-SDP and the Rank-2-SILP Models whose $\mathcal{P}_k$ consists of rank-2 quadratic functions. Here note that the Rank-2-SDP Model generates a tighter relaxed region C_{k+1} from the function set $\mathcal{P}_k$ than the Rank-2-SILP Model does. Indeed, the definitions of $\widehat{F}(C_0, \mathcal{P}_F \cup \mathcal{P}_k)$ and $\widehat{F}^L(C_0, \mathcal{P}_F \cup \mathcal{P}_k)$ in (2.9) lead to the following inclusion relation; $\widehat{F}(C_0, \mathcal{P}_F \cup \mathcal{P}_k) \subseteq \widehat{F}^L(C_0, \mathcal{P}_F \cup \mathcal{P}_k)$. Therefore, our complexity analysis on the Rank-2-SILP Model can be applied simultaneously to the Rank-2-SDP Model. Thus, we mainly focus on the Rank-2-SILP Model in Section 3.3.

3.1 Accuracy Measures of Convex Relaxation

To begin with, we need to clarify the following issues.

- Input data of the conceptual SCRMs.
- Output of the methods and its quality or "accuracy."

Our input data are a nonempty compact convex subset C_0 of R^n and a set $\mathcal{P}_F$ of finitely or infinitely many quadratic functions. We do not care about how we represent the compact convex set C_0; it may be represented in terms of finitely or infinitely many linear inequalities, nonlinear convex inequalities and/or linear matrix inequalities. Although it seems nonsense to try to define the size of such input data, we extract some quantity and quality from the input data. Concerning quality or difficulty of the input data, we introduce

$$\left.\begin{array}{rcl} \mathrm{diam}(G) & = & \sup\{\|\boldsymbol{x}-\boldsymbol{y}\| : \boldsymbol{x},\ \boldsymbol{y} \in G\} \ \text{ for } \forall G \subset R^n \\ & & \text{(the diameter of } G \subset R^n \text{),} \\ \nu_{\mathrm{lip}} & = & \nu_{\mathrm{lip}}(\mathcal{P}_F, C_0) = \sup\left\{ \dfrac{|p(\boldsymbol{x})-p(\boldsymbol{y})|}{\|\boldsymbol{x}-\boldsymbol{y}\|} : \boldsymbol{x},\ \boldsymbol{y} \in C_0, \boldsymbol{x} \neq \boldsymbol{y},\ p(\cdot) \in \mathcal{P}_F \right\} \\ & & \text{(a common Lipschitz constant for all } p(\cdot) \in \mathcal{P}_F \text{ on } C_0\text{),} \\ \nu_{\mathrm{nc}} & = & \nu_{\mathrm{nc}}(\mathcal{P}_F) = \sup\left\{ -\inf\{\lambda_{\min}(\boldsymbol{Q}) : qf(\cdot;\gamma,\boldsymbol{q},\boldsymbol{Q}) \in \mathcal{P}_F\},\ 0 \right\}, \\ \nu_{\mathrm{nl}} & = & \nu_{\mathrm{nl}}(\mathcal{P}_F) = \sup\{\sum_{i=1}^n \sum_{j=1}^n |Q_{ij}| : qf(\cdot;\gamma,\boldsymbol{q},\boldsymbol{Q}) \in \mathcal{P}_F\}. \end{array}\right\} \quad (3.1)$$

Here $\lambda_{\min}(\boldsymbol{Q})$ denotes the minimum eigenvalue of $\boldsymbol{Q} \in \mathcal{S}^n$. Note that all $\nu_{\mathrm{lip}}(\mathcal{P}_F, C_0)$, $\nu_{\mathrm{nc}}(\mathcal{P}_F)$ and $\nu_{\mathrm{nl}}(\mathcal{P}_F)$ take either a finite nonnegative value or $+\infty$. We will assume that they are finite. If $\mathcal{P}_F$ consists of a finite number of quadratic functions, this assumption is satisfied. We may regard $\nu_{\mathrm{nc}}(\mathcal{P}_F)$ as a nonconvexity measure of the set $\mathcal{P}_F$ of quadratic functions; all quadratic functions in $\mathcal{P}_F$ are convex if and only if $\nu_{\mathrm{nc}}(\mathcal{P}_F) = 0$, and $\mathcal{P}_F$ involves more nonconvexity as $\nu_{\mathrm{nc}}(\mathcal{P}_F) \geq 0$ gets larger. On the other hand, we may regard $\nu_{\mathrm{nl}}(\mathcal{P}_F)$ as a nonlinearity measure of the set $\mathcal{P}_F$ of quadratic functions; all functions in $\mathcal{P}_F$ are linear if and only if $\nu_{\mathrm{nl}}(\mathcal{P}_F) = 0$, and $\mathcal{P}_F$ involves more nonlinearity as $\nu_{\mathrm{nl}}(\mathcal{P}_F) \geq 0$ gets larger. $\nu_{\mathrm{lip}}(\mathcal{P}_F, C_0)$, $\nu_{\mathrm{nc}}(\mathcal{P}_F)$ and $\nu_{\mathrm{nl}}(\mathcal{P}_F)$ directly affect the upper bounds that we will derive for the number k of iterations required to generate an approximation of c.hull(F) with a given certain accuracy. Also, the diameter diam(C_0) of C_0 and the diameter diam(F) of F are relevant on our complexity analysis since C_0 serves as an initial approximation of c.hull(F) which we want to compute.

Our output is a compact set $C_k \subset R^n$, which is an approximation of c.hull(F). Again we don't care about its representation and size. In order to evaluate the quality or accuracy of the approximation, we introduce the notation

$$F(\epsilon) = F(\epsilon, C_0, \mathcal{P}_F) = \{\boldsymbol{x} \in C_0 : p(\boldsymbol{x}) \leq \epsilon \ (\forall p(\cdot) \in \mathcal{P}_F)\} \quad \text{for } \forall \epsilon \geq 0. \qquad (3.2)$$

By definition, $F = F(0) \subset F(\epsilon_1) \subset F(\epsilon_2) \subset C_0$ whenever $0 \leq \epsilon_1 \leq \epsilon_2$. We say that a compact convex subset C of C_0 is *an ϵ-convex-relaxation* of F if it satisfies

$$F \subset C \subset \mathrm{c.hull}(F(\epsilon)).$$

We set up an ϵ-convex-relaxation of F as our goal of the SCRMs. An ϵ-convex-relaxation of F, however, is not easy to manipulate directly in our complexity analysis of Algorithms 2.2.4 and 2.2.5. Here, we introduce another notion (ψ, Ξ)-convex-relaxation of F which is easier to manipulate, and relate it to the ϵ-convex-relaxation of F.

Let $\psi > 0$, let $\Xi \subset R^n$ be a nonempty compact convex set, and define the n-dimensional closed ball with a center $\boldsymbol{\xi} \in R^n$ and a radius $\rho > 0$ as $B(\boldsymbol{\xi}, \rho) = \{\boldsymbol{x} \in R^n : \|\boldsymbol{x} - \boldsymbol{\xi}\| \leq \rho\}$. We say that a compact convex subset C of C_0 is *an* (ψ, Ξ)*-convex-relaxation* of $F(C_0, \mathcal{P}_F)$ if

$$F \subset C \subset \text{c.relax}(F(\psi), \Xi) \;\; = \bigcap_{\boldsymbol{\xi} \in \Xi} B(\boldsymbol{\xi}, \rho(F(\psi), \boldsymbol{\xi})).$$

The definition of c.relax$(F(\psi), \Xi)$ is quite similar to that of (ψ, ρ)-approximation of c.hull(F) given in Section 5 of [33]. Note that $B(\boldsymbol{\xi}, \rho(F(\psi), \boldsymbol{\xi}))$ in the definition of c.relax$(F(\psi), \Xi)$ corresponds to the minimum ball with the center $\boldsymbol{\xi}$ that contains $F(\psi)$. It is easily verified that given an arbitrary open convex set U containing F, if $\psi > 0$ is sufficiently small and if Ξ is a sufficiently large ball with its center in C_0, then $F \subset \text{c.relax}(F(\psi), \Xi) \subset U$. See Lemma 5.1 of [33] and its proof. By definition, we also see that

$$\text{c.relax}(F(\psi), \Xi_1) \;\; \supset \;\; \text{c.relax}(F(\psi), \Xi_2) \;\; \text{if } \Xi_1 \subset \Xi_2.$$

We assume in the remainder of this section that there exists a finite common Lipschitz constant for all $p(\cdot) \in \mathcal{P}_F$;

$$\nu_{\text{lip}} \;\; = \;\; \nu_{\text{lip}}(\mathcal{P}_F, C_0) = \sup \left\{ \frac{|p(\boldsymbol{x}) - p(\boldsymbol{y})|}{\|\boldsymbol{x} - \boldsymbol{y}\|} : \boldsymbol{x},\; \boldsymbol{y} \in C_0, \boldsymbol{x} \neq \boldsymbol{y},\; p(\cdot) \in \mathcal{P}_F \right\} < \infty.$$

If $\mathcal{P}_F$ consists of a finite number of quadratic functions, then this assumption is satisfied.

Lemma 3.1.1. *Let $\boldsymbol{\xi}_0 \in C_0$ and $\epsilon > 0$. Choose positive numbers ψ, τ and a compact convex set Ξ such that*

$$\psi = \frac{\epsilon}{2}, \; \tau \geq \frac{\text{diam}(C_0)^2}{2\delta} + \frac{\delta}{2} \;\; \textit{and } \Xi = B(\boldsymbol{\xi}_0, 3\tau/2), \tag{3.3}$$

where

$$0 \;\; < \;\; \delta \leq \min\left\{ \frac{\epsilon}{2\,\nu_{\text{lip}}}, \; \frac{\text{diam}(C_0)}{4} \right\}. \tag{3.4}$$

Then c.relax$(F(\psi), \Xi) \subset$ c.hull$(F(\epsilon))$.

Proof: If $F(\psi)$ is empty, then the desired inclusion relation holds with $\emptyset \subset$ c.hull$(F(\epsilon))$. So we will deal with the case that $F(\psi)$ is not empty.

(i) Define a δ-neighborhood G of $F(\psi)$ within C_0 such that

$$\begin{aligned} G &= \{\boldsymbol{x} \in C_0 : B(\boldsymbol{x},\delta) \cap F(\psi) \neq \emptyset\} \\ &= \{\boldsymbol{x} \in C_0 : \|\boldsymbol{x}-\boldsymbol{y}\| \leq \delta \ \text{ for } \exists \boldsymbol{y} \in F(\psi)\}. \end{aligned}$$

We show that c.hull(G) $\subset$ c.hull($F(\epsilon)$). Suppose that $\boldsymbol{x} \in G$. Then $\boldsymbol{x} \in C_0$ and there exists a $\boldsymbol{y} \in F(\psi)$ such that $\|\boldsymbol{x}-\boldsymbol{y}\| \leq \delta$; hence

$$\begin{aligned} p(\boldsymbol{x}) &\leq p(\boldsymbol{y}) + |p(\boldsymbol{x}) - p(\boldsymbol{y})| \\ &\leq \psi + \nu_{\text{lip}}\, \|\boldsymbol{x}-\boldsymbol{y}\| \\ &\quad\ (\text{by } \boldsymbol{y} \in F(\psi) \text{ and the definition of } \nu_{\text{lip}}) \\ &\leq \epsilon/2 + \nu_{\text{lip}}\,\delta \ \ (\text{since } \psi = \epsilon/2 \text{ and } \|\boldsymbol{x}-\boldsymbol{y}\| \leq \delta) \\ &\leq \epsilon \ \ (\text{by } (3.4)). \end{aligned}$$

Thus we have shown that $G \subset F(\epsilon)$, which implies that c.hull(G) $\subset$ c.hull($F(\epsilon)$).

(ii) In view of (i), it suffices to prove that

$$\text{c.relax}(F(\psi), \Xi) \subset \text{c.hull}(G).$$

Assuming that $\bar{\boldsymbol{x}} \notin$ c.hull(G), we show that $\bar{\boldsymbol{x}} \notin$ c.relax($F(\psi), \Xi$). We will construct a ball $B(\bar{\boldsymbol{\xi}}, \tau) \subset R^n$ such that

$$\bar{\boldsymbol{\xi}} \in \Xi, \ \ \bar{\boldsymbol{x}} \notin B(\bar{\boldsymbol{\xi}}, \tau) \ \text{ and } F(\psi) \subset B(\bar{\boldsymbol{\xi}}, \tau). \tag{3.5}$$

Then $\bar{\boldsymbol{x}} \notin$ c.relax($F(\psi), \Xi$) follows from the definition of c.relax($F(\psi), \Xi$). Let $\bar{\boldsymbol{y}} \in$ c.hull($F(\psi)$) $\subset C_0$ be the point that minimize the distance $\|\bar{\boldsymbol{x}}-\boldsymbol{y}\|$ over $\boldsymbol{y} \in$ c.hull($F(\psi)$). Let $\bar{\delta} = \|\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}\|$ and $\bar{\boldsymbol{d}} = (\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}})/\bar{\delta}$. Then $\{\boldsymbol{y} \in R^n : \bar{\boldsymbol{d}}^T\boldsymbol{y} = \bar{\boldsymbol{d}}^T\bar{\boldsymbol{y}}\}$ forms a supporting hyperplane of c.hull($F(\psi)$) such that

$$\bar{\boldsymbol{d}}^T\boldsymbol{y} \leq \bar{\boldsymbol{d}}^T\bar{\boldsymbol{y}} \ \text{ for } \forall \boldsymbol{y} \in \text{c.hull}(F(\psi)). \tag{3.6}$$

We will show that $\bar{\delta} > \delta$. Assume on the contrary that $\bar{\delta} \leq \delta$. Since $\bar{\boldsymbol{y}}$ lies in c.hull($F(\psi)$), there are $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k \in F(\psi)$ $(1 \leq \exists k \leq n+1)$ such that

$$\bar{\boldsymbol{y}} \in \text{c.hull}\left(\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k\}\right).$$

Let

$$\boldsymbol{x}_j = \boldsymbol{y}_j + \bar{\delta}\bar{\boldsymbol{d}} \ (j = 1, 2, \ldots, k).$$

Then we see that

$$\begin{aligned} \|\boldsymbol{x}_j - \boldsymbol{y}_j\| &= \|\bar{\delta}\bar{\boldsymbol{d}}\| \leq \delta \ \text{ and } \boldsymbol{y}_j \in F(\psi) \text{ (hence } \boldsymbol{x}_j \in G) \text{ for } \forall j = 1, 2, \ldots, k, \\ \bar{\boldsymbol{x}} &= \bar{\boldsymbol{y}} + \bar{\delta}\bar{\boldsymbol{d}} \\ &\in \text{c.hull}\left(\{\boldsymbol{y}_1 + \bar{\delta}\bar{\boldsymbol{d}}, \boldsymbol{y}_2 + \bar{\delta}\bar{\boldsymbol{d}}, \ldots, \boldsymbol{y}_k + \bar{\delta}\bar{\boldsymbol{d}}\}\right) \\ &= \text{c.hull}\left(\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k\}\right). \end{aligned}$$

This implies that $\bar{\boldsymbol{x}} \in \text{c.hull}(G)$, which contradicts to the hypothesis that $\bar{\boldsymbol{x}} \notin \text{c.hull}(G)$. Thus we have shown that $\bar{\delta} > \delta$.

(iii) Now defining $\bar{\boldsymbol{\xi}} = \bar{\boldsymbol{x}} - (\tau + \bar{\delta} - \delta)\bar{\boldsymbol{d}}$, we show that $\bar{\boldsymbol{\xi}}$ and the ball $B(\bar{\boldsymbol{\xi}}, \tau)$ satisfies (3.5). From the definition of $\bar{\boldsymbol{\xi}}$ above and $\bar{\delta} > \delta$, we first observe that

$$\begin{array}{rcl} \bar{\boldsymbol{x}} & \notin & B(\bar{\boldsymbol{\xi}}, \tau) \ \ \text{(i.e., the 2nd relation of (3.5))}, \\ \bar{\boldsymbol{y}} & = & \bar{\boldsymbol{x}} - \bar{\delta}\bar{\boldsymbol{d}} = \bar{\boldsymbol{\xi}} + (\tau + \bar{\delta} - \delta)\bar{\boldsymbol{d}} - \bar{\delta}\bar{\boldsymbol{d}} = \bar{\boldsymbol{\xi}} + (\tau - \delta)\bar{\boldsymbol{d}}. \end{array} \tag{3.7}$$

Next we will show that the third relation of (3.5), *i.e.*, $\text{c.hull}(F(\psi)) \subset B(\bar{\boldsymbol{\xi}}, \tau)$. Suppose that $\boldsymbol{y} \in \text{c.hull}(F(\psi))$. Then

$$\begin{array}{rcl} \boldsymbol{y} - \bar{\boldsymbol{\xi}} & = & (\bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T)(\boldsymbol{y} - \bar{\boldsymbol{\xi}}) + \left(\boldsymbol{I} - \bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T\right)(\boldsymbol{y} - \bar{\boldsymbol{\xi}}), \\ \boldsymbol{y} - \bar{\boldsymbol{y}} & = & (\bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T)(\boldsymbol{y} - \bar{\boldsymbol{y}}) + \left(\boldsymbol{I} - \bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T\right)(\boldsymbol{y} - \bar{\boldsymbol{y}}) \\ & = & (\bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T)(\boldsymbol{y} - \bar{\boldsymbol{y}}) + \left(\boldsymbol{I} - \bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T\right)(\boldsymbol{y} - \bar{\boldsymbol{\xi}} - (\tau - \delta)\bar{\boldsymbol{d}}) \ \ \text{(by (3.7))} \\ & = & (\bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T)(\boldsymbol{y} - \bar{\boldsymbol{y}}) + \left(\boldsymbol{I} - \bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T\right)(\boldsymbol{y} - \bar{\boldsymbol{\xi}}). \end{array}$$

Note that $\bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T$ and $\left(\boldsymbol{I} - \bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T\right)$ are orthogonal projection matrices onto the one dimensional line $\{\lambda\bar{\boldsymbol{d}} : \lambda \in R\}$ in R^n and its orthogonal complement, respectively. Therefore, from the above equations, we see that $\|\boldsymbol{y} - \bar{\boldsymbol{\xi}}\|^2 = \|(\bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T)(\boldsymbol{y} - \bar{\boldsymbol{\xi}})\|^2 + \|\left(\boldsymbol{I} - \bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T\right)(\boldsymbol{y} - \bar{\boldsymbol{\xi}})\|^2$ and $\|\boldsymbol{y} - \bar{\boldsymbol{y}}\|^2 = \|(\bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T)(\boldsymbol{y} - \bar{\boldsymbol{y}})\|^2 + \|\left(\boldsymbol{I} - \bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T\right)(\boldsymbol{y} - \bar{\boldsymbol{\xi}})\|^2$, which derive that

$$\begin{array}{rcl} \|\boldsymbol{y} - \bar{\boldsymbol{\xi}}\|^2 & = & \|\boldsymbol{y} - \bar{\boldsymbol{y}}\|^2 + \|(\bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T)(\boldsymbol{y} - \bar{\boldsymbol{\xi}})\|^2 - \|(\bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T)(\boldsymbol{y} - \bar{\boldsymbol{y}})\|^2 \\ & \leq & \|\boldsymbol{y} - \bar{\boldsymbol{y}}\|^2 + \|(\bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T)(\boldsymbol{y} - \bar{\boldsymbol{\xi}})\|^2 \\ & \leq & \text{diam}(C_0)^2 + \left(\bar{\boldsymbol{d}}^T(\boldsymbol{y} - \bar{\boldsymbol{\xi}})\right)^2 \ \ \text{(since both } \boldsymbol{y},\ \bar{\boldsymbol{y}} \in (F(\psi)) \subset C_0). \end{array}$$

Furthermore

$$\begin{array}{rcl} \bar{\boldsymbol{d}}^T(\bar{\boldsymbol{y}} - \bar{\boldsymbol{\xi}}) & \geq & \bar{\boldsymbol{d}}^T(\boldsymbol{y} - \bar{\boldsymbol{\xi}}) \ \ \text{(by (3.6))} \\ & = & \bar{\boldsymbol{d}}^T(\bar{\boldsymbol{y}} - \bar{\boldsymbol{\xi}} + \boldsymbol{y} - \bar{\boldsymbol{y}}) \\ & = & \bar{\boldsymbol{d}}^T\left((\tau - \delta)\bar{\boldsymbol{d}} + \boldsymbol{y} - \bar{\boldsymbol{y}}\right) \ \ \text{(by (3.7))} \\ & \geq & \tau - \delta - \|\boldsymbol{y} - \bar{\boldsymbol{y}}\| \ \ \text{(since } \|\bar{\boldsymbol{d}}\| = 1) \\ & \geq & \dfrac{\text{diam}(C_0)^2}{2\delta} + \dfrac{\delta}{2} - \delta - \text{diam}(C_0) \\ & & \text{(by } \boldsymbol{y},\ \bar{\boldsymbol{y}} \in (F(\psi)) \subset C_0 \text{ and (3.3))} \\ & = & \text{diam}(C_0)\left(\dfrac{\text{diam}(C_0)}{2\delta} - 1\right) - \dfrac{\delta}{2} \\ & \geq & \text{diam}(C_0) - \dfrac{\delta}{2} \ \ \text{(by (3.4))} \\ & \geq & 4\delta - \dfrac{\delta}{2} \ \ \text{(by (3.4))} \\ & > & 0. \end{array}$$

Hence

$$\begin{aligned}
\|\boldsymbol{y}-\bar{\boldsymbol{\xi}}\|^2 &\leq \operatorname{diam}(C_0)^2 + \left(\bar{\boldsymbol{d}}^T(\boldsymbol{y}-\bar{\boldsymbol{\xi}})\right)^2 \\
&\leq \operatorname{diam}(C_0)^2 + \left(\bar{\boldsymbol{d}}^T(\bar{\boldsymbol{y}}-\bar{\boldsymbol{\xi}})\right)^2 \\
&= \operatorname{diam}(C_0)^2 + \left(\bar{\boldsymbol{d}}^T(\tau-\delta)\bar{\boldsymbol{d}}\right)^2 \quad \text{(by (3.7))} \\
&= \operatorname{diam}(C_0)^2 + \tau^2 - 2\delta\tau + \delta^2 \quad \text{(since } \|\bar{\boldsymbol{d}}\| = 1) \\
&\leq \operatorname{diam}(C_0)^2 + \tau^2 - 2\delta\left(\frac{\operatorname{diam}(C_0)^2}{2\delta} + \frac{\delta}{2}\right) + \delta^2 \quad \text{(by (3.3))} \\
&= \tau^2.
\end{aligned}$$

Thus we have seen that c.hull$(F(\psi)) \subset B(\bar{\boldsymbol{\xi}}, \tau)$.

(iv) Finally we will show that the first relation of (3.5), *i.e.,*, $\bar{\boldsymbol{\xi}} \in \Xi$. From the definition of τ, We know form (3.3) and (3.4) that

$$\tau \geq \frac{\operatorname{diam}(C_0)^2}{2\delta} + \frac{\delta}{2} \geq \frac{\operatorname{diam}(C_0)}{2\delta} \times \operatorname{diam}(C_0) \geq 2\operatorname{diam}(C_0).$$

Hence if we choose a $\boldsymbol{y} \in$ c.hull$(F(\psi)) \subset C_0 \cap B(\bar{\boldsymbol{\xi}}, \tau)$, then

$$\|\bar{\boldsymbol{\xi}} - \boldsymbol{\xi}_0\| \leq \|\bar{\boldsymbol{\xi}} - \boldsymbol{y}\| + \|\boldsymbol{y} - \boldsymbol{\xi}_0\| \leq \tau + \operatorname{diam}(C_0) \leq 3\tau/2.$$

This implies that $\bar{\boldsymbol{\xi}} \in \Xi = B(\boldsymbol{\xi}_0, 3\tau/2)$. ∎

Corollary 3.1.2. *Assume that*

$$\boldsymbol{\xi}_0 \in C_0 \quad \textit{and} \quad 0 < \epsilon \leq \frac{\nu_{\text{lip}} \operatorname{diam}(C_0)}{2}. \tag{3.8}$$

Let

$$\psi = \frac{\epsilon}{2}, \; \tau' = \frac{2\nu_{\text{lip}} \operatorname{diam}(C_0)^2}{\epsilon} \; \textit{and } \Xi = B(\boldsymbol{\xi}_0, \tau'). \tag{3.9}$$

Then c.relax$(F(\psi), \Xi) \subset$ c.hull$(F(\epsilon))$.

Proof: Let $\delta = \dfrac{\epsilon}{2\,\nu_{\text{lip}}}$. By (3.8), δ satisfies the inequality (3.4) of Lemma 3.1.1. We also see that

$$\begin{aligned}
\frac{\operatorname{diam}(C_0)^2}{2\delta} + \frac{\delta}{2} &= \frac{\operatorname{diam}(C_0)^2}{2\delta} + \frac{\delta^2}{2\delta} \\
&\leq \frac{\operatorname{diam}(C_0)^2}{2\delta} + \frac{\operatorname{diam}(C_0)^2}{32\delta} \quad \text{(by (3.4))} \\
&= \frac{17\operatorname{diam}(C_0)^2}{32\delta} \\
&= \frac{17\,\nu_{\text{lip}} \operatorname{diam}(C_0)^2}{16\,\epsilon}.
\end{aligned}$$

Hence if we take $\tau = \dfrac{4\,\nu_{\text{lip}} \operatorname{diam}(C_0)^2}{3\,\epsilon}$, then τ satisfies (3.3) and the desired result follows. ∎

3.2 A Spherical-SDP Model

Throughout this section, we assume that $\nu_{\rm lip} = \nu_{\rm lip}(\mathcal{P}_F, C_0) < \infty$ and $\nu_{\rm nc} = \nu_{\rm nc}(\mathcal{P}_F) < \infty$. In the Spherical-SDP Model, we take $\mathcal{P}_k = \mathcal{P}^S(C_k)$ (the set of quadratic functions that induce spherical valid inequalities for C_k) in Step 2 of Algorithm 2.2.4. Here we say that a quadratic valid inequality $p(\boldsymbol{x}) \leq 0$ for C_k is spherical if $p(\cdot) : R^n \to R$ is of the form

$$p(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{\xi})^T(\boldsymbol{x} - \boldsymbol{\xi}) - \rho^2$$

for $\exists \boldsymbol{\xi} \in R^n$ and $\exists \rho > 0$. Let $\{C_k\}$ be the sequence of compact convex sets generated by Algorithm 2.2.4 with $\mathcal{P}_k = \mathcal{P}^S(C_k)$ at every iteration. Then the sequence $\{C_k\}$ enjoys properties (a) monotonicity, (b) detecting infeasibility, and (c) asymptotic convergence, which we stated in Theorem 2.3.1. Among these properties, only the monotonicity property is relevant to the succeeding discussions.

Let ψ be an arbitrary positive number, and Ξ an arbitrary nonempty compact convex set containing C_0. In Lemma 3.2.2, we first present an upper bound $\hat{k}$ for the number k of iterations at which C_k attain an (ψ, Ξ)-convex-relaxation of F, *i.e.*, $C_k \subset \text{c.relax}(F(\psi), \Xi)$ holds for the first time. Then in Theorem 3.2.3, we will apply Corollary 3.1.2 to this bound to derive an upper bound k^* for the number k of iterations at which C_k attains an ϵ-convex-relaxation of F, *i.e.*, $C_k \subset \text{c.hull}(F(\epsilon))$ for the first time. Here ϵ denotes an arbitrarily given positive number.

Let η=diam(Ξ). Define

$$\bar{\delta} = \begin{cases} 0 & \text{if } \nu_{\rm nc}\,\eta^2 \leq \psi, \\ \left(1 - \dfrac{\psi}{\nu_{\rm nc}\,\eta^2}\right)^{\frac{1}{2}} & \text{otherwise.} \end{cases}$$

By definition, we see that $0 \leq \bar{\delta} < 1$.

Lemma 3.2.1. *Let $k \in \{0, 1, 2, \dots\}$. For $\forall \boldsymbol{\xi} \in \Xi$, define*

$$\rho'(\boldsymbol{\xi}) = \max\left\{\rho\left(F(\psi), \boldsymbol{\xi}\right),\ \bar{\delta}\rho(C_k, \boldsymbol{\xi})\right\}.$$

Then

$$C_{k+1} \subset \bigcap_{\boldsymbol{\xi} \in \Xi} B(\boldsymbol{\xi}, \rho'(\boldsymbol{\xi})).$$

Proof: It suffices to show that $C_{k+1} \subset B(\boldsymbol{\xi}, \rho'(\boldsymbol{\xi}))$ for $\forall \boldsymbol{\xi} \in \Xi$. For an arbitrarily fixed $\boldsymbol{\xi} \in \Xi$, let

$$\rho_k = \rho(C_k, \boldsymbol{\xi}) \ \text{ and } \rho' = \max\left\{\rho\left(F(\psi), \boldsymbol{\xi}\right),\ \bar{\delta}\rho_k\right\}.$$

Since $\boldsymbol{\xi} \in \Xi$ and $C_k \subset C_0 \subset \Xi$, we see that

$$\rho_k \leq \eta, \tag{3.10}$$

which will be used later. If $\rho\left(F(\psi),\boldsymbol{\xi}\right) \geq \rho_k$ then $\rho' = \rho\left(F(\psi),\boldsymbol{\xi}\right) \geq \rho_k$. In this case the desired result follows from the monotonicity, *i.e.*, $C_{k+1} \subset C_k$. Now suppose that $\rho\left(F(\psi),\boldsymbol{\xi}\right) < \rho_k$. Assuming that $\bar{\boldsymbol{x}} \not\in B(\boldsymbol{\xi},\rho')$, we derive $\bar{\boldsymbol{x}} \not\in C_{k+1}$. If $\bar{\boldsymbol{x}} \not\in C_k$, we obviously see that $\bar{\boldsymbol{x}} \not\in C_{k+1}$ because $C_{k+1} \subset C_k$. Hence we only need to deal with the case that

$$\bar{\boldsymbol{x}} \in C_k \subset C_0,\ \rho\left(F(\psi),\boldsymbol{\xi}\right) \leq \rho' < \|\bar{\boldsymbol{x}}-\boldsymbol{\xi}\| \leq \rho_k. \tag{3.11}$$

We will show that

$$\begin{aligned} qf(\cdot;\bar{\gamma},\bar{\boldsymbol{q}},\bar{\boldsymbol{Q}}) + g(\cdot) &\in \mathcal{Q}_+, && (3.12)\\ qf(\bar{\boldsymbol{x}};\bar{\gamma},\bar{\boldsymbol{q}},\bar{\boldsymbol{Q}}) + g(\bar{\boldsymbol{x}}) &> 0. && (3.13)\end{aligned}$$

for $\exists qf(\cdot;\bar{\gamma},\bar{\boldsymbol{q}},\bar{\boldsymbol{Q}}) \in \mathcal{P}_F$ and $\exists g(\cdot) \in \text{c.cone}\left(\mathcal{P}_k\right) = \text{c.cone}\left(\mathcal{P}^S(C_k)\right)$ in 3 steps (i), (ii) and (iii) below. Then $\bar{\boldsymbol{x}} \not\in C_{k+1}$ follows from Lemma 2.2.3.

(i) The relations in (3.11) imply that $\bar{\boldsymbol{x}} \in C_0$ and $\bar{\boldsymbol{x}} \not\in F(\psi)$. Hence there exists a quadratic function $qf(\cdot;\bar{\gamma},\bar{\boldsymbol{q}},\bar{\boldsymbol{Q}}) \in \mathcal{P}_F$ such that $qf(\bar{\boldsymbol{x}};\bar{\gamma},\bar{\boldsymbol{q}},\bar{\boldsymbol{Q}}) > \psi$.

(ii) By the definition of ρ_k, we also know that the quadratic function

$$p(\boldsymbol{x}) = (\boldsymbol{x}-\boldsymbol{\xi})^T(\boldsymbol{x}-\boldsymbol{\xi}) - \rho_k^2 \ (\forall \boldsymbol{x} \in R^n)$$

is a member of $\mathcal{P}^S(C_k)$. Let $g(\cdot) = \nu_{\rm nc}\, p(\cdot)$. By the definition of $\nu_{\rm nc}$, all the eigenvalues of the Hessian matrix of the quadratic function $qf(\cdot;\bar{\gamma},\bar{\boldsymbol{q}},\bar{\boldsymbol{Q}}) \in \mathcal{P}_F$ are not less than $-\nu_{\rm nc}$. Hence (3.12) follows.

(iii) Finally we observe that

$$\begin{aligned} g(\bar{\boldsymbol{x}}) + qf(\bar{\boldsymbol{x}};\bar{\gamma},\bar{\boldsymbol{q}},\bar{\boldsymbol{Q}}) &= \nu_{\rm nc}\, p(\bar{\boldsymbol{x}}) + qf(\bar{\boldsymbol{x}};\bar{\gamma},\bar{\boldsymbol{q}},\bar{\boldsymbol{Q}})\\ &> \nu_{\rm nc}\ \left((\bar{\boldsymbol{x}}-\boldsymbol{\xi})^T(\bar{\boldsymbol{x}}-\boldsymbol{\xi}) - \rho_k^2\right) + \psi\\ &\geq \nu_{\rm nc}\ \left((\bar{\delta}\rho_k)^2 - \rho_k^2\right) + \psi \ \ (\text{since } \bar{\delta}\rho_k \leq \rho' < \|\bar{\boldsymbol{x}}-\boldsymbol{\xi}\|)\\ &= -\nu_{\rm nc}\ \left(1-\bar{\delta}^2\right)\rho_k^2 + \psi\\ &\geq -\nu_{\rm nc}\ \left(1-\bar{\delta}^2\right)\eta^2 + \psi \ \ (\text{by (3.10)})\\ &\geq 0 \ \ (\text{by the definition of } \bar{\delta}).\end{aligned}$$

Thus we have shown (3.13). ∎

Lemma 3.2.2. *Suppose that* $\operatorname{diam}(F) > 0$. *Define*

$$\hat{k} = \begin{cases} 1 & \textit{if } \nu_{\rm nc}\,\eta^2 \leq \psi,\\ \left\lceil \dfrac{2\nu_{\rm nc}\,\eta^2}{\psi} \ln \dfrac{2\eta}{\operatorname{diam}(F)} \right\rceil & \textit{otherwise.}\end{cases}$$

If $k \geq \hat{k}$, *then* $C_k \subset \text{c.relax}(F(\psi),\Xi)$.

Proof: For every $\boldsymbol{\xi} \in \Xi$ and $k = 0, 1, 2, \dots$, define

$$\rho_k(\boldsymbol{\xi}) = \max \left\{ \rho\left(F(\psi), \boldsymbol{\xi}\right), \rho(C_k, \boldsymbol{\xi}) \right\}.$$

It suffices to show that if $k \geq \hat{k}$ then

$$\rho_k(\boldsymbol{\xi}) \leq \rho\left(F(\psi), \boldsymbol{\xi}\right) \quad \text{for } \forall \boldsymbol{\xi} \in \Xi. \tag{3.14}$$

In fact, if (3.14) holds, then

$$C_k \subset \bigcap_{\boldsymbol{\xi} \in \Xi} B(\boldsymbol{\xi}, \rho_k(\boldsymbol{\xi})) \subset \bigcap_{\boldsymbol{\xi} \in \Xi} B\left(\boldsymbol{\xi}, \rho\left(F(\psi), \boldsymbol{\xi}\right)\right) = \text{c.relax}(F(\psi), \Xi).$$

By Lemma 3.2.1,

$$\rho_{k+1}(\boldsymbol{\xi}) \leq \max \left\{ \rho\left(F(\psi), \boldsymbol{\xi}\right), \bar{\delta} \rho_k(\boldsymbol{\xi}) \right\} \quad \text{for } \forall k = 0, 1, 2, \dots .$$

This implies that

$$\rho_k(\boldsymbol{\xi}) \leq \max \left\{ \rho\left(F(\psi), \boldsymbol{\xi}\right), \bar{\delta}^k \rho_0(\boldsymbol{\xi}) \right\} \quad \text{for } \forall \boldsymbol{\xi} \in B(\eta) \text{ and } \forall k = 0, 1, 2, \dots .$$

Hence, for each $\boldsymbol{\xi} \in \Xi$, if k satisfies the inequality

$$\bar{\delta}^k \rho_0(\boldsymbol{\xi}) \leq \rho\left(F(\psi), \boldsymbol{\xi}\right), \tag{3.15}$$

then $\rho_k(\boldsymbol{\xi}) \leq \rho\left(F(\psi), \boldsymbol{\xi}\right)$. When $\nu_{\text{nc}}\, \eta^2 \leq \psi$, we see that $\bar{\delta} = 0$ by definition. Hence (3.15) holds for $k = 1$. Now assume that $\nu_{\text{nc}}\, \eta^2 > \psi$. Then $\bar{\delta} > 0$. Since

$$F \subset F(\psi) \subset C_0 \subset \Xi,$$

we see that

$$\text{diam}(F)/2 \leq \rho\left(F(\psi), \boldsymbol{\xi}\right) \leq \rho_0(\boldsymbol{\xi}) \leq \eta \quad \text{for } \forall \boldsymbol{\xi} \in \Xi,$$

Hence, if $\bar{\delta}^k \eta \leq \text{diam}(F)/2$, or equivalently, if

$$k(-\ln \bar{\delta}) \geq \ln \frac{2\eta}{\text{diam}(F)},$$

then (3.15) holds. We also see from the definition of $\bar{\delta}$ that

$$-\ln \bar{\delta} \;=\; -\frac{1}{2} \ln \left(1 - \frac{\psi}{\nu_{\text{nc}}\, \eta^2}\right) \geq \frac{\psi}{2\nu_{\text{nc}}\, \eta^2} > 0.$$

Therefore if $\dfrac{k\psi}{2\nu_{\text{nc}}\, \eta^2} \geq \ln \dfrac{2\eta}{\text{diam}(F)}$, then (3.15) holds. Consequently we have shown that if $k \geq \dfrac{2\nu_{\text{nc}}\, \eta^2}{\psi} \ln \dfrac{2\eta}{\text{diam}(F)}$, then (3.14) holds. ∎

Now we are ready to present our main result in this section.

Theorem 3.2.3. *Assume (3.8) as in Corollary 3.1.2 and* $\mathrm{diam}(F) > 0$*. Define*

$$k^* = \begin{cases} 1 \quad \textit{if } (\nu_{\mathrm{lip}})^2\ \nu_{\mathrm{nc}}\ \mathrm{diam}(C_0)^4 \leq \dfrac{\epsilon^3}{32}, \\ \left\lceil \dfrac{64\ (\nu_{\mathrm{lip}})^2\ \nu_{\mathrm{nc}}\ \mathrm{diam}(C_0)^4}{\epsilon^3} \ln \dfrac{8\ \nu_{\mathrm{lip}}\ \mathrm{diam}(C_0)^2}{\epsilon\ \mathrm{diam}(F)} \right\rceil & \textit{otherwise.} \end{cases}$$

If $k \geq k^*$*, then* $C_k \subset \mathrm{c.hull}(F(\epsilon))$.

It is interesting to note that the bound k^* is proportional to the nonconvexity ν_{nc} of $\mathcal{P}_F$, and $k^* = 1$ when quadratic functions in $\mathcal{P}_F$ are almost convex.

Proof of Theorem 3.2.3: Choose positive numbers ψ, τ' and a compact convex set $\Xi \subset R^n$ as in (3.9) of Corollary 3.1.2. Then $\mathrm{c.relax}(F(\psi), \Xi) \subset \mathrm{c.hull}(F(\epsilon))$. Let $\eta = \mathrm{diam}(\Xi) = 2\tau'$. Now, define $\hat{k}$ as in Lemma 3.2.2. By Lemma 3.2.2, if $k \geq \hat{k}$ then

$$C_k \subset \mathrm{c.relax}(F(\psi), \Xi) \subset \mathrm{c.hull}(F(\epsilon)).$$

On the other hand, we see that

$$\begin{aligned} \nu_{\mathrm{nc}}\ \eta^2 &= \nu_{\mathrm{nc}}\ (2\tau')^2 \\ &= \nu_{\mathrm{nc}} \left(\frac{4\ \nu_{\mathrm{lip}}\ \mathrm{diam}(C_0)^2}{\epsilon} \right)^2 \\ &= \frac{16\ (\nu_{\mathrm{lip}})^2\ \nu_{\mathrm{nc}}\ \mathrm{diam}(C_0)^4}{\epsilon^2}. \end{aligned}$$

Hence the inequality $\nu_{\mathrm{nc}}\ \eta^2 \leq \psi$ appeared in the definition of $\hat{k}$ can be rewritten as

$$(\nu_{\mathrm{lip}})^2\ \nu_{\mathrm{nc}}\ \mathrm{diam}(C_0)^4 \leq \frac{\epsilon^3}{32}.$$

We also see that

$$\begin{aligned} & \frac{2\ \nu_{\mathrm{nc}}\ \eta^2}{\psi} \ln \frac{2\eta}{\mathrm{diam}(F)} \\ &= \frac{2\ \nu_{\mathrm{nc}}\ (2\tau')^2}{\epsilon/2} \ln \frac{4\ \tau'}{\mathrm{diam}(F)} \\ &= \frac{64\ (\nu_{\mathrm{lip}})^2\ \nu_{\mathrm{nc}}\ \mathrm{diam}(C_0)^4}{\epsilon^3} \ln \frac{8\ \nu_{\mathrm{lip}}\ \mathrm{diam}(C_0)^2}{\epsilon\ \mathrm{diam}(F)}. \end{aligned}$$

Therefore $k^* = \hat{k}$. ∎

3.3 Rank-2 Models

We discuss Rank-2-SDP and Rank-2-SILP Models of conceptual SCRMs simultaneously, which are introduced in Section 2.3.1. For simplicity of notation, we use $\widetilde{\mathcal{P}}^2(C_k)$ instead of $\mathcal{P}^2(C_k, \pm D_1, D_2)$ with $D_1 = \pm I$ and $D_2 = \overline{D}$, i.e.,

$$\widetilde{\mathcal{P}}^2(C_k) \quad = \quad \{ r2sf(\cdot; C_k, \boldsymbol{d}_1, \boldsymbol{d}_2) : \boldsymbol{d}_1 \in \pm I,\ \boldsymbol{d}_2 \in \overline{D} \}.$$

in this section. In both Rank-2 models, we take $\mathcal{P}_k = \widetilde{\mathcal{P}}^2(C_k)$ in Step 2 of Algorithms 2.2.4 and 2.2.5, respectively. Let $\{C_k\}$ be the sequence of compact convex sets generated by Algorithm 2.2.4 or Algorithm 2.2.5. Then the sequence $\{C_k\}$ enjoys properties (a) monotonicity, (b) detecting infeasibility, and (c) asymptotic convergence, which we stated in Section 2.3.1. Let ϵ be an arbitrarily given positive number. We will derive an upper bound k^* in Theorem 3.3.4 at the end of this section for the number k of iterations at which C_k attains an ϵ-convex-relaxation of F. The argument of the derivation of the bound k^* will proceed in a similar way as in the Spherical-SDP Model, although it is more complicated. In the derivation of the bound in the Spherical-SDP Model, it was a key to show the existence of a quadratic function $g(\cdot) \in \text{c.cone}(\mathcal{P}_k) = \text{c.cone}\left(\mathcal{P}^S(C_k)\right)$ satisfying the relations (3.12) and (3.13). We need a sophisticated argument, which we develop in Section 3.3.1, to construct a quadratic function $g(\cdot) \in \text{c.cone}(\mathcal{P}_k) = \text{c.cone}\left(\widetilde{\mathcal{P}}^2(C_k)\right)$ satisfying the corresponding relations (3.22) and (3.23) in the Rank-2-SDP and the SILP Models.

3.3.1 Convex Cones of Rank-2 Quadratic Supporting Functions for n-dimensional Balls

In this subsection, we are concerned with $\text{c.cone}\left(\widetilde{\mathcal{P}}^2(B(\boldsymbol{\xi},\rho))\right)$, where $\boldsymbol{\xi} \in R^n$ and $\rho > 0$. Let $\boldsymbol{d}_1 \in \pm I$ and $\boldsymbol{d}_2 \in \overline{D}$. Note that if $C_k \subset B(\boldsymbol{\xi},\rho)$, $r2sf(\cdot; B(\boldsymbol{\xi},\rho), \boldsymbol{d}_1, \boldsymbol{d}_2) \in \widetilde{\mathcal{P}}^2(B(\boldsymbol{\xi},\rho))$ and $r2sf(\cdot; C_k, \boldsymbol{d}_1, \boldsymbol{d}_2) \in \widetilde{\mathcal{P}}^2(B(\boldsymbol{\xi},\rho))$, then $r2sf(\cdot; B(\boldsymbol{\xi},\rho), \boldsymbol{d}_1, \boldsymbol{d}_2)$ induces a weaker valid inequality for C_k than $r2sf(\cdot; C_k, \boldsymbol{d}_1, \boldsymbol{d}_2)$ in the sense that

$$r2sf(\boldsymbol{x}; B(\boldsymbol{\xi},\rho), \boldsymbol{d}_1, \boldsymbol{d}_2) \le r2sf(\boldsymbol{x}; C_k, \boldsymbol{d}_1, \boldsymbol{d}_2) \le 0 \ \text{ for } \forall \boldsymbol{x} \in C_k.$$

This fact will be utilized in our complexity analysis of the Rank-2 Models in the next subsection. For simplicity of notation, we only deal with the case that $\boldsymbol{\xi} = \boldsymbol{0}$.

For $\forall\theta \in (0, \pi/8]$, $\forall \boldsymbol{w} \in \overline{D}$, and $\forall i = 1, 2, \ldots, n$, define

$$\left.\begin{array}{ll} \nu_i(\theta,\boldsymbol{w}) = \|\boldsymbol{w}\cos\theta + \boldsymbol{e}_i \sin\theta\|, & \bar{\nu}_i(\theta,\boldsymbol{w}) = \|\boldsymbol{w}\cos\theta - \boldsymbol{e}_i \sin\theta\|, \\ \boldsymbol{b}_i(\theta,\boldsymbol{w}) = \dfrac{\boldsymbol{w}\cos\theta + \boldsymbol{e}_i\sin\theta}{\nu_i(\theta,\boldsymbol{w})} \in \overline{D}, & \bar{\boldsymbol{b}}_i(\theta,\boldsymbol{w}) = \dfrac{\boldsymbol{w}\cos\theta - \boldsymbol{e}_i\sin\theta}{\bar{\nu}_i(\theta,\boldsymbol{w})} \in \overline{D}, \\ \lambda_i(\theta,\boldsymbol{w}) = \dfrac{\nu_i(\theta,\boldsymbol{w})}{2\sin\theta}, & \bar{\lambda}_i(\theta,\boldsymbol{w}) = \dfrac{\bar{\nu}_i(\theta,\boldsymbol{w})}{2\sin\theta}. \end{array}\right\}$$

For $\forall \boldsymbol{x} \in R^n$, $\forall \rho > 0$, $\forall \theta \in (0, \pi/8]$, $\forall \boldsymbol{w} \in \overline{D}$, $\forall i = 1, 2, \ldots, n$, and $\forall j = 1, 2, \ldots, n$, define

$$\left.\begin{array}{rcl} f^+_{ij}(\boldsymbol{x};\rho,\theta,\boldsymbol{w}) & = & \lambda_i(\theta,\boldsymbol{w})\ r2sf(\boldsymbol{x}; B(\boldsymbol{0},\rho), -\boldsymbol{e}_j, \boldsymbol{b}_i(\theta,\boldsymbol{w})) \\ & & +\bar{\lambda}_i(\theta,\boldsymbol{w})\ r2sf(\boldsymbol{x}; B(\boldsymbol{0},\rho), \boldsymbol{e}_j, \bar{\boldsymbol{b}}_i(\theta,\boldsymbol{w})) \\ & = & -\lambda_i(\theta,\boldsymbol{w})\left(\boldsymbol{b}_i(\theta,\boldsymbol{w})^T\boldsymbol{x} - \rho\right)\left(-\boldsymbol{e}_j^T\boldsymbol{x} - \alpha(-\boldsymbol{e}_j, C_0)\right) \\ & & -\bar{\lambda}_i(\theta,\boldsymbol{w})\left(\bar{\boldsymbol{b}}_i(\theta,\boldsymbol{w})^T\boldsymbol{x} - \rho\right)\left(\boldsymbol{e}_j^T\boldsymbol{x} - \alpha(\boldsymbol{e}_j, C_0)\right), \\ & & \\ f^-_{ij}(\boldsymbol{x};\rho,\theta,\boldsymbol{w}) & = & \lambda_i(\theta,\boldsymbol{w})\ r2sf(\boldsymbol{x}; B(\boldsymbol{0},\rho), \boldsymbol{e}_j, \boldsymbol{b}_i(\theta,\boldsymbol{w})) \\ & & +\bar{\lambda}_i(\theta,\boldsymbol{w})\ r2sf(\boldsymbol{x}; B(\boldsymbol{0},\rho), -\boldsymbol{e}_j, \bar{\boldsymbol{b}}_i(\theta,\boldsymbol{w})) \\ & = & -\lambda_i(\theta,\boldsymbol{w})\left(\boldsymbol{b}_i(\theta,\boldsymbol{w})^T\boldsymbol{x} - \rho\right)\left(\boldsymbol{e}_j^T\boldsymbol{x} - \alpha(\boldsymbol{e}_j, C_0)\right) \\ & & -\bar{\lambda}_i(\theta,\boldsymbol{w})\left(\bar{\boldsymbol{b}}_i(\theta,\boldsymbol{w})^T\boldsymbol{x} - \rho\right)\left(-\boldsymbol{e}_j^T\boldsymbol{x} - \alpha(-\boldsymbol{e}_j, C_0)\right). \end{array}\right\} \quad (3.16)$$

Essentially the same functions as the quadratic functions $f^+_{ij}(\cdot;\rho,\theta,\boldsymbol{w})$ and $f^-_{ij}(\cdot;\rho,\theta,\boldsymbol{w})$ $(i,j=1,2,\ldots,n)$ were introduced in the paper [33], and the lemma below is a further elaboration of Lemma 4.4 of [33] on their basic properties.

Lemma 3.3.1. *Let $\rho>0$, $\theta\in(0,\pi/8]$, $\boldsymbol{w}\in\overline{D}$, $i\in\{1,2,\ldots,n\}$, and $j\in\{1,2,\ldots,n\}$.*

(i) $f^+_{ij}(\cdot;\rho,\theta,\boldsymbol{w}),\ f^-_{ij}(\cdot;\rho,\theta,\boldsymbol{w})\in c.cone\Big(\widetilde{\mathcal{P}}^2(B(\mathbf{0},\rho))\Big)$.

(ii)$^+$ The Hessian matrix of the quadratic function $f^+_{ij}(\cdot;\rho,\theta,\boldsymbol{w}):R^n\to R$ coincides with the $n\times n$ matrix $\dfrac{\boldsymbol{e}_i\boldsymbol{e}_j^T+\boldsymbol{e}_j\boldsymbol{e}_i^T}{2}$.

(ii)$^-$ The Hessian matrix of the quadratic function $f^-_{ij}(\cdot;\rho,\theta,\boldsymbol{w}):R^n\to R$ coincides with the $n\times n$ matrix $-\dfrac{\boldsymbol{e}_i\boldsymbol{e}_j^T+\boldsymbol{e}_j\boldsymbol{e}_i^T}{2}$.

(iii) Suppose that $\delta\in[0,1]$ and $\delta\rho\boldsymbol{w}\in C_0$. Then

$$f^+_{ij}(\delta\rho\boldsymbol{w};\rho,\theta,\boldsymbol{w}),\ f^-_{ij}(\delta\rho\boldsymbol{w};\rho,\theta,\boldsymbol{w})\in\left[-\rho\,\mathrm{diam}(C_0)\left(\frac{1-\delta(1-\theta^2/2)}{\theta-\theta^3/6}+1-\delta+\frac{\theta^2}{2}\right),\ 0\right].$$

(iv) Suppose that $\kappa\geq0$, $1\geq\delta\geq1-\kappa\theta\geq0$ and $\delta\rho\boldsymbol{w}\in C_0$. Then

$$f^+_{ij}(\delta\rho\boldsymbol{w};\rho,\theta,\boldsymbol{w}),\ f^-_{ij}(\delta\rho\boldsymbol{w};\rho,\theta,\boldsymbol{w})\in[-2\rho\,\mathrm{diam}(C_0)(\kappa+\theta),\ 0].$$

Proof: We will only show the relations on $f^+_{ij}(\cdot;\rho,\theta,\boldsymbol{w})$ because we can derive the corresponding relations on $f^-_{ij}(\cdot;\rho,\theta,\boldsymbol{w})$ quite similarly. First, assertion (i) follows directly from the definitions (3.16).

(ii)$^+$. The Hessian matrix of the quadratic function $f^+_{ij}(\cdot;\rho,\theta,\boldsymbol{w}):R^n\to R$ turns out to be the symmetric part of the matrix

$$\begin{aligned}&-\Big(-\lambda_i(\theta,\boldsymbol{w})\ \boldsymbol{b}_i(\theta,\boldsymbol{w})\boldsymbol{e}_j^T+\bar{\lambda}_i(\theta,\boldsymbol{w})\ \bar{\boldsymbol{b}}_i(\theta,\boldsymbol{w})\boldsymbol{e}_j^T\Big)\\ &=\ -\frac{1}{2\sin\theta}\Big(-(\boldsymbol{w}\cos\theta+\boldsymbol{e}_i\sin\theta)\boldsymbol{e}_j^T+(\boldsymbol{w}\cos\theta-\boldsymbol{e}_i\sin\theta)\boldsymbol{e}_j^T\Big)\\ &=\ -\frac{1}{2\sin\theta}\Big(-2(\sin\theta)(\boldsymbol{e}_i\boldsymbol{e}_j^T)\Big)\\ &=\ \boldsymbol{e}_i\boldsymbol{e}_j^T.\end{aligned}$$

Thus we have shown (ii)$^+$.

(iii) By definition, we have that

$$\begin{aligned}&f^+_{ij}(\delta\rho\boldsymbol{w};\rho,\theta,\boldsymbol{w})\\ &\quad=\ -\frac{1}{2\sin\theta}\Big(\ \rho\Big((\boldsymbol{w}\cos\theta+\boldsymbol{e}_i\sin\theta)^T(\delta\boldsymbol{w})-\nu_i(\theta,\boldsymbol{w})\Big)\Big(-\boldsymbol{e}_j^T(\delta\rho\boldsymbol{w})-\alpha(-\boldsymbol{e}_j,C_0)\Big)\\ &\qquad\quad+\ \rho\Big((\boldsymbol{w}\cos\theta-\boldsymbol{e}_i\sin\theta)^T(\delta\boldsymbol{w})-\bar{\nu}_i(\theta,\boldsymbol{w})\Big)\Big(\boldsymbol{e}_j^T(\delta\rho\boldsymbol{w})-\alpha(\boldsymbol{e}_j,C_0)\Big)\ \Big).\end{aligned}$$

We will evaluate each term appeared in the right hand side. We first observe that

$$\begin{aligned}
\nu_i(\theta,\boldsymbol{w}) &= \|\boldsymbol{w}\cos\theta+\boldsymbol{e}_i\sin\theta\| \\
&= \left(\|\boldsymbol{w}\cos\theta\|^2+\|\boldsymbol{e}_i\sin\theta\|^2+2\boldsymbol{e}_i^T\boldsymbol{w}\cos\theta\sin\theta\right)^{1/2} \\
&= \left(1+2\boldsymbol{e}_i^T\boldsymbol{w}\cos\theta\sin\theta\right)^{1/2} \\
&\leq 1+\boldsymbol{e}_i^T\boldsymbol{w}\cos\theta\sin\theta,
\end{aligned}$$

where the last inequality follows from

$$0 \leq (1+\xi)^{1/2} \leq 1+\frac{1}{2}\xi \ \text{ for } \forall\xi \geq -1.$$

Hence

$$\begin{aligned}
0 &\geq (\boldsymbol{w}\cos\theta+\boldsymbol{e}_i\sin\theta)^T(\delta\boldsymbol{w})-\nu_i(\theta,\boldsymbol{w}) \\
&\geq \delta(\cos\theta+\boldsymbol{e}_i^T\boldsymbol{w}\sin\theta)-\left(1+\boldsymbol{e}_i^T\boldsymbol{w}\cos\theta\sin\theta\right) \\
&= \delta\cos\theta-1+\boldsymbol{e}_i^T\boldsymbol{w}(\sin\theta)(\delta-\cos\theta) \\
&\geq \delta\cos\theta-1-(\sin\theta)|\cos\theta-\delta|.
\end{aligned}$$

Similarly

$$\begin{aligned}
\bar{\nu}_i(\theta,\boldsymbol{w}) &= \|\boldsymbol{w}\cos\theta-\boldsymbol{e}_i\sin\theta\| \\
&= \left(\|\boldsymbol{w}\cos\theta\|^2+\|\boldsymbol{e}_i\sin\theta\|^2-2\boldsymbol{e}_i^T\boldsymbol{w}\cos\theta\sin\theta\right)^{1/2} \\
&= \left(1-2\boldsymbol{e}_i^T\boldsymbol{w}\cos\theta\sin\theta\right)^{1/2} \\
&\leq 1-\boldsymbol{e}_i^T\boldsymbol{w}\cos\theta\sin\theta, \\
0 &\geq (\boldsymbol{w}\cos\theta-\boldsymbol{e}_i\sin\theta)^T(\delta\boldsymbol{w})-\bar{\nu}_i(\theta,\boldsymbol{w}) \\
&\geq \delta(\cos\theta-\boldsymbol{e}_i^T\boldsymbol{w}\sin\theta)-\left(1-\boldsymbol{e}_i^T\boldsymbol{w}\cos\theta\sin\theta\right) \\
&= \delta\cos\theta-1-\boldsymbol{e}_i^T\boldsymbol{w}(\sin\theta)(\delta-\cos\theta) \\
&\geq \delta\cos\theta-1-(\sin\theta)|\cos\theta-\delta|.
\end{aligned}$$

Since $\delta\rho\boldsymbol{w}\in C_0$, we also see that

$$\begin{aligned}
0 &\geq -\boldsymbol{e}_j^T(\delta\rho\boldsymbol{w})-\alpha(-\boldsymbol{e}_j,C_0) &&\geq -\mathrm{diam}(C_0), \\
0 &\geq \boldsymbol{e}_j^T(\delta\rho\boldsymbol{w})-\alpha(\boldsymbol{e}_j,C_0) &&\geq -\mathrm{diam}(C_0).
\end{aligned}$$

Therefore

$$\begin{aligned}
f_{ij}^+(\delta\rho\boldsymbol{w};\rho,\theta,\boldsymbol{w}) &\geq -\frac{\rho}{2\sin\theta}\times 2\left(\delta\cos\theta-1-(\sin\theta)|\cos\theta-\delta|\right)(-\mathrm{diam}(C_0)) \\
&= -\frac{\rho\,\mathrm{diam}(C_0)}{\sin\theta}\left(1-\delta\cos\theta+(\sin\theta)|\cos\theta-\delta|\right) \\
&= -\rho\,\mathrm{diam}(C_0)\left(\frac{1-\delta\cos\theta}{\sin\theta}+|\cos\theta-\delta|\right) \\
&= -\rho\,\mathrm{diam}(C_0)\left(\frac{1-\delta(1-\theta^2/2+\tilde{\theta}^4/24)}{\theta-\theta^3/6+\bar{\theta}^5/120}+|1-\hat{\theta}^2/2-\delta|\right) \\
&\geq -\rho\,\mathrm{diam}(C_0)\left(\frac{1-\delta(1-\theta^2/2)}{\theta-\theta^3/6}+1-\delta+\frac{\theta^2}{2}\right) \quad (\text{since } 1\geq\delta).
\end{aligned}$$

Here $\tilde{\theta},\ \bar{\theta},\ \hat{\theta} \in [0,\theta]$.

(iv) We see from (iii) that

$$\begin{aligned}
f^+_{ij}(\delta\rho\boldsymbol{w};\rho,\theta,\boldsymbol{w}) &\geq -\rho\,\mathrm{diam}(C_0)\left(\frac{1-\delta(1-\theta^2/2)}{\theta(1-\theta^2/6)} + 1 - \delta + \frac{\theta^2}{2}\right) \\
&\geq -\rho\,\mathrm{diam}(C_0)\left(\frac{\kappa\theta + \delta\theta^2/2}{\theta(1-\theta^2/6)} + \kappa\theta + \frac{\theta^2}{2}\right) \quad (\text{since } 1 \geq \delta \geq 1-\kappa\theta) \\
&\geq -\rho\,\mathrm{diam}(C_0)\left(\frac{\kappa + \theta/2}{1-\theta^2/6} + \kappa\theta + \frac{\theta^2}{2}\right) \quad (\text{since } 1 \geq \delta) \\
&\geq -2\rho\,\mathrm{diam}(C_0)(\kappa+\theta) \quad (\text{since } \theta \in (0,\ \pi/8]).
\end{aligned}$$

∎

3.3.2 Complexity Analysis

In the remainder of the section, we assume that $\nu_{\mathrm{lip}} = \nu_{\mathrm{lip}}(\mathcal{P}_F, C_0) < \infty$ and $\nu_{\mathrm{nl}} = \nu_{\mathrm{nl}}(\mathcal{P}_F) < \infty$. Let $\{C_k\}$ be the sequence of compact convex sets generated by either Algorithm 2.2.4 or Algorithm 2.2.5 with taking $\mathcal{P}_k = \widetilde{\mathcal{P}}^2(C_k)$ at each iteration. Let ψ be an arbitrary positive number, and Ξ an arbitrary compact convex set containing C_0. In Lemma 3.3.3, we derive an upper bound $\hat{k}$ for the number k of iterations at which C_k attains a (ψ,Ξ)-convex-relaxation of F, *i.e.*, $C_k \subset \mathrm{c.relax}(F(\psi),\Xi)$ holds for the first time. Then we will combine Lemmas 3.1.1 and 3.3.3 to derive an upper bound k^* for the number k of iterations at which C_k attains an ϵ-convex-relaxation of F. Here ϵ denotes an arbitrarily given positive number.

Let η=diam(Ξ). Define

$$\begin{aligned}
\bar{\kappa} &= \begin{cases} \dfrac{8}{\pi} & \text{if } \nu_{\mathrm{nl}} \leq \dfrac{\pi\,\psi}{32\,\eta\,\mathrm{diam}(C_0)}, \\[2ex] \dfrac{\psi}{4\eta\,\nu_{\mathrm{nl}}\,\mathrm{diam}(C_0)} & \text{otherwise,} \end{cases} \\
\bar{\theta} &= \begin{cases} \dfrac{\pi}{8} & \text{if } \nu_{\mathrm{nl}} \leq \dfrac{2\psi}{\pi\,\eta\,\mathrm{diam}(C_0)}, \\[2ex] \dfrac{\psi}{4\eta\,\nu_{\mathrm{nl}}\,\mathrm{diam}(C_0)} & \text{otherwise,} \end{cases} \\
\bar{\delta} &= 1 - \bar{\kappa}\bar{\theta}.
\end{aligned}$$

It should be noted that $\bar{\kappa}$ and $\bar{\theta}$ are not greater than $\dfrac{8}{\pi}$ and $\dfrac{\pi}{8}$, respectively, and also that $\dfrac{\pi\,\psi}{32\,\eta\,\mathrm{diam}(C_0)} \leq \dfrac{2\psi}{\pi\,\eta\,\mathrm{diam}(C_0)}$. By definition, we see that

$$\left.\begin{array}{lllll} 1 & = & \bar{\kappa}\bar{\theta} & & \text{if } \nu_{\mathrm{nl}} \leq \dfrac{\pi\,\psi}{32\,\eta\,\mathrm{diam}(C_0)}, \\[2ex] 1 & > & \bar{\kappa}\bar{\theta} & \geq \left(\dfrac{\psi}{4\eta\,\nu_{\mathrm{nl}}\,\mathrm{diam}(C_0)}\right)^2 & \text{otherwise,} \end{array}\right\} \tag{3.17}$$

$$\left.\begin{array}{rcl} \bar{\delta} & = & 0 \quad \text{if } \nu_{\rm nl} \leq \dfrac{\pi\,\psi}{32\,\eta\,\mathrm{diam}(C_0)}, \\ 1 \; > \; \bar{\delta} & > & 0 \quad \text{otherwise}, \end{array}\right\} \tag{3.18}$$

$$\frac{\psi}{2\eta\,\mathrm{diam}(C_0)} \;\geq\; \left(\bar{\kappa}+\bar{\theta}\right)\nu_{\rm nl} \;\geq\; 0. \tag{3.19}$$

Lemma 3.3.2. *Let $k \in \{0,1,2,\dots\}$. For $\forall \boldsymbol{\xi} \in \Xi$, define*

$$\rho'(\boldsymbol{\xi}) = \max\left\{\rho\left(F(\psi),\boldsymbol{\xi}\right),\ \bar{\delta}\rho(C_k,\boldsymbol{\xi})\right\}.$$

Then

$$C_{k+1} \subset \bigcap_{\boldsymbol{\xi}\,\in\,\Xi} B(\boldsymbol{\xi},\rho'(\boldsymbol{\xi})).$$

Proof: It suffices to show that $C_{k+1} \subset B(\boldsymbol{\xi},\rho'(\boldsymbol{\xi}))$ for $\forall\boldsymbol{\xi} \in \Xi$. For an arbitrarily fixed $\boldsymbol{\xi} \in \Xi$, let

$$\rho_k = \rho(C_k,\boldsymbol{\xi}) \ \text{ and } \rho' = \max\left\{\rho\left(F(\psi),\boldsymbol{\xi}\right),\ \bar{\delta}\rho_k\right\}.$$

We may assume without loss of generality that $\boldsymbol{\xi} = \mathbf{0}$ because Algorithms 2.2.4 and 2.2.5 with taking $\mathcal{P}_k = \widetilde{\mathcal{P}}^2(C_k)$ at each iteration are invariant under any parallel transformation. See [33] for more details. Since $\boldsymbol{\xi} = \mathbf{0} \in \Xi$ and $C_k \subset C_0 \subset \Xi$, we see that

$$\rho_k \leq \eta, \tag{3.20}$$

which will be used later. If $\rho\left(F(\psi),\mathbf{0}\right) \geq \rho_k$ then $\rho' = \rho\left(F(\psi),\mathbf{0}\right) \geq \rho_k$. In this case the desired result follows from $C_{k+1} \subset C_k$. Now suppose that $\rho\left(F(\psi),\mathbf{0}\right) < \rho_k$. Assuming that $\bar{\boldsymbol{x}} \notin B(\mathbf{0},\rho')$, we will derive that $\bar{\boldsymbol{x}} \notin C_{k+1}$. If $\bar{\boldsymbol{x}} \notin C_k$, we obviously see $\bar{\boldsymbol{x}} \notin C_{k+1}$ because $C_{k+1} \subset C_k$. Hence we only need to deal with the case that

$$\bar{\boldsymbol{x}} \in C_k \subset C_0,\ \rho\left(F(\psi),\mathbf{0}\right) \leq \rho' < \|\bar{\boldsymbol{x}} - \mathbf{0}\| \leq \rho_k. \tag{3.21}$$

We will show that

$$qf(\cdot;\bar{\gamma},\bar{\boldsymbol{q}},\bar{\boldsymbol{Q}}) + g(\cdot) \in \mathcal{L}, \tag{3.22}$$
$$qf(\bar{\boldsymbol{x}};\bar{\gamma},\bar{\boldsymbol{q}},\bar{\boldsymbol{Q}}) + g(\bar{\boldsymbol{x}}) > 0. \tag{3.23}$$

for $\exists qf(\cdot;\bar{\gamma},\bar{\boldsymbol{q}},\bar{\boldsymbol{Q}}) \in \mathcal{P}_F$ and $\exists g(\cdot) \in \mathcal{P}_k = \widetilde{\mathcal{P}}^2(C_k)$ in 3 steps (i), (ii) and (iii) below. Since $\mathcal{L} \subset \mathcal{Q}_+$, Lemma 2.2.3 shows $\bar{\boldsymbol{x}} \notin C_{k+1}$ in both cases of the Rank-2-SDP Model (Algorithm 2.2.4) and the SILP Model (Algorithm 2.2.5).

(i) The relations in (3.21) imply that $\bar{\boldsymbol{x}} \in C_0$ and $\bar{\boldsymbol{x}} \notin F(\psi)$. Hence there exists a quadratic function $qf(\cdot;\bar{\gamma},\bar{\boldsymbol{q}},\bar{\boldsymbol{Q}}) \in \mathcal{P}_F$ such that $qf(\bar{\boldsymbol{x}};\bar{\gamma},\bar{\boldsymbol{q}},\bar{\boldsymbol{Q}}) > \psi$.

(ii) Let $\bar{\boldsymbol{w}} = \bar{\boldsymbol{x}}/\|\bar{\boldsymbol{x}}\|$, and $\delta = \|\bar{\boldsymbol{x}}\|/\rho_k$. Then we see that

$$\begin{array}{rcl} \bar{\boldsymbol{x}} & = & \|\bar{\boldsymbol{x}}\|\bar{\boldsymbol{w}} = \delta\rho_k\bar{\boldsymbol{w}}, \\ 1 & \geq & \delta = \|\bar{\boldsymbol{x}}\|/\rho_k > \rho'/\rho_k \geq \bar{\delta} = 1 - \bar{\kappa}\bar{\theta} \geq 0. \end{array}$$

We will represent the symmetric matrix $\bar{\boldsymbol{Q}}$ as the difference of two $n \times n$ symmetric matrices $\bar{\boldsymbol{Q}}^+$ and $\bar{\boldsymbol{Q}}^-$ with nonnegative elements such that

$$\bar{\boldsymbol{Q}} = \bar{\boldsymbol{Q}}^+ - \bar{\boldsymbol{Q}}^-, \ \bar{Q}^+_{ij} \geq 0, \ \bar{Q}^-_{ij} \geq 0, \ \text{ and } \bar{Q}^+_{ij}\bar{Q}^-_{ij} = 0 \ (i,j = 1,2,\ldots,n).$$

For $\forall \boldsymbol{x} \in R^n$, $\forall i = 1,2,\ldots,n$, and $\forall j = 1,2,\ldots,n$, we now define

$$\left.\begin{array}{rcl} g^+_{ij}(\boldsymbol{x}) & = & \lambda_i(\bar{\theta},\bar{\boldsymbol{w}})\ r2sf(\boldsymbol{x}; C_k, -\boldsymbol{e}_j, \boldsymbol{b}_i(\bar{\theta},\bar{\boldsymbol{w}})) \\ & & +\bar{\lambda}_i(\bar{\theta},\bar{\boldsymbol{w}})\ r2sf(\boldsymbol{x}; C_k, \boldsymbol{e}_j, \bar{\boldsymbol{b}}_i(\bar{\theta},\bar{\boldsymbol{w}})) \\ & = & -\lambda_i(\bar{\theta},\bar{\boldsymbol{w}}) \left(\boldsymbol{b}_i(\bar{\theta},\bar{\boldsymbol{w}})^T\boldsymbol{x} - \alpha(\boldsymbol{b}_i(\bar{\theta},\bar{\boldsymbol{w}}), C_k)\right) \left(-\boldsymbol{e}_j^T\boldsymbol{x} - \alpha(-\boldsymbol{e}_j, C_0)\right) \\ & & -\bar{\lambda}_i(\bar{\theta},\bar{\boldsymbol{w}}) \left(\bar{\boldsymbol{b}}_i(\bar{\theta},\bar{\boldsymbol{w}})^T\boldsymbol{x} - \alpha(\bar{\boldsymbol{b}}_i(\bar{\theta},\bar{\boldsymbol{w}}), C_k)\right) \left(\boldsymbol{e}_j^T\boldsymbol{x} - \alpha(\boldsymbol{e}_j, C_0)\right), \\ & & \\ g^-_{ij}(\boldsymbol{x}) & = & \lambda_i(\bar{\theta},\bar{\boldsymbol{w}})\ r2sf(\boldsymbol{x}; C_k, \boldsymbol{e}_j, \boldsymbol{b}_i(\bar{\theta},\bar{\boldsymbol{w}})) \\ & & +\bar{\lambda}_i(\bar{\theta},\bar{\boldsymbol{w}})\ r2sf(\boldsymbol{x}; C_k, -\boldsymbol{e}_j, \bar{\boldsymbol{b}}_i(\bar{\theta},\bar{\boldsymbol{w}})) \\ & = & -\lambda_i(\bar{\theta},\bar{\boldsymbol{w}}) \left(\boldsymbol{b}_i(\bar{\theta},\bar{\boldsymbol{w}})^T\boldsymbol{x} - \alpha(\boldsymbol{b}_i(\bar{\theta},\bar{\boldsymbol{w}}), C_k)\right) \left(\boldsymbol{e}_j^T\boldsymbol{x} - \alpha(\boldsymbol{e}_j, C_0)\right) \\ & & -\bar{\lambda}_i(\bar{\theta},\bar{\boldsymbol{w}}) \left(\bar{\boldsymbol{b}}_i(\bar{\theta},\bar{\boldsymbol{w}})^T\boldsymbol{x} - \alpha(\bar{\boldsymbol{b}}_i(\bar{\theta},\bar{\boldsymbol{w}}), C_k)\right) \left(-\boldsymbol{e}_j^T\boldsymbol{x} - \alpha(-\boldsymbol{e}_j, C_0)\right), \\ & & \\ g(\boldsymbol{x}) & = & \sum_{i=1}^n \sum_{j=1}^n \left(\bar{Q}^-_{ij} g^+_{ij}(\boldsymbol{x}) + \bar{Q}^+_{ij} g^-_{ij}(\boldsymbol{x})\right), \\ f(\boldsymbol{x}) & = & \sum_{i=1}^n \sum_{j=1}^n \left(\bar{Q}^-_{ij} f^+_{ij}(\boldsymbol{x}; \rho_k, \bar{\theta}, \bar{\boldsymbol{w}}) + \bar{Q}^+_{ij} f^-_{ij}(\boldsymbol{x}; \rho_k, \bar{\theta}, \bar{\boldsymbol{w}})\right). \end{array}\right\} \quad (3.24)$$

Here $f^+_{ij}(\cdot; \rho_k, \bar{\theta}, \bar{\boldsymbol{w}})$ and $f^-_{ij}(\boldsymbol{x}; \rho_k, \bar{\theta}, \bar{\boldsymbol{w}}))$ are defined as in (3.16). By construction, $g(\cdot) \in \widetilde{\mathcal{P}}^2(C_k)$. We also see that $g(\cdot)$ has the same Hessian matrix $\bar{\boldsymbol{Q}}^- - \bar{\boldsymbol{Q}}^+ = -\bar{\boldsymbol{Q}}$ as $f(\cdot)$ does. See Lemma 3.3.1. Hence the Hessian matrix of the quadratic function $g(\cdot) + qf(\cdot; \bar{\gamma}, \bar{\boldsymbol{q}}, \bar{\boldsymbol{Q}})$ vanishes. Thus we have shown (3.22).

(iii) Finally we observe that

$$\begin{array}{rcl} 0 & \geq & g(\bar{\boldsymbol{x}}) \\ & \geq & f(\bar{\boldsymbol{x}}) \text{ (since } \bar{\boldsymbol{x}} \in C_k \subset B(\boldsymbol{0}, \rho_k)) \\ & = & f(\delta\rho_k\bar{\boldsymbol{w}}) \\ & = & \displaystyle\sum_{i=1}^n \sum_{j=1}^n \left(\bar{Q}^+_{ij} f^+_{ij}(\delta\rho_k\bar{\boldsymbol{w}}; \rho_k, \bar{\theta}, \bar{\boldsymbol{w}}) + \bar{Q}^-_{ij} f^-_{ij}(\delta\rho_k\bar{\boldsymbol{w}}; \rho_k, \bar{\theta}, \bar{\boldsymbol{w}})\right) \\ & \geq & \displaystyle\sum_{i=1}^n \sum_{j=1}^n \left(\bar{Q}^+_{ij}\left(-2\rho_k \operatorname{diam}(C_0)(\bar{\kappa}+\bar{\theta})\right) + \bar{Q}^-_{ij}\left(-2\rho_k \operatorname{diam}(C_0)(\bar{\kappa}+\bar{\theta})\right)\right) \\ & & \text{(by (iv) of Lemma 3.3.1)} \\ & = & \displaystyle -2\rho_k \operatorname{diam}(C_0)(\bar{\kappa}+\bar{\theta}) \sum_{i=1}^n \sum_{j=1}^n \left(\bar{Q}^+_{ij} + \bar{Q}^-_{ij}\right) \end{array}$$

$$\begin{aligned}
&= -2\rho_k \operatorname{diam}(C_0)(\bar{\kappa}+\bar{\theta})\sum_{i=1}^{n}\sum_{j=1}^{n}|\bar{Q}_{ij}| \\
&\geq -2\eta \operatorname{diam}(C_0)(\bar{\kappa}+\bar{\theta})\ \nu_{\rm nl} \\
&\geq -\psi \quad \text{(by (3.19))}.
\end{aligned}$$

Therefore $g(\bar{\boldsymbol{x}}) + qf(\bar{\boldsymbol{x}};\bar{\gamma},\bar{\boldsymbol{q}},\bar{\boldsymbol{Q}}) > 0$. ∎

Lemma 3.3.3. *Suppose that* $\operatorname{diam}(F) > 0$. *Define*

$$\hat{k} = \begin{cases} 1 & \textit{if } \nu_{\rm nl} \leq \dfrac{\pi\ \psi}{32\ \eta \operatorname{diam}(C_0)}, \\[2ex] \left\lceil \left(\dfrac{4\eta\ \nu_{\rm nl} \operatorname{diam}(C_0)}{\psi}\right)^2 \ln \dfrac{2\eta}{\operatorname{diam}(F)} \right\rceil & \textit{otherwise.} \end{cases}$$

If $k \geq \hat{k}$, *then* $C_k \subset \text{c.relax}(F(\psi),\Xi)$.

Proof: For every $\boldsymbol{\xi} \in \Xi$ and $k = 0,1,2,\dots$, define

$$\rho_k(\boldsymbol{\xi}) = \max\left\{\rho\left(F(\psi),\boldsymbol{\xi}\right), \rho(C_k,\boldsymbol{\xi})\right\}.$$

It suffices to show that if $k \geq \hat{k}$ then

$$\rho_k(\boldsymbol{\xi}) \leq \rho\left(F(\psi),\boldsymbol{\xi}\right) \ \text{ for } \forall \boldsymbol{\xi} \in \Xi. \tag{3.25}$$

In fact, if (3.25) holds, then

$$C_k \subset \bigcap_{\boldsymbol{\xi}\in\Xi} B(\boldsymbol{\xi},\rho_k(\boldsymbol{\xi})) \subset \bigcap_{\boldsymbol{\xi}\in\Xi} B\left(\boldsymbol{\xi},\rho\left(F(\psi),\boldsymbol{\xi}\right)\right) = \text{c.relax}(F(\psi),\Xi).$$

By Lemma 3.3.2,

$$\rho_{k+1}(\boldsymbol{\xi}) \leq \max\left\{\rho\left(F(\psi),\boldsymbol{\xi}\right), \bar{\delta}\rho_k(\boldsymbol{\xi})\right\} \ (k = 0,1,2,\dots).$$

This implies that

$$\rho_k(\boldsymbol{\xi}) \leq \max\left\{\rho\left(F(\psi),\boldsymbol{\xi}\right), \bar{\delta}^k\rho_0(\boldsymbol{\xi})\right\} \quad \text{for } \forall\boldsymbol{\xi}\in\Xi \ \text{ and } \forall k = 0,1,2,\dots .$$

Hence, for each $\boldsymbol{\xi} \in \Xi$, if k satisfies the inequality

$$\bar{\delta}^k\rho_0(\boldsymbol{\xi}) \leq \rho\left(F(\psi),\boldsymbol{\xi}\right), \tag{3.26}$$

then $\rho_k(\boldsymbol{\xi}) \leq \rho\left(F(\psi),\boldsymbol{\xi}\right)$. When $\nu_{\rm nl} \leq \dfrac{\pi\ \psi}{32\ \eta \operatorname{diam}(C_0)}$, we see by (3.18) that $\bar{\delta} = 0$. Hence (3.25) holds for $k = 1$. Now assume that $\nu_{\rm nl} > \dfrac{\pi\ \psi}{32\ \eta \operatorname{diam}(C_0)}$. Then $\bar{\delta} > 0$ by (3.18). Since

$$F \subset F(\psi) \subset C_0 \subset \Xi,$$

we see that

$$\mathrm{diam}(F)/2 \leq \rho\left(F(\psi), \boldsymbol{\xi}\right) \leq \rho_0(\boldsymbol{\xi}) \leq \eta \ \text{ for } \forall \boldsymbol{\xi} \in \Xi,$$

Hence, if $\bar{\delta}^k \eta \leq \mathrm{diam}(F)/2$, or equivalently, if

$$k(-\ln \bar{\delta}) \geq \ln \frac{2\eta}{\mathrm{diam}(F)},$$

then (3.26) holds. We also see by the definition of $\bar{\delta}$ and (3.17) that

$$-\ln \bar{\delta} \ = \ -\ln\left(1 - \bar{\kappa}\bar{\theta}\right) \geq \bar{\kappa}\bar{\theta} \geq \left(\frac{\psi}{4\eta\ \nu_{\mathrm{nl}}\ \mathrm{diam}(C_0)}\right)^2 > 0.$$

Therefore if $k\left(\frac{\psi}{4\eta\ \nu_{\mathrm{nl}}\ \mathrm{diam}(C_0)}\right)^2 \geq \ln \frac{2\eta}{\mathrm{diam}(F)}$, then (3.26) holds. Consequently we have shown that if $k \geq \left(\frac{4\eta\ \nu_{\mathrm{nl}}\ \mathrm{diam}(C_0)}{\psi}\right)^2 \ln \frac{2\eta}{\mathrm{diam}(F)}$ then (3.25) holds. ∎

Theorem 3.3.4. *Assume (3.8) as in Corollary 3.1.2 and* $\mathrm{diam}(F) > 0$. *Define*

$$k^* = \begin{cases} 1 \quad \textit{if } \nu_{\mathrm{nl}} \leq \dfrac{\pi\ \epsilon^2}{256\ \nu_{\mathrm{lip}}\ \mathrm{diam}(C_0)^3}, \\ \left\lceil \left(\dfrac{32\ \nu_{\mathrm{lip}}\ \nu_{\mathrm{nl}}\ \mathrm{diam}(C_0)^3}{\epsilon^2}\right)^2 \ln \dfrac{8\ \nu_{\mathrm{lip}}\ \mathrm{diam}(C_0)^2}{\epsilon\ \mathrm{diam}(F)} \right\rceil \quad \textit{otherwise.} \end{cases}$$

If $k \geq k^*$, then $C_k \subset \mathrm{c.hull}(F(\epsilon))$.

Note that the bound k^* is proportional to the square of the nonlinearity ν_{nl} of $\mathcal{P}_F$, and also that $k^* = 1$ when any function in $\mathcal{P}_F$ is almost linear.

Proof of Theorem 3.3.4: Choose positive numbers ψ, τ' and a compact convex set $\Xi \subset R^n$ as in (3.9) of Corollary 3.1.2. Then $\mathrm{c.relax}(F(\psi), \Xi) \subset \mathrm{c.hull}(F(\epsilon))$. Let $\eta = \mathrm{diam}(\Xi) = 2\tau'$. Now, define $\hat{k}$ as in Lemma 3.3.3. By Lemma 3.3.3, if $k \geq \hat{k}$ then

$$C_k \subset \mathrm{c.relax}(F(\psi), \Xi) \subset \mathrm{c.hull}(F(\epsilon)).$$

On the other hand, we see that

$$\begin{aligned} \frac{\pi\ \psi}{32\ \eta\ \mathrm{diam}(C_0)} &= \frac{\pi\ \epsilon/2}{32\ (2\tau')\ \mathrm{diam}(C_0)} \\ &= \frac{\pi\ \epsilon}{128\ \mathrm{diam}(C_0)}\left(\frac{\epsilon}{2\ \nu_{\mathrm{lip}}\ \mathrm{diam}(C_0)^2}\right) \quad \text{(by (3.9))} \\ &= \frac{\pi\ \epsilon^2}{256\ \nu_{\mathrm{lip}}\ \mathrm{diam}(C_0)^3}. \end{aligned}$$

Hence the inequality $\nu_{\rm nl} \leq \dfrac{\pi\ \psi}{32\ \eta\ \mathrm{diam}(C_0)}$ appeared in the definition of $\hat{k}$ can be rewritten as

$$\nu_{\rm nl} \leq \frac{\pi\ \epsilon^2}{256\ \nu_{\rm lip}\ \mathrm{diam}(C_0)^3}.$$

We also see that

$$\begin{aligned}
&\left(\frac{4\eta\ \nu_{\rm nl}\ \mathrm{diam}(C_0)}{\psi}\right)^2 \ln\frac{2\eta}{\mathrm{diam}(F)} \\
&= \left(\frac{8\tau'\ \nu_{\rm nl}\ \mathrm{diam}(C_0)}{\epsilon/2}\right)^2 \ln\frac{4\tau'}{\mathrm{diam}(F)} \\
&= \left(\frac{16(2\ \nu_{\rm lip}\ \mathrm{diam}(C_0)^2)\ \nu_{\rm nl}\ \mathrm{diam}(C_0)}{\epsilon^2}\right)^2 \ln\frac{4(2\ \nu_{\rm lip}\ \mathrm{diam}(C_0)^2)}{\epsilon\ \mathrm{diam}(F)} \\
&= \left(\frac{32\ \nu_{\rm lip}\ \nu_{\rm nl}\ \mathrm{diam}(C_0)^3}{\epsilon^2}\right)^2 \ln\frac{8\ \nu_{\rm lip}\ \mathrm{diam}(C_0)^2}{\epsilon\ \mathrm{diam}(F)}
\end{aligned}$$

Therefore $k^* = \hat{k}$. ∎

3.4 Some Remarks

(A) In the Spherical-SDP Model, we have assumed that every ball with a given center $\boldsymbol{\xi} \in R^n$ that contains the kth iterate C_k is available. This assumption means that for a given $\boldsymbol{\xi} \in R^n$, we can solve a norm maximization problem

$$\max\ \|\boldsymbol{x} - \boldsymbol{\xi}\| \ \text{ subject to } \boldsymbol{x} \in C_k. \tag{3.27}$$

In fact, if we denote the maximum value of this problem by $\rho(C_k, \boldsymbol{\xi})$, then we can represent the set of all balls containing C_k as

$$\{B(\boldsymbol{\xi}, \tau) : \tau \geq \rho(C_k, \boldsymbol{\xi}),\ \boldsymbol{\xi} \in R^n\},$$

and $\mathcal{P}_k = \mathcal{P}^S(C_k)$ consists of the spherical quadratic functions $p(\cdot; \boldsymbol{\xi}, \tau)$ ($\boldsymbol{\xi} \in \boldsymbol{R}^n,\ \tau \geq \rho(C_k, \boldsymbol{\xi})$ such that

$$p(\boldsymbol{x}; \boldsymbol{\xi}, \tau) = (\boldsymbol{x} - \boldsymbol{\xi})^T(\boldsymbol{x} - \boldsymbol{\xi}) - \tau^2 \text{ for } \forall \boldsymbol{x} \in R^n.$$

We can also observe from the argument in Section 3.2 that among the spherical quadratic functions in $\mathcal{P}^S(C_k)$, only the supporting spherical quadratic functions $p(\cdot; \boldsymbol{\xi}, \rho(C_k, \boldsymbol{\xi}))$ ($\boldsymbol{\xi} \in R^n$) are relevant in constructing the next iterate $C_{k+1} = F(C_k, \mathcal{P}_F \cup \mathcal{P}^S(C_k))$.

In the Rank-2-SDP and the Rank-2-SILP Models, we have assumed that the maximum value $\alpha(\boldsymbol{d}, C_k)$ of the linear function $\boldsymbol{d}^T\boldsymbol{x}$ over $\boldsymbol{x} \in C_k$ is available for every $\boldsymbol{d} \in \overline{D}$. From the practical point of view, the latter assumption on the Rank-2-SDP and the Rank-2-SILP Models looks much weaker than the former assumption on the Spherical-SDP Model because a maximization of a linear function $\boldsymbol{d}^T\boldsymbol{x}$ over C_k is a convex program while the norm maximization problem (3.27) is a nonconvex program. But the latter assumption still requires to know the maximum value $\alpha(\boldsymbol{d}, C_k)$ for $\forall \boldsymbol{d}$ in the set $\overline{D}$ consisting of infinitely many directions, so that these models are not implementable yet. As we showed discretized-localized SCRMs in Chapter 2, Kojima and Tunçel [33] proposed discretization and localization techniques to implement the Rank-2-SDP and the Rank-2-SILP Models. Moreover, Takeda, Dai, Fukuda and Kojima [62] reported numerical results on practical versions of the Rank-2-SILP Model. We will show the practical versions in Chapter 4.

(B) In Section 3.3, we have focused on the Rank-2-SILP Model and diverted its complexity analysis to the Rank-2-SDP Model since the SDP relaxation is at least as tight as the semi-infinite LP relaxation. But this is somewhat loose. In particular, we should mention one important difference between the upper bounds required to attain an ϵ-convex-relaxation of F in Theorems 3.3.4 and 3.2.3: the upper bound in Theorems 3.3.4 depends on the nonlinearity $\nu_{\rm nl}(\mathcal{P}_F)$ of $\mathcal{P}_F$, while the upper bound in Theorem 3.2.3 depends on the nonconvexity $\nu_{\rm nc}(\mathcal{P}_F)$ but not on the nonlinearity $\nu_{\rm nl}$. This difference is critical when all the quadratic functions are convex but nonlinear. Here we state a complexity analysis on the Rank-2-SDP Model which leads to an upper bound depending on the nonconvexity $\nu_{\rm nc}$ of $\mathcal{P}_F$ but not on the nonlinearity $\nu_{\rm nl}$ of $\mathcal{P}_F$.

Define $\bar{\kappa}, \bar{\theta}$ and δ as

$$\begin{aligned}
\bar{\kappa} &= \begin{cases} \dfrac{8}{\pi} & \text{if } \nu_{\rm nc} \leq \dfrac{\pi\,\psi}{32 n\eta\,\mathrm{diam}(C_0)}, \\[2ex] \dfrac{\psi}{4n\eta\,\nu_{\rm nc}\,\mathrm{diam}(C_0)} & \text{otherwise,} \end{cases} \\
\bar{\theta} &= \begin{cases} \dfrac{\pi}{8} & \text{if } \nu_{\rm nc} \leq \dfrac{2\psi}{\pi n\eta\,\mathrm{diam}(C_0)}, \\[2ex] \dfrac{\psi}{4n\eta\,\nu_{\rm nc}\,\mathrm{diam}(C_0)} & \text{otherwise,} \end{cases} \\
\bar{\delta} &= 1 - \bar{\kappa}\bar{\theta}.
\end{aligned}$$

By definition, we see that $\bar{\kappa} \geq 0$, $1 \geq \bar{\delta} \geq 0$ and $(\bar{\kappa} + \bar{\theta})\nu_{\rm nc} \leq \dfrac{\psi}{2n\eta\,\mathrm{diam}(C_0)}$. It is easily seen that the assertion of Lemma 3.3.2 remains valid with the definition above. The proof is quite similar except replacing the quadratic function $g(\cdot)$ in (3.24) by

$$g(\boldsymbol{x}) = \nu_{\rm nc} \sum_{i=1}^{n} g_{ii}^{+}(\boldsymbol{x}) \text{ for } \forall \boldsymbol{x} \in R^n.$$

By using similar arguments as in Lemma 3.3.3, Theorem 3.3.4 and their proofs, we consequently obtain the following result: Assume (3.8) as in Corollary 3.1.2 and $\mathrm{diam}(F) > 0$.

Define

$$k^* = \begin{cases} 1 \quad \text{if } \nu_{\text{nc}} \leq \dfrac{\pi\,\epsilon^2}{256\,n\,\nu_{\text{lip}}\,\text{diam}(C_0)^3}, \\ \left\lceil \left(\dfrac{32\,n\,\nu_{\text{lip}}\,\nu_{\text{nc}}\,\text{diam}(C_0)^3}{\epsilon^2} \right)^2 \ln \dfrac{8\,\nu_{\text{lip}}\,\text{diam}(C_0)^2}{\epsilon\,\text{diam}(F)} \right\rceil \quad \text{otherwise.} \end{cases}$$

If $k \geq k^*$, then $C_k \subset \text{c.hull}(F(\epsilon))$.

Note that now the bound k^* is proportional to the square of the nonconvexity ν_{nc} of $\mathcal{P}_F$, and also that $k^* = 1$ when quadratic functions in $\mathcal{P}_F$ are almost convex.

Chapter 4

Practical Algorithms and Their Implementations

For any given small value $\epsilon > 0$, a discretized-localized SCRM (successive convex relaxation method) generates an ϵ-approximation of the maximum objective function value if it includes appropriately organized discretization and localization procedures. This is the claim of Theorem 2.3.3 in Chapter 2. However, the details of discretization and localization have not been studied, and the effect of specifically chosen discretization and localization procedures on the efficiency of a SCRM is not clear. Focusing on the discretized-localized version of the successive SDP (abbreviated by DLSSDP) relaxation method and the discretized-localized version of successive semi-infinite LP (abbreviated by DLSSILP) relaxation method, we propose implementable algorithms and study the behavior of the algorithms through computational experiments. Our numerical results demonstrate that the proposed DLSSILP algorithm generates relatively good upper bounds for the maximum objective function value for most test problems. It produces a better approximation when compared with one application of the SDP such as RLT-SDP (2.21), and semi-infinite LP relaxations such as RLT-LP (2.18). See Section 2.4.2 for the problems RLT-SDP and RLT-LP.

4.1 Practical Successive Convex Relaxation Methods

For arbitrarily given $\epsilon > 0$ and $\kappa > 0$, there exists $\delta > 0$ such that if we take a δ-net D_2 of $D(\boldsymbol{c}, \kappa)$, the discretized-localized SCRM generates an ϵ-approximation of the maximum objective function value for QOP (1.2) within finite iterations. This has been shown in Theorem 2.3.3. However, no specific result on the relations between $\epsilon > 0$, $\kappa > 0$ and $\delta > 0$ has been clarified. What we know is that in order to obtain an ϵ-approximation of the optimum value with small $\epsilon > 0$, we have to choose a δ-net with sufficiently small $\delta > 0$ for a fixed $\kappa > 0$. When κ is fixed, a δ-net with a smaller δ contains more vectors. Consequently, the number of linear constraints of each C_k and the number of SDPs or LPs to be solved will increase as ϵ becomes small.

The primary goal of this research is to study efficient implementations of the DLSSDP and DLSSILP relaxation methods. Concerning implementations, some issues have to be addressed. (a) How large neighborhood should we take in the objective direction $\boldsymbol{c}$ so that a better upper bound for the maximum objective function value can be achieved? (b) How many direction vectors should be included in a δ-net? (c) How do we distribute them?

In this section, we discuss these issues, and present practical versions of the DLSSDP and DLSSILP relaxation methods.

4.1.1 Choices of δ-nets

In their papers [32, 33], Kojima and Tunçel presented a certain δ-net consisting of finitely many direction vectors, which was used for the proof of the global convergence of their SCRMs. We use some of those vectors as a primary choice for our δ-net.

For $\forall\theta \in [0, \pi/2]$ and the coefficient vector $\boldsymbol{c}$ of the linear objective function of QOP (1.2), define

$$\left.\begin{array}{r} \boldsymbol{b}_i(\theta) \equiv (\boldsymbol{c}\cos\theta + \boldsymbol{e}_i \sin\theta)/\|\boldsymbol{c}\cos\theta + \boldsymbol{e}_i\sin\theta\|, \\ \bar{\boldsymbol{b}}_i(\theta) \equiv (\boldsymbol{c}\cos\theta - \boldsymbol{e}_i \sin\theta)/\|\boldsymbol{c}\cos\theta - \boldsymbol{e}_i\sin\theta\|, \\ (i = 1, 2, \dots, n) \end{array}\right\} \tag{4.1}$$

$$D(\theta) \equiv \{\boldsymbol{c},\ \boldsymbol{b}_i(\theta), \bar{\boldsymbol{b}}_i(\theta)\ (i = 1, 2, \dots, n)\}. \tag{4.2}$$

Then $\boldsymbol{b}_i(\theta), \bar{\boldsymbol{b}}_i(\theta) \in \overline{D}$ and $D(\theta) \subseteq \overline{D}$. The distance of any pair of vectors from $D(\theta)$ is determined by the value of θ. Define

$$\kappa \equiv \max\{\|\boldsymbol{d} - \boldsymbol{c}\| \mid \boldsymbol{d} \in D(\theta)\}$$

and

$$\delta \equiv \max_{\boldsymbol{d}\in D(\boldsymbol{c},\kappa)} \min_{\boldsymbol{d}'\in D(\theta)} \|\boldsymbol{d} - \boldsymbol{d}'\|.$$

Then the set $D(\theta)$ is a δ-net of $D(\boldsymbol{c}, \kappa)$. By changing the value of θ, we obtain different values of κ and δ. The discretized-localized procedure introduced in Section 2.3.2 takes the function-set $\mathcal{P}_k = \mathcal{P}^L(C_0, D_1) \cup \mathcal{P}^2(C_k, D_1, D_2)$ with $D_1 = \pm I, D_2 =$ (a δ-net of $D(\boldsymbol{c}, \kappa)$), *i.e.*,

$$\left.\begin{array}{rcl} \mathcal{P}_k & = & \{\ell sf(\boldsymbol{x}; C_0, \boldsymbol{d}_1) : \boldsymbol{d}_1 \in \pm I\} \ \cup\ \{r2sf(\boldsymbol{x}; C_k, \boldsymbol{d}_1, \boldsymbol{d}_2) : \boldsymbol{d}_1 \in \pm I,\ \boldsymbol{d}_2 \in D(\theta)\} \\ & = & \{\boldsymbol{d}_1^T\boldsymbol{x} - \alpha(C_0, \boldsymbol{d}_1) : \boldsymbol{d}_1 \in \pm I\} \cup \\ & & \quad \{-(\boldsymbol{d}_1^T\boldsymbol{x} - \alpha(C_0, \boldsymbol{d}_1))(\boldsymbol{d}_2^T\boldsymbol{x} - \alpha(C_k, \boldsymbol{d}_2)) : \boldsymbol{d}_1 \in \pm I,\ \boldsymbol{d}_2 \in D(\theta)\}, \end{array}\right\} \tag{4.3}$$

in Algorithm 2.2.4 (or Algorithm 2.2.5).

We include the vector $\boldsymbol{c}$ in the δ-net, since when the value $\zeta_k = \alpha(C_k, \boldsymbol{c})$ is updated, the corresponding rank-2 quadratic constraints in $\mathcal{P}_k$:

$$-(\boldsymbol{d}_1^T\boldsymbol{x} - \alpha(C_0, \boldsymbol{d}_1))(\boldsymbol{c}^T\boldsymbol{x} - \alpha(C_k, \boldsymbol{c})) \leq 0,\ \ (\forall \boldsymbol{d}_1 \in \pm I)$$

are expected to cut off some unnecessary area so that a better convex relaxation of F in a neighborhood of the objective direction $\boldsymbol{c}$ might be obtained. These constraints also force $\{\zeta_k \ (k = 0, 1, 2, ...,)\}$ to be a nonincreasing sequence, which correspond to upper bounds of QOP(1.2).

It should be noted that with κ and θ defined above, we are not guaranteed to obtain an ϵ-approximation of the maximum objective function value after a reasonable number of iterations. One needs to find a δ-net which is sufficiently fine so that the ϵ-approximation of the maximum objective function value can be obtained in a finite number of iterations. However, no previous information is available on how to make a refinement of the δ-nets. Since Algorithm 2.2.4 (or Algorithm 2.2.5) solves SDPs (or LPs) for many times, increasing the number of vectors in a δ-net will result in a substantial increase in the amount of work spent on solving SDPs (or LPs) due to the increase of the size of constraints as well as to the increase of the number of SDPs (or LPs) to be solved. More precisely, if we take $D(\theta)$ as D_2, the number of quadratic functions in $\mathcal{P}_k$ will be as many as $O(n^2)$. Although the refinement may lead to the decrease in iterations of the algorithm for obtaining a solution with a previously given precision, it still seems reasonable to use a δ-net with cardinality of $O(n)$.

A possible refinement of a δ-net $D(\theta)$ with cardinality $O(n)$ could be constructed as follows. For a given θ, take $\theta' = \theta/2$, and generate vectors $\boldsymbol{b}_i(\theta')$ and $\bar{\boldsymbol{b}}_i(\theta')$ $(i = 1, 2, \ldots, n)$ as above. Add those vectors to the set $D(\theta)$, then the new set is also a δ-net of $D(\boldsymbol{c}, \kappa)$. Actually it is a δ'-net of $D(\boldsymbol{c}, \kappa)$ for some $\delta' < \delta$. By our preliminary testing, the refinement did make an improvement on the quality of the approximation. However, the effect is relatively small and the computation time increases significantly. Therefore we shall use $D_2 \equiv D(\theta)$ as the δ-net of $D(\boldsymbol{c}, \kappa)$ hereafter for our implementation of the algorithm. Note that the normalization of the vectors $\boldsymbol{c}$, $\boldsymbol{b}_i(\theta)$, $\bar{\boldsymbol{b}}_i(\theta)$ is only for the convenience of the proof of Theorem 2.3.3. It is not necessary to do this when we implement the algorithm.

Recall that to construct $\mathcal{P}_k = \mathcal{P}^L(C_0, D_1) \cup \mathcal{P}^2(C_k, D_1, D_2)$ in Step 2 of Algorithm 2.2.4 (or Algorithm 2.2.5) with the current sets D_1 and D_2, we have to solve $(2n + 1)$ SDPs (or LPs). The number of supporting functions in the set $\mathcal{P}_k$ is $2n + (2n)(2n + 1)$, where the first term is the number of linear supporting functions of $\mathcal{P}^L(C_k, D_1)$. From our preliminary testing, it turns out that the values

$$\alpha(C_k, \boldsymbol{d}_1) = \max\{\boldsymbol{d}_1^T \boldsymbol{x} : \boldsymbol{x} \in C_k\} \ \ (\forall \boldsymbol{d}_1 \in \pm I) \tag{4.4}$$

change very little when C_k is updated. In addition to the statement of Remark 2.3.4, this is another reason to adopt $\mathcal{P}^L(C_0, D_1)$ instead of $\mathcal{P}^L(C_k, D_1)$, and use $\alpha(C_0, \boldsymbol{d}_1)$ $(\forall \boldsymbol{d}_1 \in D_1)$ and $\alpha(C_k, \boldsymbol{d}_2)$ $(\forall \boldsymbol{d}_2 \in D_2)$ for rank-2 quadratic supporting functions at each iteration. This will save a considerable amount of work on solving (4.4).

We also change the distribution of the vectors in the δ-net while running the algorithm. The vectors in the δ-net are replaced when the function value reaches a plateau. This is carried out by decreasing the value of θ and replacing old $\boldsymbol{b}_i(\theta), \bar{\boldsymbol{b}}_i(\theta)$ $(i = 1, 2, \ldots, n)$ in the δ-net with those corresponding to the new θ. But this procedure will be only repeated for K times till θ becomes a prescribed small value, since getting a better approximation by any further decrease of θ seems to be unlikely then. The rate of decreasing of θ will be determined by a sequence $\{\sigma_\ell\}_{\ell=0}^K$ $(1 = \sigma_0 > \sigma_1 > \cdots > \sigma_K)$, i.e., if the initial value of θ is θ_0, then at the ℓth $(\ell = 1, ..., K)$ replacement, θ is set to be equal to $\sigma_\ell \theta_0$.

4.1.2 Algorithms

With the δ-nets described in the previous subsection, we are ready to give a practical version of Algorithms DLSSDP and DLSSILP.

Algorithm 4.1.1. (DLSSDP relaxation method)

Input : an integer $K > 0$, real values $\theta_0 > 0,\ \epsilon_1 > 0,\ \epsilon_2 > 0$, and a sequence $\{\sigma_\ell\}_{\ell=0}^K$.
Output : a value $\bar{\zeta}$.

Step 0: Let $D_1 = \pm I$, $D_2 = D(\theta_0)$. Let $\zeta_{-1} = +\infty$.

Step 1: Compute $\alpha(C_0, \boldsymbol{d}) = \max\{\boldsymbol{d}^T\boldsymbol{x} : \boldsymbol{x} \in C_0\}$ $(\forall \boldsymbol{d} \in D_1 \cup D_2)$, let $k = 0$, $\ell = 0$.

Step 2: If $C_k = \emptyset$, let $\bar{\zeta} = -\infty$ and stop. Otherwise compute $\zeta_k = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in C_k\}$.
If $\ell = K$ and $\dfrac{\zeta_{k-1} - \zeta_k}{\max\{|\zeta_k|,\ \epsilon_2\}} \leq \sigma_K \epsilon_1$, let $\bar{\zeta} = \zeta_k$ and stop.

Step 3: If $\dfrac{\zeta_{k-1} - \zeta_k}{\max\{|\zeta_k|,\ \epsilon_2\}} \leq \sigma_\ell \epsilon_1$, then let $\ell = \ell + 1$, $\theta_\ell = \sigma_\ell \theta_0$, and replace the δ-net D_2 by the set of new vectors $D(\theta_\ell)$.

Step 4: Compute $\alpha(C_k, \boldsymbol{d}_2) = \max\{\boldsymbol{d}_2^T\boldsymbol{x} : \boldsymbol{x} \in C_k\}$ for $\forall \boldsymbol{d}_2 \in D_2$.

Step 5: Let $\mathcal{P}_k = \mathcal{P}^L(C_0, D_1) \cup \mathcal{P}^2(C_k, D_1, D_2)$.

Step 6: Let

$$\begin{aligned} C_{k+1} &= \widehat{F}(C_0, \mathcal{P}_F \cup \mathcal{P}_k) \\ &\equiv \left\{ \boldsymbol{x} \in C_0 : \begin{array}{l} \exists \boldsymbol{X} \in \mathcal{S}^n \text{ such that } \begin{pmatrix} 1 & \boldsymbol{x}^T \\ \boldsymbol{x} & \boldsymbol{X} \end{pmatrix} \in \mathcal{S}_+^{1+n} \text{ and} \\ \gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{Q} \bullet \boldsymbol{X} \leq 0 \ (\forall qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \in \mathcal{P}_F \cup \mathcal{P}_k) \end{array} \right\}. \end{aligned}$$

Step 7: Let $k = k + 1$, and go to Step 2.

Algorithm 4.1.2. (DLSSILP relaxation method)

Input : an integer $K > 0$, real values $\theta_0 > 0,\ \epsilon_1 > 0,\ \epsilon_2 > 0$, and a sequence $\{\sigma_\ell\}_{\ell=0}^K$.
Output : a value $\bar{\zeta}$.

Step 0, $1, 2, \ldots, 5$: The same as Steps $0, 1, 2 \ldots, 5$ of Algorithm 4.1.1, respectively.

Step 6: Let

$$\begin{array}{rcl} C_{k+1} & = & \widehat{F}^L(C_0, \mathcal{P}_F \cup \mathcal{P}_k) \\ & \equiv & \left\{ \boldsymbol{x} \in C_0 : \begin{array}{l} \exists \boldsymbol{X} \in \mathcal{S}^n \text{ such that} \\ \gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{Q} \bullet \boldsymbol{X} \leq 0 \ (\forall qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \in \mathcal{P}_F \cup \mathcal{P}_k) \end{array} \right\}. \end{array}$$

Step 7: The same as Step 7 of Algorithm 4.1.1.

Remark 4.1.3. In Step 4 of Algorithm 4.1.2, we solve $(2n+1)$ LPs with the same feasible region C_k. It is not necessary to solve each of them from scratch. After the first one is solved, solutions of subsequent LPs could be found by the standard reoptimization procedure developed for sensitivity study of LPs. This contributes a great deal to the efficiency of the implementation.

Remark 4.1.4. Algorithm 4.1.1 (or Algorithm 4.1.2) generates a sequence of convex relaxations $C_k \subseteq C_0$ of F $(k = 1, 2, \ldots)$ and a sequence of real numbers ζ_k $(k = 1, 2, \ldots)$ satisfying

$$\begin{array}{l} C_0 \supseteq C_k \supseteq C_{k+1} \supseteq F \ (k = 1, 2, \ldots), \\ \zeta_k \geq \zeta_{k+1} \geq \zeta^* \equiv \sup\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in F\} \ (k = 1, 2, \ldots). \end{array}$$

If in addition we take $D_2 = \overline{D}$, then C_k and ζ_k converge to the convex hull of F and ζ^*, respectively. See Section 2.3 for more details on the convergence of C_k and ζ_k.

4.1.3 Other Versions of the Algorithms

As we mentioned before, $\mathcal{P}_k$ is not a quadratic representation of C_k in general. To save the amount of work in each iteration, we could even use less quadratic functions; for example, we could take

$$\bar{\mathcal{P}}^2(C_k, D_1, D_2) = \left\{ \begin{array}{l} -(\pm\boldsymbol{e}_i^T\boldsymbol{x} - \alpha(C_0, \pm\boldsymbol{e}_i))(\boldsymbol{b}_j(\theta)^T\boldsymbol{x} - \alpha(C_k, \boldsymbol{b}_j(\theta))), \\ -(\pm\boldsymbol{e}_i^T\boldsymbol{x} - \alpha(C_0, \pm\boldsymbol{e}_i))(\bar{\boldsymbol{b}}_j(\theta)^T\boldsymbol{x} - \alpha(C_k, \bar{\boldsymbol{b}}_j(\theta))), \\ -(\pm\boldsymbol{e}_i^T\boldsymbol{x} - \alpha(C_0, \pm\boldsymbol{e}_i))(\boldsymbol{c}^T\boldsymbol{x} - \alpha(C_k, \boldsymbol{c})), \\ (1 \leq \forall i \leq \forall j \leq n) \end{array} \right\}.$$

The number of quadratic supporting functions in $\bar{\mathcal{P}}^2(C_k, D_1, D_2)$ is $2n(n+2)$, which is almost half of that in $\mathcal{P}^2(C_k, D_1, D_2)$. Replacing $\mathcal{P}^2(C_k, D_1, D_2)$ with $\bar{\mathcal{P}}^2(C_k, D_1, D_2)$ in $\mathcal{P}_k$, we obtain other versions of Algorithms DLSSDP and DLSSILP, which we call Algorithms DLSSDP' and DLSSILP', respectively. Algorithms DLSSDP' and DLSSILP' may be regarded as more practical versions, since they further overcome a rapid explosion in the number of convex programs to be solved at each iteration.

4.2 Computational Experiments on QOPs

In this section, we report computational results of Algorithms 4.1.1 and 4.1.2 for quadratic optimization test problems. We use CPLEX (Ver 6.5) as LP solver and SDPA (Ver. 5.00) [21] as SDP solver. The program was coded in C++ and was run on a Digital Alpha Workstation (CPU Alpha 2116-400 MHz with 512 MB of memory).

4.2.1 Test Problems

Our set of test problems for the computational study consists of six types of problems from literature.

(a) Minimization of a linear function over linear and quadratic constraints (LQCP);

(b) Minimization of a concave quadratic function with linear constraints (CQP);

(c) Bilinear programs (BLP);

(d) Mixed quadratic 0-1 integer programs (MQI);

(e) Sum of linear fractional programs (SLFP);

(f) Bilevel quadratic programs (BLevel).

The transformation from problems (a)-(c) to QOP (1.2) is straightforward. Concerning problems (d)-(f), we already presented their transformed quadratic problems in Section 2.1.

We suppose that those problems include lower and upper bounding constraints for each variable. Unless we impose box constraints on each variable, the Reformulation-Linealization Technique compared with our methods in numerical experiments, may fail due to the unboundness of variables. Therefore it is necessary to derive lower and upper bounds for all variables before we start the algorithms. We have preprocessed the test problems to obtain box constraints in the case they are not given by the original data.

If the original problem has a quadratic objective function $g(\boldsymbol{x})$, an artificial variable t is introduced to replace the quadratic function, and the original problem is transformed into QOP (1.2). The bounds for the new variable t is calculated as follows. Suppose that l_i and u_i such that $l_i \leq x_i \leq u_i$ $(i = 1, 2, \ldots, n)$ are given in the original problem. The quadratic function $g(\boldsymbol{x})$ contains nonlinear terms $x_i x_j$. Values of $\min\{l_i u_i, l_i u_j, l_j u_i, l_j u_j\}$ and $\max\{l_i u_i, l_i u_j, l_j u_i, l_j u_j\}$ can be used as lower and upper bounds for those nonlinear terms, respectively. Then lower and upper bounds for t can be generated by using bounds for the nonlinear terms and those for the original variables.

In problem (e), linear fractional terms appear in a objective function. The transformation of the problem (e) into QOP (1.2) uses an artificial variable, say t, to replace each linear fractional term, say $g(\boldsymbol{x})$. See Example 2.1.2 of Section 2.1. As an upper bound for $g(\boldsymbol{x})$, it is sufficient to take the ratio of the maximum value of the numerator to the minimum value of the denominator. Similarly, the ratio of the minimum value of the numerator to the maximum value of the denominator can serve as a lower bound for $g(\boldsymbol{x})$. Since both the denominator and the numerator are linear functions of the original variables, the maximum and the minimum values of the functions are not difficult to be calculated if box constraints of the original variables are given.

The only exception is problems of type (f). For those problems, since the computation of upper bounds for the Lagrange multipliers u_i ($i = 1, 2, \dots$) induced from the Karush-Kuhn-Tucker optimality condition is not trivial, we only use a sufficiently large value ($u_i = 1000$) as an upper bound for each u_i. In Chapter 5, we will focus on bilevel quadratic problems (f) and introduce a technique to avoid such *a priori* explicit bound $u_i = 1000$.

Here we denote QOP (1.2) as

$$\max \ \boldsymbol{c}^T \boldsymbol{x} \quad \text{subject to} \quad \boldsymbol{x} \in F, \tag{4.5}$$

where

$$F \ \equiv \ \left\{ \boldsymbol{x} \in R^n : \begin{array}{l} \gamma_i + 2\boldsymbol{q}_i^T \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{Q}_i \boldsymbol{x} \leq 0, \ \ i = 1, \dots, m \\ l_j \leq x_j \leq u_j, \ \ j = 1, \dots, n \end{array} \right\}. \tag{4.6}$$

Table 4.1 gives the test problems that we used in computational study.

4.2.2 Numerical Results

For the comparison, we implemented Algorithm DLSSILP with two different $\mathcal{P}_F$ and $\mathcal{P}_k$, and two other typical algorithms. The latter two algorithms apply the Reformulation-Linealization Technique (RLT) [55, 57] for a quadratic problem (4.5) and reformulate a new quadratic problem with additional constraints and variables. Then, take the SDP relaxation or the LP relxation for the reformulated problem. Concerning the reformulation with RLT, see problems RLT-SDP (2.21) and RLT-LP (2.18) of Section 2.4.2. We call these four algorithms DLSSILP, DLSSILP+RLT, RLT-SDP and RLT-LP, respectively.

Input-data $\mathcal{P}_F$ for QOP: The input data $\mathcal{P}_F$ for the three algorithms DLSSILP+RLT, RLT-SDP and RLT-LP are identical. In below, we designate the input data for Algorithm DLSSILP by DataDLSSILP, and for the other three algorithms by DataRLT. We construct the input data DataDLSSILP from (4.6) as

$$\mathcal{P}_F \equiv \left\{ \begin{array}{l} \gamma_i + 2\boldsymbol{q}_i^T \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{Q}_i \boldsymbol{x}, \ \ i = 1, \dots, m \\ l_j - x_j, \ \ -u_j + x_j, \ \ j = 1, \dots, n \end{array} \right\}.$$

The input data DataRLT, which is identical for DLSSILP+RLT, RLT-SDP and RLT-LP, is generated as follows. On the idea of RLT explained in Section 2.4.2, we generate quadratic constraints by taking pairwise products for all linear constraints of QOP (4.5).

Table 4.1: The test problems

Problem	Type	Source	n	m	#QP
LQCP1	(a)	[17]	3	3	1
LQCP2	(a)	[17]	7	7	3
LQCP3	(a)	[17]	8	6	3
LQCP4	(a)	[17]	6	7	7
LQCP5	(a)	[17]	9	10	4
LQCP6	(a)	[69]	7	7	3
LQCP7	(a)	[69]	2	4	2
LQCP8	(a)	[15, 61]	3	3	2
LQCP9	(a)	[15, 61]	10	11	4
CQP1	(b)	[17]	7	3	1
CQP2	(b)	[17]	11	6	1
CQP3	(b)	[17]	6	2	1
CQP4	(b)	[17]	21	11	1
CQP5	(b)	[17]	14	10	1
CQP6	(b)	[17]	7	6	1
CQP7	(b)	[17]	11	12	1
CQP8	(b)	[69]	5	7	1
BLP1	(c)	[48]	11	11	1
BLP2	(c)	[4]	11	14	1
BLP3	(c)	[4]	11	11	1
MQI1	(d)	[49]	5	11	7
MQI2	(d)	[25]	6	14	8
SLFP1	(e)	[16]	4	4	2
SLFP2	(e)	[16]	5	7	2
SLEP3	(e)	[16]	4	4	2
BLevel1	(f)	[70]	7	16	10
BLevel2	(f)	[70]	7	14	9
BLevel3	(f)	[70]	9	15	9
BLevel4	(f)	[70]	6	11	7
BLevel5	(f)	[70]	10	19	12

Legend : n and m denote the number of variables and the number of constraints (not including box constraints) in the transformed QOP, respectively. The notation #QP denotes the number of quadratic constraints. The second column gives the type of problems.

Table 4.2: Parameters for implementations

parameter	value
ϵ_1	0.001
ϵ_2	1.0
θ_0	90°
σ_0	1
σ_1	8/9
$\sigma_\ell(\ell \geq 2)$	$\sigma_1 * (\text{ratio}\,\theta)^{\ell-1}$
ratio θ	0.5
K	3

Those new and original quadratic constraints are considered as input data for the three algorithms.

An interesting observation is that if Algorithm DLSSILP is started with $\theta = 90°$, it actually generates quadratic constraints which correspond to those from the pairwise products of lower and upper bound constraints. More precisely, when $\theta = 90°$, vectors $\boldsymbol{b}_i, \bar{\boldsymbol{b}}_i$ $(i = 1, 2, \ldots, n)$ in the δ-net are corresponding to unit vectors $\boldsymbol{e}_i, -\boldsymbol{e}_i$ $(i = 1, 2, \ldots, n)$, and the values $\alpha(\boldsymbol{e}_i, C_0), -\alpha(-\boldsymbol{e}_i, C_0)$ $(i = 1, 2, \ldots, n)$ are upper and lower bounds for the variable x_i $(i = 1, 2, \ldots, n)$, respectively. The set $\mathcal{P}^2(C_k, D_1, D_2)$ constructed by Algorithm DLSSILP contains all quadratic functions from the pairwise products of lower and upper bound constraints. Therefore if the feasible region of the original problem is given only by those box constraints, then Sherali and Tuncbilek's method is identical to the first step of our algorithm starting with $\theta = 90°$.

Parameters: The parameters used by the algorithms are given in Table 4.2. We start the algorithm with $\theta = 90°$, since we observed from our preliminary numerical experiments that it generates better upper bounds for most of the test problems. When we replace a δ-net, we decrease θ by a factor of 8/9 at the first time, and then by a factor of 1/2 for the other replacements $(\ell \geq 2)$. This was also decided empirically.

Computational Results : We report numerical results for four algorithms in Table 4.4. The legends are listed in Table 4.3. Statistical results of relative errors of the four algorithms are summarized in Table 4.5. Each number in columns 2-5 of Table 4.5 indicates the number of cases where relative errors of maximum objective function values are within the corresponding range. The last column gives the number of cases successfully solved by each algorithm. Since the SDP relaxation could only solve 2/3 of the cases, we only compare the results of the first three algorithms. The failure of the SDP relaxation may be caused by the numerical instability due to the pairwise products of the linear constraints, since they could result in very large or very small coefficients at the same time.

Table 4.3: Legends

Problem	:	the type of problem;
DLSSILP	:	the algorithm using $\mathcal{P}^L(C_0, D_1) \cup \mathcal{P}^2(C_k, D_1, D_2)$ as $\mathcal{P}_k$ and DataDLSSILP as $\mathcal{P}_F$;
DLSSILP+RLT	:	the algorithm using $\mathcal{P}^L(C_0, D_1) \cup \mathcal{P}^2(C_k, D_1, D_2)$ as $\mathcal{P}_k$ and DataRLT as $\mathcal{P}_F$;
RLT-LP	:	the LP relaxation algorithm with input DataRLT;
RLT-SDP	:	the SDP relaxation algorithm with input DataRLT;
R.error	:	the relative error of a solution, i.e., $\dfrac{\lvert\bar{\zeta} - \zeta^*\rvert}{\max\{\lvert\zeta^*\rvert,\ \epsilon_2\}}$, where ζ^* is the maximum or best known objective function value, and $\bar{\zeta}$ the upper bound obtained by each algorithm for ζ^*;
cpu	:	the cpu time in seconds;
iter.	:	the number of iterations (Steps 2- 7) of Algorithm 4.1.2.

Table 4.5 shows that on average, Algorithm DLSSILP+RLT performed best in generating good upper bounds for the test problems, and Algorithm RLT-LP generated less good ones. We also observed that Algorithm DLSSILP+RLT reported 24 cases with relative errors less than 0.01, and Algorithm DLSSILP reported 21 cases. We could conclude that Algorithm DLSSILP is also competent to obtain good upper bounds for the maximum objective function value. From the aspect of computing time, Algorithm RLT-LP needs much less time, while Algorithm DLSSILP+RLT is more time-consuming on average. However, there are a few exceptions. For example, comparing with Algorithm DLSSILP, Algorithm DLSSILP+RLT produced a better solution for problem LQCP3 in less time. Another observation is that both of SSILP relaxation algorithms seem to be more capable of solving difficult types of problems such as types (e) and (f) than Algorithm RLT-LP does.

Behavior of the algorithms : Experiments were conducted in order to see how the other factors affect the behavior of the algorithms. The following three factors were considered : (i) different quadratic representation $\mathcal{P}_k$'s of C_k; (ii) δ-nets without replacement; and (iii) the number of replacements of the δ-net. We picked up one specific problem with relatively large numbers of variables and constraints from each group of the problems in Table 4.1 for the analysis.

(i) As we discussed in Section 4.1.3, the set $\bar{\mathcal{P}}^2(C_k, D_1, D_2)$ could be chosen as an alternative to the set $\mathcal{P}^2(C_k, D_1, D_2)$ to represent $\mathcal{P}_k$ in the algorithm. Algorithm DLSSILP' uses $\bar{\mathcal{P}}^2(C_k, D_1, D_2)$ instead of $\mathcal{P}^2(C_k, D_1, D_2)$ in the representation of $\mathcal{P}_k$, and all parameters were fixed as the same values as in Algorithm DLSSILP. The numerical results of these two algorithms are presented in Tables 4.6-4.7. Algorithm DLSSILP is slightly better than Algorithm DLSSILP' in terms of generating good upper bounds. But the former takes more time. There exists almost no difference between the algorithms DLSSILP+RLT and

Table 4.4: Computational results of four algorithms

Problem	DLSSILP			DLSSILP+RLT			RLT-LP		RLT-SDP	
	R.error	cpu	iter.	R.error	cpu	iter.	R.error	cpu	R.error	cpu
LQCP1	4.68e-4	0.53	46	3.70e-4	0.43	30	4.23e-1	0.00	4.23e-1	0.48
LQCP2	1.02e-4	0.38	7	0.00e+0	0.52	5	1.83e-16	0.02	7.93e-6	16.93
LQCP3	2.94e-1	165.78	289	2.79e-2	35.18	51	5.20e-1	0.03	———	——
LQCP4	1.07e-3	0.25	6	9.70e-4	0.38	6	4.42e-3	0.02	———	——
LQCP5	1.56e-15	3.87	6	1.28e-15	7.13	6	2.50e-1	0.03	———	——
LQCP6	1.02e-4	0.47	8	4.77e-15	0.35	5	3.67e-16	0.02	9.39e-5	17.08
LQCP7	2.08e-2	0.13	27	1.35e-2	0.17	25	6.07e-2	0.00	7.56e-5	0.20
LQCP8	3.64e-3	0.25	35	1.93e-3	0.30	28	5.79e-1	0.00	5.79e-1	0.35
LQCP9	1.28e-15	3.85	7	1.85e-15	31.83	6	2.50e-1	0.05	———	——
CQP1	9.34e-16	0.20	5	0.00e+0	0.33	5	0.00e+0	0.03	1.66e-6	7.02
CQP2	1.18e-3	9.22	20	5.47e-15	5.58	6	2.12e-2	0.20	2.18e-2	80.07
CQP3	4.05e-2	2.63	37	2.06e-2	2.45	28	6.82e-2	0.02	6.82e-2	8.85
CQP4	4.26e-2	352.73	63	———	——	–	2.08e-4	6.40	2.80e-3	1931.52
CQP5	0.00e+0	3.57	5	5.92e-16	2.97	5	2.37e-16	0.15	3.26e-6	243.65
CQP6	3.23e-16	0.33	5	1.61e-16	1.18	5	0.00e+0	0.05	2.00e-5	15.43
CQP7	4.03e-4	1.42	7	1.18e-7	9.52	5	1.18e-7	0.53	1.57e-5	199.42
CQP8	2.73e-16	0.15	6	0.00e+0	0.27	5	0.00e+0	0.02	2.96e-6	7.30
BLP1	8.25e-7	1.05	6	4.18e-6	4.38	5	4.18e-6	0.15	1.72e-4	178.22
BLP2	1.09e-3	9.08	15	5.14e-6	74.02	5	5.14e-6	1.03	1.24e-1	240.27
BLP3	6.38e-6	1.07	6	1.37e-6	4.17	5	1.37e-6	0.15	1.25e-4	142.53
MQI1	2.02e-16	0.11	5	0.00e+0	0.18	5	2.02-16	0.03	1.66e-6	4.15
MQI2	6.92e-2	10.17	116	6.34e-16	1.92	8	3.33e-1	0.02	6.74e-6	13.85
SLFP1	1.78e-16	0.18	9	3.04e-8	0.23	9	2.92e-1	0.00	———	——
SLFP2	9.61e-4	0.57	16	9.76e-5	1.30	14	3.54e-1	0.02	3.54e-1	4.68
SLFP3	3.16e-2	1.23	44	2.96e-2	1.48	40	2.58e-1	0.00	2.58e-1	1.08
BLevel1	5.74e-2	73.43	509	3.57e-8	0.73	5	3.57e-8	0.03	———	——
BLevel2	2.95e-1	58.52	382	3.37e-3	23.95	72	7.06e-1	0.03	———	——
BLevel3	2.23e-4	1.78	15	1.65e-12	2.72	7	2.22e-1	0.05	———	——
BLevel4	1.65e-2	0.90	27	1.38e-2	1.27	22	1.00e+0	0.02	———	——
BLevel5	6.76e-3	52.10	80	1.59e-4	32.65	23	1.16e-1	0.05	———	——

The empty entries show that the algorithm could not obtain solutions.

Table 4.5: Statistical results of relative errors

Algorithm	range of relative errors				#. cases solved
	$[0, 0.01)$	$[0.01, 0.1)$	$[0.1, 1)$	$[1, +\infty)$	
DLSSILP	21	7	2	0	30
DLSSILP+RLT	24	5	0	0	29
RLT-LP	14	3	12	1	30
RLT-SDP	13	2	5	0	20

DLSSILP'+RLT as far as the quality of the solution is concerned. But the former consumes more time in most cases.

(ii) Using Problem CQP3 as an example, we see the behavior of some variants of Algorithm DLSSILP where each δ-net was fixed (*i.e.*, θ of the set $D(\theta)$ defined by (4.2) was fixed) throughout the execution of the algorithm. The values of θ for the fixed δ-nets were set as $10°$, $40°$ and $80°$, respectively. In Figure 4.1, CQP3-10, CQP3-40 and CQP3-80 are corresponding to each such case, respectively. CQP3-replace indicates the result where the δ-net was replaced. It is obvious from the figure that the replacement of the δ-net is effective to reduce the relative error quickly.

(iii) Table 4.8 shows the results of Algorithm DLSSILP when different numbers of replacements, e.g. $K = 3,\ 5$, were selected. It seems that we could obtain relatively good upper bounds when $K = 3$. This is also confirmed from Figure 4.1, since the drastic decrease of the relative error occurs at an early stage of the execution of the algorithm.

In this chapter, we presented a practically implementable version of the discretized-localized SSILP relaxation method [33] proposed by Kojima and Tunçel. We studied the behavior of the method through computational experiments. It could generate relatively good upper bounds for the maximum objective function value for most of the test problems.

Several issues are left for further study. The most challenging problem is to give theoretical analysis for the convergence rate of the algorithm. As a practical implementation, attempting different types of δ-nets by exploring special structures of problems is important not only in making the method more efficient but also in obtaining information for the theoretical study. Moreover, suitable data structures which enable to solve problems of large size should be developed.

Table 4.6: Algorithms DLSSILP and DLSSILP'

Problem	DLSSILP			DLSSILP'		
	R.error	cpu	iter.	R.error	cpu	iter.
LQCP3	2.94e-1	165.78	289	3.37e-1	115.65	436
CQP4	4.26e-2	352.73	63	1.14e-1	61.45	33
BLP2	1.09e-3	9.08	15	4.92e-3	4.70	19
MQI2	6.92e-2	10.17	116	1.36e-1	4.20	99
SLFP3	3.16e-2	1.23	44	3.92e-2	0.60	34
BLevel2	2.95e-1	58.52	382	8.82e-1	1.43	25

Table 4.7: Algorithms DLSSILP+RLT and DLSSILP'+RLT

Problem	DLSSILP+RLT			DLSSILP'+RLT		
	R.error	cpu	iter.	R.error	cpu	iter.
LQCP3	2.78e-2	35.18	51	3.59e-2	22.20	51
CQP4	———	——	–	———	——	–
BLP2	5.14e-6	74.02	5	5.14e-6	39.83	5
MQI2	6.34e-16	1.92	8	2.54e-16	1.32	9
SLFP3	2.96e-2	1.48	40	4.36e-2	0.73	29
BLevel2	3.37e-3	23.95	72	6.76e-2	28.75	77

The empty entries show that the algorithm could not obtain solutions.

Figure 4.1: Behavior of Algorithm DLSSILP with fixed and replaced δ-nets for problem CQP3

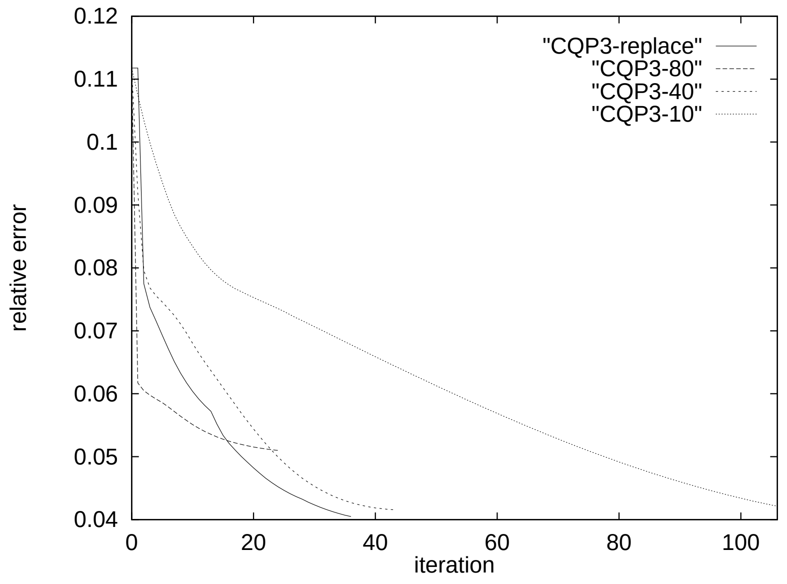


Table 4.8: Different numbers of replacements for the δ-net

Problem	DLSSILP (K= 3)			DLSSILP (K= 5)		
	R.error	cpu	iter.	R.error	cpu	iter.
LQCP3	2.94e-1	166.72	289	2.67e-1	236.65	363
CQP3	4.05e-2	2.85	37	3.84e-2	4.00	53
CQP4	4.26e-2	343.35	63	3.99e-2	551.02	91
BLP2	1.09e-3	8.97	15	1.02e-3	11.10	17
MQI2	6.92e-2	10.28	116	5.02e-2	15.18	181
SLFP3	3.30e-2	1.37	45	1.88e-2	2.42	83
BLevel2	2.95e-1	57.00	382	2.95e-1	57.20	384

Chapter 5

An Application of Successive Convex Relaxation Methods to Bilevel Quadratic Optimization Problems

So far we have handled the most general class of QOPs (quadratic optimization problems) of the form (1.2). In this chapter, we focus on QOPs with additional special structure, induced from some class of bilevel programming (abbreviated by BP), and propose special versions of SCRMs (successive convex relaxation methods) to such QOPs.

BPs belong to a class of nonconvex global optimization problems. It arises where decisions are made in a two-level hierarchical order and each decision maker has no direct control or influence upon the decision of the other, but actions taken by one decision maker affect returns from the other. Such problems can be formulated as two levels of nested mathematical programs as follows:

$$\left.\begin{array}{ll} \max\limits_{\boldsymbol{x}} & F(\boldsymbol{x},\boldsymbol{y}) \\ \text{subject to} & \boldsymbol{y} \in \operatorname*{argmin}\limits_{\boldsymbol{y}}\{G(\boldsymbol{x},\boldsymbol{y}) : \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) \leq 0\}, \\ & \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}) \leq 0, \\ & \boldsymbol{w} = \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} \in G_0, \end{array}\right\} \tag{5.1}$$

where $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) : R^n \rightarrow R^{m_1}$, $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}) : R^n \rightarrow R^{m_2}$, $G(\boldsymbol{x},\boldsymbol{y}) : R^n \rightarrow R$, $F(\boldsymbol{x},\boldsymbol{y}) : R^n \rightarrow R$, and G_0 denotes a nonempty compact polyhedral subset of R^n. Given an action $\boldsymbol{x}$ at the upper level, the lower level decision maker returns a minimizer $\boldsymbol{y}(\boldsymbol{x})$ of $G(\boldsymbol{x},\boldsymbol{y})$ subject to the constraints $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) \leq \boldsymbol{0}$ to the upper level. As a whole, the upper level decision maker needs to maximize his objective function $F(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{x}))$ subject to the constraints $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{x})) \leq \boldsymbol{0}$ and $(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{x})) \in G_0$.

Applications of BPs are numerous; for example, (i) hierarchical decision making policy problems in mixed economies, where policy makers at the top level influence the decisions of private individuals and companies, (ii) network facility location with delivered price competition, (iii) the Portland Aluminium Smelter in Victoria, Australia [41], aiming to maximize the aluminium production while minimizing the main cost associated with the production. See Vicente and Calamai [68] for a recent comprehensive review of the literature.

We call a BP lower-convex if the lower level objective function $G(\boldsymbol{x}, \boldsymbol{y})$ and constraint functions $g_i(\boldsymbol{x}, \boldsymbol{y})$ $(i = 1, 2, \ldots, m_1)$ are all convex in $\boldsymbol{y}$ for each fixed value of $\boldsymbol{x}$. Among the BPs, lower-convex BPs have received most of the attention in the literature. The advantage of dealing with lower-convex BPs is that under an appropriate constraint qualification, the lower level problem can be replaced by its Karush-Kuhn-Tucker (KKT) optimality (or equilibrium) condition to obtain an equivalent (one-level) mathematical program, a special case of the Mathematical Program with Equilibrium Constraints (MPEC) which has been studied intensively in recent years. See Luo, Pang and Ralph [36] for more details about MPEC.

There are three important classes of lower-convex BPs, namely:

(i) linear BPs, where all functions involved are affine,

(ii) linear-quadratic BPs, where $G(\boldsymbol{x}, \boldsymbol{y})$ is convex quadratic and all remaining functions are affine,

(iii) quadratic BPs, which differ from linear-quadratic BPs in that $F(\boldsymbol{x}, \boldsymbol{y})$ can also be a quadratic function.

We propose new techniques for solving a more general class of BPs than the class (iii) by allowing that some of $g_i(\boldsymbol{x}, \boldsymbol{y})$ $(i = 1, 2, \ldots, m_1)$ are convex quadratic functions and some of $f_j(\boldsymbol{x}, \boldsymbol{y})$ $(j = 1, 2, \ldots, m_2)$ are (not necessarily convex) quadratic functions. We could further weaken the assumption of quadratic functions to a wide class of nonlinear functions according to the technique proposed by [30], but for simplicity of discussions, we will restrict ourselves to bilevel quadratic optimization problems (abbreviated by BQOPs) where all functions involved in BPs are linear or quadratic. In these cases, the application of the KKT optimality condition to the lower level problem results in a one-level nonconvex QOP including complementarity constraints. We further transform the QOP into a bounded constraint QOP having a linear objective function in order to utilize SCRMs.

We adapt the discretized-localized SSDP (successive semidefinite programming) relaxation method and the discretized-localized SSILP (successive semi-infinite LP) relaxation method to BQOPs, taking full advantage of a special structure of the transformed QOPs. The main purpose of this chapter is to attain a relatively tight upper bound for the optimum objective function value of the BQOP with reduced computational expense. In the next section, we formally define a BQOP, and present its reformulation via the KKT optimality condition. In Section 5.2, we introduce two types of techniques exploiting the special structure of the QOP transformed from a BQOP, and illustrate the techniques using a small example. In Section 5.3, we report some numerical results, which show the effectiveness of our proposed techniques.

5.1 Transformation into a QOP Formulation

Consider a bilevel quadratic optimization problem (BQOP) having a linear objective function at the upper level:

$$\left.\begin{array}{ll} \max\limits_{\boldsymbol{u}} & \boldsymbol{c}_1^T \boldsymbol{v} + \boldsymbol{c}_2^T \boldsymbol{u} \\ \text{subject to} & \boldsymbol{v} \in \operatorname*{argmin}\limits_{\boldsymbol{v}} \{g_0(\boldsymbol{v},\boldsymbol{u}) : g_i(\boldsymbol{v},\boldsymbol{u}) \leq 0 \ (i=1,\ldots,m_1)\}, \\ & f_j(\boldsymbol{v},\boldsymbol{u}) \leq 0 \ (j=1,\ldots,m_2), \quad \boldsymbol{w} = \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{u} \end{pmatrix} \in G_0. \end{array}\right\} \tag{5.2}$$

where

$$\begin{array}{rcl} \boldsymbol{v} \in R^{n_1} & : & \text{the lower level variable vector,} \\ \boldsymbol{u} \in R^{n_2} & : & \text{the upper level variable vector,} \\ n & \equiv & n_1 + n_2, \\ G_0 & : & \text{a nonempty compact polyhedral subset of } R^n, \\ \boldsymbol{c}_1 & \in & R^{n_1}, \ \boldsymbol{c}_2 \in R^{n_2}, \\ g_i(\cdot,\cdot), f_j(\cdot,\cdot) \in \mathcal{Q} & : & \text{a quadratic function on } R^n \\ & & (i=0,1,\ldots,m_1, \ j=1,2,\ldots,m_2), \\ \boldsymbol{g}(\cdot,\cdot) & = & \begin{pmatrix} g_1(\cdot,\cdot) \\ \vdots \\ g_{m_1}(\cdot,\cdot) \end{pmatrix} : R^n \to R^{m_1}, \\ \boldsymbol{f}(\cdot,\cdot) & = & \begin{pmatrix} f_1(\cdot,\cdot) \\ \vdots \\ f_{m_2}(\cdot,\cdot) \end{pmatrix} : R^n \to R^{m_2}. \end{array}$$

If a given BQOP has a quadratic objective function at the upper level, we could transform the problem into the form (5.2) above by adding a constraint "the objective function$-t=0$" and replacing the objective function by t. We impose the following condition on the BQOP (5.2).

Condition 5.1.1.

(i) The quadratic function $g_i(\cdot,\boldsymbol{u})$ is convex on $R^{n_1} \times \{\boldsymbol{u}\}$ for fixed $\boldsymbol{u} \in R^{n_2}$ $(i=0,\ldots,m_1)$.

(ii) If

$$\sum_{i=1}^{m_1} \lambda_i \nabla_v g_i(\boldsymbol{v},\boldsymbol{u}) = \boldsymbol{0}, \ g_i(\boldsymbol{v},\boldsymbol{u}) \leq 0, \ \lambda_i g_i(\boldsymbol{v},\boldsymbol{u}) = 0, \ 0 \leq \lambda_i \ (i=1,\ldots,m_1),$$
$$f_j(\boldsymbol{v},\boldsymbol{u}) \leq 0 \ (j=1,\ldots,m_2), \ \boldsymbol{w} = \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{u} \end{pmatrix} \in G_0,$$

then $\boldsymbol{\lambda} = (\lambda_1,\ldots,\lambda_{m_1})^T = \boldsymbol{0}$ (A strong version of the Mangasarian Fromovitz constraint qualification).

Applying the KKT optimality condition to the lower level convex problem of the BQOP (5.2) under Condition 5.1.1, we can reduce the constraint

$$\boldsymbol{v} \in \operatorname*{argmin}_{\boldsymbol{v}} \{g_0(\boldsymbol{v}, \boldsymbol{u}) : g_i(\boldsymbol{v}, \boldsymbol{u}) \leq 0 \ (i = 1, \ldots, m_1)\}$$

to the constraint

$$\begin{array}{l} \nabla_{\!v} g_0(\boldsymbol{v}, \boldsymbol{u}) + \displaystyle\sum_{i=1}^{m_1} \lambda_i \nabla_{\!v} g_i(\boldsymbol{v}, \boldsymbol{u}) = \boldsymbol{0}, \\ g_i(\boldsymbol{v}, \boldsymbol{u}) \leq 0, \ \lambda_i g_i(\boldsymbol{v}, \boldsymbol{u}) = 0, \ 0 \leq \lambda_i \ (i = 1, \ldots, m_1), \end{array}$$

where λ_i denotes the Lagrange multiplier of the ith constraint $g_i(\boldsymbol{v}, \boldsymbol{u}) \leq 0 \ (i = 1, \ldots, m_1)$. Thus, introducing slack variables $s_i \geq 0 \ (i = 1, \ldots, m_1)$ for the quadratic inequality constraints $g_i(\boldsymbol{v}, \boldsymbol{u}) \leq 0 \ (i = 1, \ldots, m_1)$, we can rewrite the BQOP (5.2) as a QOP

$$\left.\begin{array}{ll} \displaystyle\max_{\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{w}} & \boldsymbol{c}_1^T \boldsymbol{v} + \boldsymbol{c}_2^T \boldsymbol{u} \\ \text{subject to} & \nabla_{\!v} g_0(\boldsymbol{v}, \boldsymbol{u}) + \displaystyle\sum_{i=1}^{m_1} \lambda_i \nabla_{\!v} g_i(\boldsymbol{v}, \boldsymbol{u}) = \boldsymbol{0}, \\ & g_i(\boldsymbol{v}, \boldsymbol{u}) + s_i = 0, \ 0 \leq s_i \leq \bar{s}_i \ (i = 1, \ldots, m_1), \\ & \lambda_i s_i = 0, \ 0 \leq \lambda_i \ (i = 1, \ldots, m_1), \\ & f_j(\boldsymbol{v}, \boldsymbol{u}) \leq 0 \ (j = 1, \ldots, m_2), \quad \boldsymbol{w} = \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{u} \end{pmatrix} \in G_0. \end{array}\right\} \tag{5.3}$$

Here $\bar{s}_i \ (i = 1, 2, \ldots, m_1)$ denotes a positive number such that

$$\bar{s}_i \geq \max \{s_i : g_i(\boldsymbol{v}, \boldsymbol{u}) + s_i = 0, \ s_i \geq 0, \ \boldsymbol{w} \in G_0\}. \tag{5.4}$$

Since G_0 is a compact polyhedral set, such a finite positive number $\bar{s}_i$ always exists. If some inequality constraint $g_i(\boldsymbol{v}, \boldsymbol{u}) \leq 0 \ (i \in \{1, \ldots, m_1\})$ is linear, it is not necessary to introduce the slack variable s_i since the complementarity constraint $\lambda_i g_i(\boldsymbol{v}, \boldsymbol{u}) = 0$ itself is a quadratic equality constraint. We also know from (ii) of Condition 5.1.1 that the Lagrange multipliers $\lambda_i \ (i = 1, 2, \ldots, m_1)$ involved in the QOP (5.3) are bounded. So we assume to know a sufficiently large number M which bounds the Lagrange multipliers. The bound is necessary because the SSDP and SSILP relaxation methods require initial compact polyhedral relaxation C_0 for the feasible region of the problem to be solved. We must say, however, that the tightness of upper bounds for the Lagrange multipliers deeply affects the performance of the SSDP and SSILP relaxation methods. In Section 5.2, we will present an additional technique to confine the Lagrange multipliers into a bounded convex set without setting *a priori* explicit bound M for them.

To meet the formulation (1.2) for the SSDP and SSILP relaxations, we now rewrite the QOP (5.3) as

$$\max \ \boldsymbol{c}^T \boldsymbol{x} \text{ subject to } \boldsymbol{x} \in F, \tag{5.5}$$

where

$$
\begin{aligned}
\boldsymbol{x} &= \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{s} \\ \boldsymbol{w} \end{pmatrix} \in R^{2m_1+n}, \ \boldsymbol{c} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \bar{\boldsymbol{c}} \end{pmatrix} \in R^{2m_1+n}, \ \bar{\boldsymbol{c}} = \begin{pmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \end{pmatrix} \in R^n, \\
F &= \{\boldsymbol{x} \in C_0 : p(\boldsymbol{x}) \leq 0 \ \ (\forall p(\cdot) \in \mathcal{P}_F)\}, \\
\mathcal{P}_F &= \left\{ \begin{array}{l} q_i(\cdot) \ (i = 1, \ldots, m_1 + n_1 + m_1 + m_2), \\ -q_j(\cdot) \ (j = 1, \ldots, m_1 + n_1 + m_1) \end{array} \right\}, \\
\boldsymbol{q}(\boldsymbol{x}) &= \begin{pmatrix} q_1(\boldsymbol{x}) \\ \vdots \\ q_{m_1}(\boldsymbol{x}) \\ \vdots \\ q_{m_1+n_1}(\boldsymbol{x}) \\ q_{m_1+n_1+1}(\boldsymbol{x}) \\ \vdots \\ q_{m_1+n_1+m_1}(\boldsymbol{x}) \\ q_{m_1+n_1+m_1+1}(\boldsymbol{x}) \\ \vdots \\ q_{m_1+n_1+m_1+m_2}(\boldsymbol{x}) \end{pmatrix} = \begin{pmatrix} \lambda_1 s_1 \\ \vdots \\ \lambda_{m_1} s_{m_1} \\ \nabla_v g_0(\boldsymbol{v}, \boldsymbol{u}) + \sum_{i=1}^{m_1} \lambda_i \nabla_v g_i(\boldsymbol{v}, \boldsymbol{u}) \\ g_1(\boldsymbol{v}, \boldsymbol{u}) + s_1 \\ \vdots \\ g_{m_1}(\boldsymbol{v}, \boldsymbol{u}) + s_{m_1} \\ f_1(\boldsymbol{v}, \boldsymbol{u}) \\ \vdots \\ f_{m_2}(\boldsymbol{v}, \boldsymbol{u}) \end{pmatrix} \quad \text{for } \forall \boldsymbol{x} = \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{s} \\ \boldsymbol{w} \end{pmatrix}, \\
C_0 &= \left\{ \boldsymbol{x} = \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{s} \\ \boldsymbol{w} \end{pmatrix} \in R^{2m_1+n} : \begin{array}{l} 0 \leq \lambda_i \leq M \ (i = 1, \ldots, m_1), \\ 0 \leq s_i \leq \bar{s}_i \ (i = 1, \ldots, m_1), \\ \boldsymbol{w} \in G_0 \end{array} \right\}.
\end{aligned}
$$

Note that $\mathcal{P}_F$ is a set of quadratic functions on R^{2m_1+n}. In particular, it includes nonconvex quadratic functions induced from the complementarity constraints.

5.2 Special Techniques for BQOPs

Here we consider the discretized-localized SSDP and SSILP relaxation methods given in Section 2.3.2 and their practical algorithms (Algorithms 4.1.1 and 4.1.2). For the discretized-localized SSILP, its numerical results on the general quadratic test problems were already reported in Section 4.2.2. In this section, regarding to a special quadratic problem QOP (5.5), we will present two types of techniques to enhance the efficiency of Algorithms 4.1.1 and 4.1.2; one technique reformulates the special QOP (5.5) into an equivalent scaled problem with explicit bounds for the Lagrange multipliers, and the other technique tightens upper or lower bounds for each complementary pair of variables.

5.2.1 Scaling Lagrange Multipliers

In order to apply Algorithms 4.1.1 and 4.1.2 to the formulation (5.5), we need to give an initial compact polyhedral relaxation C_0 of the feasible region F. Among the variable vectors $\boldsymbol{\lambda}$, $\boldsymbol{w}$ and $\boldsymbol{s}$, $\boldsymbol{w}$ is confined into the compact polyhedral set G_0 and $\boldsymbol{s}$ into the compact polyhedral set $\Pi_{i=1}^{m_1}[0, \bar{s}_i]$. Here $\bar{s}_i$ denotes a finite positive number given in (5.4). We have also assumed to know a positive number M which bounds the Lagrange multipliers λ_i $(i = 1, 2, \ldots, m_1)$ in Section 5.1. Such a positive number certainly exists in view of (ii) of Condition 5.1.1. It is usually difficult, however, to estimate (tight) upper bounds for the Lagrange multipliers from the QOP formulation (5.3) of the BQOP (5.2).

To avoid such a difficulty, we artificially restrict the range of values for Lagrange multipliers by introducing a scaling variable α into the QOP (5.3):

$$\left.\begin{array}{ll}
\displaystyle\max_{\boldsymbol{\mu},\boldsymbol{s},\boldsymbol{w},\alpha} & \boldsymbol{c}_1^T\boldsymbol{v} + \boldsymbol{c}_2^T\boldsymbol{u} \\
\text{subject to} & \displaystyle\alpha\nabla_v g_0(\boldsymbol{v},\boldsymbol{u}) + \sum_{i=1}^{m_1}\mu_i\nabla_v g_i(\boldsymbol{v},\boldsymbol{u}) = \boldsymbol{0}, \\
 & g_i(\boldsymbol{v},\boldsymbol{u}) + s_i = 0,\ 0 \le s_i \le \bar{s}_i\ (i = 1,\ldots,m_1), \\
 & \mu_i s_i = 0,\ 0 \le \mu_i \le 1\ (i = 1,\ldots,m_1), \\
 & \displaystyle\alpha + \sum_{i=1}^{m_1}\mu_i = 1,\ 0 \le \alpha \le 1, \\
 & f_i(\boldsymbol{v},\boldsymbol{u}) \le 0\ (i = 1,\ldots,m_2),\ \boldsymbol{w} = \begin{pmatrix}\boldsymbol{v}\\ \boldsymbol{u}\end{pmatrix} \in G_0.
\end{array}\right\} \tag{5.6}$$

Lemma 5.2.1. *The QOP (5.3) is equivalent to the scaled QOP (5.6) in the sense that* $(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{w}) \in R^{2m_1+n}$ *is a feasible solution of the QOP (5.3) if and only if* $(\boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{w}, \alpha) \in R^{2m_1+n+1}$ *is a feasible solution of the scaled QOP (5.6) with* $\boldsymbol{\mu} = \alpha\boldsymbol{\lambda}$ *and some* $\alpha > 0$.

Proof: Suppose that $(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{w}) \in R^{2m_1+n}$ is a feasible solution of the QOP (5.3). Let

$$\alpha = \frac{1}{1 + \displaystyle\sum_{i=1}^{m_1}\lambda_i} > 0 \ \text{ and } \boldsymbol{\mu} \ = \ \alpha\boldsymbol{\lambda}.$$

The above equalities derive $\alpha\,(1 + \sum_{i=1}^{m_1}\lambda_i) = \alpha + \sum_{i=1}^{m_1}\mu_i = 1$, which corresponds to one of the constraints in (5.6). Clearly, $\boldsymbol{\mu}$ and α defined above satisfy the other constraints of (5.6), and hence, $(\boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{w}, \alpha) \in R^{2m_1+n+1}$ is a feasible solution of the scaled BQOP (5.6). Now suppose that $(\boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{w}, \alpha)$ is a feasible solution of the scaled QOP (5.6). If $\alpha = 0$, then the constraints $\alpha + \sum_{i=1}^{m_1}\mu_i = 1$ and $\alpha\nabla_v g_0(\boldsymbol{v},\boldsymbol{u}) + \sum_{i=1}^{m_1}\mu_i\nabla_v g_i(\boldsymbol{v},\boldsymbol{u}) = \boldsymbol{0}$ of (5.6) contradict to (ii) of Condition 5.1.1. Hence $\alpha > 0$. Let $\boldsymbol{\lambda} = \boldsymbol{\mu}/\alpha$. Then $(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{w})$ turns out to be a feasible solution of the QOP (5.3). ∎

We rewrite the scaled QOP (5.6) as

$$\max\ \boldsymbol{c}^T\boldsymbol{x} \text{ subject to } \boldsymbol{x} \in F, \tag{5.7}$$

where

$$
\begin{aligned}
\boldsymbol{x} &= \begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{s} \\ \boldsymbol{w} \\ \alpha \end{pmatrix} \in R^{2m_1+n+1},\ \boldsymbol{c} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \bar{\boldsymbol{c}} \\ 0 \end{pmatrix} \in R^{2m_1+n+1},\ \bar{\boldsymbol{c}} = \begin{pmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \end{pmatrix} \in R^n, \\
F &= \{\boldsymbol{x} \in C_0 : p(\boldsymbol{x}) \leq 0 \ \ (\forall p(\cdot) \in \mathcal{P}_F)\}, \\
\mathcal{P}_F &= \left\{ \begin{array}{l} q_i(\cdot) \ (i = 1, \ldots, m_1 + n_1 + m_1 + m_2), \\ -q_j(\cdot) \ (j = 1, \ldots, m_1 + n_1 + m_1) \end{array} \right\}, \\
\boldsymbol{q}(\boldsymbol{x}) &= \begin{pmatrix} q_1(\boldsymbol{x}) \\ \vdots \\ q_{m_1}(\boldsymbol{x}) \\ \vdots \\ q_{m_1+n_1}(\boldsymbol{x}) \\ q_{m_1+n_1+1}(\boldsymbol{x}) \\ \vdots \\ q_{m_1+n_1+m_1}(\boldsymbol{x}) \\ q_{m_1+n_1+m_1+1}(\boldsymbol{x}) \\ \vdots \\ q_{m_1+n_1+m_1+m_2}(\boldsymbol{x}) \end{pmatrix} = \begin{pmatrix} \mu_1 s_1 \\ \vdots \\ \mu_{m_1} s_{m_1} \\ \alpha \nabla_v g_0(\boldsymbol{v}, \boldsymbol{u}) + \sum_{i=1}^{m_1} \mu_i \nabla_v g_i(\boldsymbol{v}, \boldsymbol{u}) \\ g_1(\boldsymbol{v}, \boldsymbol{u}) + s_1 \\ \vdots \\ g_{m_1}(\boldsymbol{v}, \boldsymbol{u}) + s_{m_1} \\ f_1(\boldsymbol{v}, \boldsymbol{u}) \\ \vdots \\ f_{m_2}(\boldsymbol{v}, \boldsymbol{u}) \end{pmatrix} \quad \text{for } \forall \boldsymbol{x} = \begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{s} \\ \boldsymbol{w} \\ \alpha \end{pmatrix}, \\
C_0 &= \left\{ \boldsymbol{x} = \begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{s} \\ \boldsymbol{w} \\ \alpha \end{pmatrix} \in R^{2m_1+n+1} : \begin{array}{l} 0 \leq \mu_i \leq 1 \ (i = 1, \ldots, m_1), \\ \alpha + \sum_{i=1}^{m_1} \mu_i = 1, \ 0 \leq \alpha \leq 1, \\ 0 \leq s_i \leq \bar{s}_i \ (i = 1, \ldots, m_1), \\ \boldsymbol{w} \in G_0 \end{array} \right\}.
\end{aligned}
$$

Now C_0 is a compact polyhedral set, so that we can apply Algorithms 4.1.1 and 4.1.2 to the problem (5.7).

5.2.2 Tightening Upper Bounds for Each Complementary Pair of Variables

If $\boldsymbol{x}$ is a feasible solution of the QOPs (5.5) or (5.7), then it must satisfy the complementarity condition:

$$x_i \geq 0, \ x_{m_1+i} \geq 0 \ \text{ and } x_i x_{m_1+i} = 0, \quad i = 1, 2 \ldots m_1. \tag{5.8}$$

Exploiting the complementarity condition, we can tighten upper bounds on some of the nonnegative variables x_i $(i = 1, 2, \ldots, 2m_1)$. In fact, if the set $\{\boldsymbol{x} \in C_k : x_{m_1+i} = 0\}$ is empty, we can conclude that $x_{m_1+i} > 0$ and $x_i = 0$ for every $\boldsymbol{x} \in F$. Otherwise

$$x_i \leq \max\{\boldsymbol{e}_i^T \boldsymbol{x} : \boldsymbol{x} \in C_k, \ x_{m_1+i} = 0\} \ \text{ for every } \boldsymbol{x} \in F.$$

Therefore, in Step 1 and Step 4 with $D_2 = D(\pi/2)$ of Algorithms 4.1.1 and 4.1.2, we can replace $\alpha(C_k, \boldsymbol{e}_i)$ by

$$\alpha'(C_k, \boldsymbol{e}_i) = \begin{cases} 0 & \text{if } \{\boldsymbol{x} \in C_k : x_{m_1+i} = 0\} = \emptyset, \\ \max\{\boldsymbol{e}_i^T \boldsymbol{x} : \boldsymbol{x} \in C_k, \ x_{m_1+i} = 0\} & \text{otherwise,} \end{cases}$$

and similarly $\alpha(C_k, \boldsymbol{e}_{m_1+i})$ by

$$\alpha'(C_k, \boldsymbol{e}_{m_1+i}) = \begin{cases} 0 & \text{if } \{\boldsymbol{x} \in C_k : x_i = 0\} = \emptyset, \\ \max\{\boldsymbol{e}_{m_1+i}^T \boldsymbol{x} : \boldsymbol{x} \in C_k,\ x_i = 0\} & \text{otherwise.} \end{cases}$$

It should be noted that $\alpha'(C_k, \boldsymbol{e}_i) \leq \alpha(C_k, \boldsymbol{e}_i)$ and $\alpha'(C_k, \boldsymbol{e}_{m_1+i}) \leq \alpha(C_k, \boldsymbol{e}_{m_1+i})$ in general. If the strict inequality holds above, then we can strengthen the SDP (or SILP) relaxation in Step 6 of Algorithm 4.1.1 (or Algorithm 4.1.2). We call Algorithm 4.1.1 (or Algorithm 4.1.2) combined with this technique as the modified Algorithm 4.1.1 (or modified Algorithm 4.1.2).

5.2.3 An Illustrating Example

In this section, we present an example to highlight the main idea of this paper. This example of bilevel quadratic program is taken from Shimizu and Aiyoshi [58].

$$\left.\begin{array}{ll} \max\limits_{x} & F(x,y) = -x^2 - (y-10)^2 \\ \text{subject to} & y \in \operatorname*{argmin}\limits_{y} \left\{ (x+2y-30)^2 : \begin{array}{l} x+y \leq 20, \\ 0 \leq y \leq 20 \end{array} \right\}, \\ & 0 \leq x \leq 15,\ -x+y \leq 0. \end{array}\right\} \tag{5.9}$$

Applying the KKT optimality condition to the lower level problem and introducing a scaling variable α, we transform the problem (5.9) into

$$\left.\begin{array}{ll} \max & t \\ \text{subject to} & \\ & \left.\begin{array}{l} x^2 + (y-10)^2 + t = 0, \\ x+y \leq 20,\ -x+y \leq 0, \end{array}\right\} \text{(feasibility)} \\ & \alpha(4x+8y-120) + \mu_1 + \mu_2 - \mu_3 = 0,\ \text{(stationarity)}, \\ & \left.\begin{array}{l} \mu_1(20-x-y) = 0, \\ \mu_2(20-y) = 0,\ \ \mu_3 y = 0, \end{array}\right\} \text{(complementarity)} \\ & \left.\begin{array}{l} \alpha + \sum_{i=1}^{3} \mu_i = 1,\ \ 0 \leq \alpha \leq 1, \\ 0 \leq \mu_i \leq 1\ (i = 1, \cdots, 3), \\ 0 \leq x \leq 15,\ 0 \leq y \leq 20. \end{array}\right\} \text{(bounds)} \end{array}\right\} \tag{5.10}$$

In this example (5.9), the upper level problem has a quadratic objective function, so that we have replaced it by a new single variable t to make a linear objective function.

In the scaled QOP formulation (5.10), the scaling technique presented in Section 5.2.1 has derived the quadratic equality constraint

$$\alpha(4x+8y-120) + \mu_1 + \mu_2 - \mu_3 = 0,$$

while if we assume a sufficiently large number M as upper bounds for the Lagrange multipliers λ_i $(i = 1, \ldots, 3)$ of the lower level problem of (5.9), we have the linear equality constraint

$$4x + 8y - 120 + \lambda_1 + \lambda_2 - \lambda_3 = 0.$$

Therefore, the scaling technique has created an additional quadratic equality constraint which may worsen the quality of approximation of the maximum objective function value, although the technique is important because M chosen may not be guaranteed to bound the Lagrange multipliers λ_i $(i = 1, \dots, 3)$.

Now we apply the technique proposed in Section 5.2.2 to the complementarity constraints $\mu_1(20 - x - y) = 0, \mu_2(20 - y) = 0$ and $\mu_3 y = 0$. For simplicity of notation, we use a variable itself instead of the corresponding unit coordinate vector $\boldsymbol{e}_i$ below; for example, $\alpha(C_k, y)$ stands for $\alpha(C_k, \boldsymbol{e}_i)$ where $\boldsymbol{e}_i$ denotes the unit coordinate vector corresponding to the y axis. When a set $D_k \subset \overline{D}$ of direction vectors at the kth iterate is equal to $D(\pi/2) = \pm I$, we compute

$$\begin{aligned} \alpha'(C_k, y) &= \max\{y : (x, y, \boldsymbol{\mu}, \alpha, t) \in C_k,\ \mu_3 = 0\}, \\ \alpha'(C_k, -y) &= \max\{-y : (x, y, \boldsymbol{\mu}, \alpha, t) \in C_k,\ \mu_2 = 0\}, \\ \alpha'(C_k, \mu_1) &= \max\{\mu_1 : (x, y, \boldsymbol{\mu}, \alpha, t) \in C_k,\ x + y = 20\}, \end{aligned}$$

instead of $\alpha(C_k, y), \alpha(C_k, -y)$ and $\alpha(C_k, \mu_1)$. Also, if $\alpha'(C_k, y) < 20$ or $\alpha'(C_k, -y) < 0$ holds in this example, we can set $\mu_2 = 0$ or $\mu_3 = 0$ in the succeeding iterations, respectively. Regarding to upper bounds for μ_2 and μ_3, we also obtain tighter values by

$$\begin{aligned} \alpha'(C_k, \mu_2) &= \max\{\mu_2 : (x, y, \boldsymbol{\mu}, \alpha, t) \in C_k,\ y = 20,\ \mu_3 = 0\}, \\ \alpha'(C_k, \mu_3) &= \max\{\mu_3 : (x, y, \boldsymbol{\mu}, \alpha, t) \in C_k,\ y = 0,\ \mu_2 = 0\}. \end{aligned}$$

5.3 Computational Experiments on BQOPs

For some quadratic bilevel programming problems, we now present numerical results that illustrate the behavior of the specialized Algorithm 4.1.2 with techniques proposed in Section 5.2. The program was coded in C++ language and run on a DEC Alpha Workstation (600 MHz with 1GB of memory). We used CPLEX Version 6.0 as LP solver to compute $\alpha(C_k, \boldsymbol{d})$ in Steps 1, 2 and 4 of Algorithm 4.1.2.

5.3.1 Some Implementation Details

We start Algorithm 4.1.2 by constructing a set $D_2 = D(\theta)$ with $\theta = \pi/2$ of direction vectors according to the definition (4.2). If the decrease in the upper bound ζ_k for the maximum objective function value becomes little in some iteration, we reduce the value θ to replace $D(\theta)$. Otherwise, we use the same set of direction vectors as that of the previous iteration. Throughout the computational experiments, we use the following replacement rule:

Let $\ell = 0$, $\theta_0 = \pi/2$, $K = 3$ and $\{\sigma_j\}_{j=0}^{K}$ be a decreasing sequence such that $\{1, \frac{8}{9}, \frac{4}{9}, \frac{2}{9}\}$. If the upper bound ζ_k generated at the kth iteration remains to satisfy

$$\frac{\zeta_{k-1} - \zeta_k}{\max\{|\zeta_k|,\ 1.0\}} \geq 0.001 \times \sigma_\ell,$$

then set $\theta_k = \theta_{k-1}$. Else if $\ell < K$, then set $\ell = \ell + 1$ and $\theta_k = \sigma_\ell \theta_0$, which implies the replacement of $D(\theta_k)$. Otherwise stop the procedure.

For the comparison, we implemented Algorithm 4.1.2 with another typical algorithm related with the lift-and-project procedure for quadratic programs; the RLT (Reformulation-Linearization Technique) proposed by [55, 57]. The QOP formulations (5.3) and (5.6) of BQOPs usually have some linear constraints such as bound constraints in addition to quadratic ones. Following the idea of the RLT, we generate quadratic constraints through the the products of pairwise linear constraints. Together with the original quadratic constraints, those new ones are added as input data for the algorithm. We call the expanded input data as "DataRLT" while the original data as "DataDLSSILP". We compare the following five cases:

DLSSILP	:	Algorithm 4.1.2 with input DataDLSSILP;
DLSSILP+RLT	:	Algorithm 4.1.2 with input DataRLT;
mDLSSILP	:	the modified Algorithm 4.1.2 (Section 5.2.2) with input DataDLSSILP;
mDLSSILP+RLT	:	the modified Algorithm 4.1.2 (Section 5.2.2) with input DataRLT;
RLT	:	the LP relaxation for input DataRLT;

with respect to the following items:

$f$up	:	the solution value found by each algorithm,
R.error	:	the relative error of a solution, *i.e.*, $\dfrac{\lvert f\text{up} - f_{\text{opt}}\rvert}{\max\{\lvert f_{\text{opt}}\rvert, 1.0\}}$, where f_{opt} is the global optimum value;
cpu	:	the cpu time in second;
iter.	:	the number of iterations the algorithm repeated.

5.3.2 Numerical Results

We evaluate the five methods described in the previous subsection (Section 5.3.1) using a set of test problems. Table 5.1 shows some characteristics of the test problems. n_1, n_2, m_1 and m_2 are introduced in the formulation (5.2), and #conv (or #non-conv) indicates the number of convex (or nonconvex, respectively) quadratic constraints in the lower or upper level problem. The transformed one-level mathematical program (5.3) via the KKT optimality condition, includes additional m_1 nonconvex complementarity constraints. Moreover, the technique proposed in Section 5.2.1 increases the number of nonconvex quadratic constraints by n_1. Therefore, the QOP (5.3) (or the scaled QOP (5.6)) has #non-conv+m_1 (or #non-conv+m_1+n_1, respectively) nonconvex quadratic constraints. The test problems of Table 5.1 are basically chosen from the literature. A nonconvex BQOP, which we call "bard2", is constructed from Problem "bard1" by multiplying the upper level convex objective function by -1.

Table 5.1: The test problems

Problem	Source	n_1	n_2	m_1	m_2	f_{opt}
bard1	[10]	1	1	5	2	-17.00
bard2	–	1	1	5	2	68.78
bard3	[10]	2	2	5	4	14.36
shimizu1	[58]	1	1	4	4	-100.00
shimizu2	[58]	2	2	5	4	-225.00
aiyoshi1	[2]	2	2	7	5	-60.00
aiyoshi2	[1]	4	4	14	10	6600.00

Tables 5.2 and 5.3 present numerical results on the test problems when they are reformulated into the QOP (5.3) and the scaled QOP (5.6), respectively. Figures 5.1 and 5.2 show how the upper bound ζ_k for the maximum objective function value of Problem "shimizu1" decreases as the iteration proceeds. Problem "shimizu1" of Figure 5.1 takes the QOP formulation (5.3) and that of Figure 5.2 the scaled QOP formulation (5.6). The lines "DLSSILP+RLT" and "mDLSSILP+RLT" in Figure 5.2 designate the similar performance of achieving the global optimum value for Problem "shimizu1", though the initial upper bounds for Lagrange multipliers are different.

Our experiments were conducted in order to see how the following three factors affect the behavior of the algorithms: (i) the scaling technique for Lagrange multipliers (Section 5.2.1); (ii) the technique for tightening the bound constraints (Section 5.2.2); (iii) the effect of the RLT.

(i) In the scaled QOP formulation (5.6), we need no deductive upper bounds for Lagrange multipliers, while in the QOP formulation (5.3) we assumed that $\lambda_i \leq 1000$, $i = 1, \ldots, m_2$ by taking $M = 1000$ for all the test problems. Comparing the results in Table 5.2 with those in Table 5.3, we see that the scaling technique works effectively in several instances such as Problems "bard1", "bard2" and "shimizu1" in which better upper bounds for the maximum objective function values were attained. As we have discussed in Section 5.2.3, however, the scaling technique generates new quadratic equality constraints, which influence the performance of Algorithm 4.1.2 applied to the scaled QOP formulation (5.6). In Problems "shimizu2" and "aiyoshi1" of Tables 5.2 and 5.3, we observe that the scaled QOP formulation (5.6) makes the quality of the upper bound worse.

(ii) Comparing the DLSSILP and mDLSSILP (or, DLSSILP+RLT and mDLSSILP+RLT) cases in Tables 5.2 and 5.3, we see the effect of tightening the bounds for some variables. Especially, Problem "aiyoshi1" in Table 5.2 shows the fast convergence to a better upper bound for the maximum objective function value due to the tight bound

Table 5.2: DataDLSSILP of the QOP (5.3)

Problem	DLSSILP				DLSSILP+RLT				RLT		
	$f_{\rm up}$	R.error	cpu	iter.	$f_{\rm up}$	R.error	cpu	iter.	$f_{\rm up}$	R.error	cpu
bard1	-3.07	8.2e-1	4.6	46	-14.92	1.2e-1	4.4	30	-5.00	7.1e-1	0.0
bard2	100.96	4.7e-1	0.9	16	68.86	1.2e-3	0.8	7	75.78	1.0e-1	0.0
bard3	19.99	3.9e-1	2.8	11	14.54	1.2e-2	3.5	14	33.64	1.3e+0	0.0
shimizu1	-95.37	4.6e-2	0.3	14	-96.45	3.6e-2	0.3	11	0.00	1.0e+0	0.0
shimizu2	-224.13	3.9e-3	0.8	11	-225.00	6.1e-15	0.8	6	-125.00	4.4e-1	0.0
aiyoshi1	-55.41	7.7e-2	21.6	53	-59.69	5.2e-3	6.9	11	-41.70	3.1e-1	0.0
aiyoshi2	6786.19	2.8e-2	38.1	8	6625.64	3.9e-3	140.0	7	7200.00	9.1e-2	0.3

Problem	mDLSSILP				mDLSSILP+RLT			
	$f_{\rm up}$	R.error	cpu	iter.	$f_{\rm up}$	R.error	cpu	iter.
bard1	-3.07	8.2e-1	4.2	46	-14.92	1.2e-1	4.2	30
bard2	100.91	4.7e-1	0.8	15	68.86	1.2e-3	0.8	7
bard3	18.90	3.2e-1	5.5	21	14.57	1.5e-2	2.9	12
shimizu1	-95.75	4.3e-2	0.2	11	-96.74	3.3e-2	0.2	10
shimizu2	-225.00	5.3e-15	0.4	6	-225.00	3.8e-16	0.7	6
aiyoshi1	-59.82	3.0e-3	2.3	9	-60.00	1.2e-15	1.4	6
aiyoshi2	6647.18	7.2e-3	20.3	6	6614.68	2.2e-3	100.2	6

Figure 5.1: Problem “shimizu1 (QOP)”

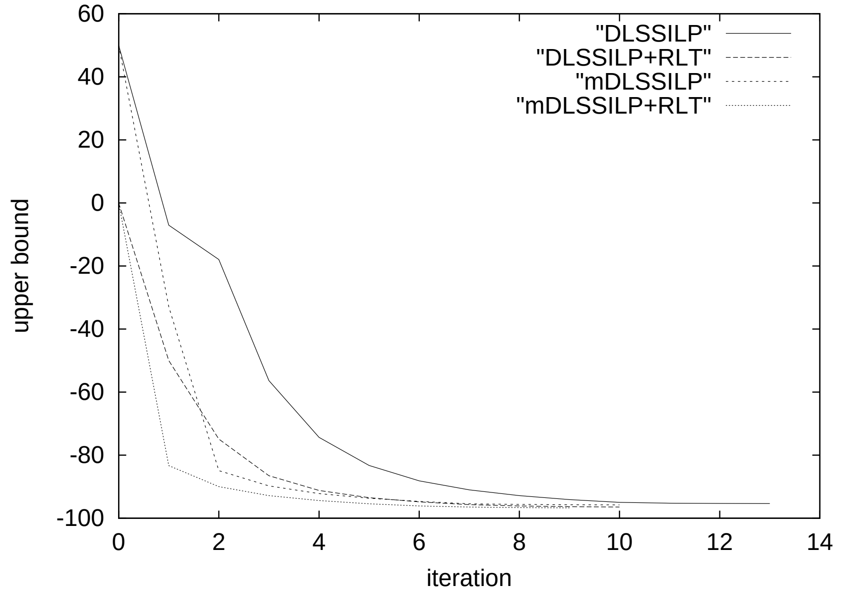
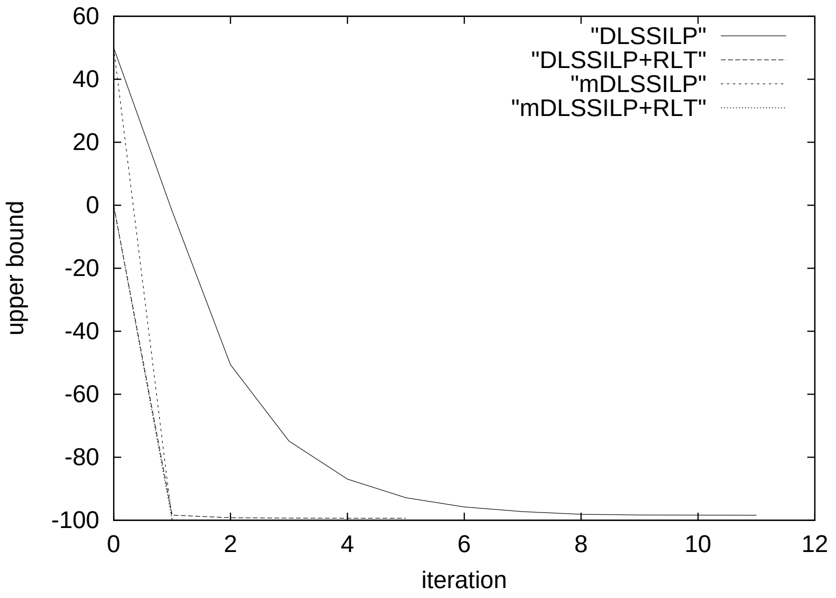


Table 5.3: DataDLSSILP of the scaled QOP (5.6)

Problem	DLSSILP				DLSSILP+RLT				RLT		
	$f_{\rm up}$	R.error	cpu	iter.	$f_{\rm up}$	R.error	cpu	iter.	$f_{\rm up}$	R.error	cpu
bard1	-14.75	1.3e-1	6.3	40	-16.40	3.6e-2	4.6	26	-3.00	8.2e-1	0.0
bard2	72.01	4.7e-2	1.8	24	68.78	3.0e-9	0.8	8	94.67	3.8e-1	0.0
bard3	14.80	3.1e-2	4.2	16	14.57	1.4e-2	3.9	13	33.64	1.3e+0	0.0
shimizu1	-98.39	1.6e-2	0.4	12	-99.36	6.4e-3	0.2	6	0.00	1.0e+0	0.0
shimizu2	-99.54	5.6e-1	1.9	11	-122.71	4.6e-1	6.6	24	-25.00	8.9e-1	0.0
aiyoshi1	-59.43	9.5e-3	29.6	26	-55.90	6.8e-2	29.8	31	-31.00	4.8e-1	0.0
aiyoshi2	6841.73	3.7e-2	19.5	8	6769.46	2.6e-2	48.1	6	7200.00	9.1e-2	0.1

Problem	mDLSSILP				mDLSSILP+RLT			
	$f_{\rm up}$	R.error	cpu	iter.	$f_{\rm up}$	R.error	cpu	iter.
bard1	-14.79	1.3e-1	6.1	40	-16.42	3.4e-2	4.2	23
bard2	72.35	5.2e-2	1.7	23	68.78	3.0e-9	0.8	8
bard3	14.80	3.0e-2	4.0	15	14.56	1.4e-2	3.5	13
shimizu1	-100.00	1.4e-15	0.2	6	-100.00	2.7e-15	0.3	6
shimizu2	-168.58	2.5e-1	3.3	30	-210.25	6.6e-2	1.5	10
aiyoshi1	-59.34	1.1e-2	25.3	26	-56.07	6.6e-2	22.0	32
aiyoshi2	6841.72	3.7e-2	18.5	8	6768.79	2.6e-2	46.4	6

Figure 5.2: Problem “shimizu1 (Scaled QOP)”



constraints, and also, Problem “shimizu1” in Table 5.3 shows a significant improvement. There are, however, some exceptions, e.g. in Problem “aiyoshi1” of Table 5.3, the DLSSILP case shows a little better performance than the mDLSSILP case. This is due to the difference in timing when the parameter θ of Algorithm 4.1.2 changes in the cases DLSSILP and mDLSSILP.

(iii) While the sole use of the RLT generates rough upper bounds, the RLT enhances the efficiency of Algorithm 4.1.2 as Tables 5.2, 5.3 and Figures 5.1, 5.2 show. Although the combined method (mDLSSILP+RLT) required the greatest computational effort, it achieved the tightest upper bound with less computing time in several instances, due to its fast convergence.

Our method consumes relatively much computing time in order to achieve tighter upper bounds for maximum objective function values. However, note that Figures 5.1 and 5.2 show great improvements in upper bounds for the maximum objective function values at the first several iterations. Further extension of the research could be to incorporate the first several iterations of the SCRM into the branch-and-bound method for solving difficult BQOPs including several nonconvex quadratic constraints.

Chapter 6

Parallel Implementations of Successive Convex Relaxation Methods

As observed in previous chapters, SCRMs (successive convex relaxation methods) with the use of SDP (semidefinite programming) or SILP (semi-infinite LP) relaxation solve a large number of SDPs or LPs at every iteration in order to constitute relaxed convex regions C_k ($k = 1, 2, \ldots$) of the feasible region F of a QOP (quadratic optimization problem). On that occasion, SCRMs generate a large number of problems simultaneously and solve them sequentially on a single processor. In this chapter, we pay attention to the structure of SCRMs suitable to parallel computing, and propose parallel SCRMs which process some problems at the same time using multiple processors on a client-server parallel computing system.

To enhance the effect of parallel computing, we reduce the work of a client machine and also, decrease transmitting time between computers as much as possible. The result is a highly parallel algorithm, which we implement on the Ninf (network based information library for high performance computing) system [50, 52]. Moreover, the parallel SCRMs adopt new construction for C_k ($k = 1, 2, \ldots$) so that the number of constraints of each SDP or LP is considerably decreased. As a result, we can deal with some larger-sized QOPs, which not only previous SCRMs but also a popular bounding technique [57] called RLT (Reformulation-Linearization Technique) cannot process.

This chapter consists of the following four sections. In Section 6.1, we introduce basic discretized-localized SCRMs with successive SDP or SILP relaxations and presents their serial implementation. In Section 6.2 we provide new construction for convex relaxations C_k ($k = 1, 2, \ldots$) of the feasible region of a QOP. The construction increases the number of SDPs or LPs to be solved sometimes, but decreases the number of constraints included in each SDP or LP considerably. Then, we show a parallel algorithm of the discretized-localized SCRMs. In Section 6.3, we report numerical results of a parallel SCRM on a client-server parallel computing system.

6.1 Properties of Previous SCRMs

As practical discretized-localized SCRMs, we proposed Algorithms 4.1.1 for the SSDP (successive semidefinite programming) relaxation method, and Algorithms 4.1.2 for the SSILP (successive semi-infinite LP) relaxation method in Chapter 4. Concerning the SSILP relaxation method, its numerical results were already reported there.

An aim of this chapter is to provide new SCRMs suitable for parallel computing, whose function-sets P_k and relaxations C_k are different from the ones described in Chapter 4. To highlight the difference between the previous SCRMs and our new SCRMs presented here, we will present how to construct P_k from the direction-sets D_1 and D_2 for each SCRM. Before taking up the main subject, we describe our problem dealt with in this chapter, and depict a basic algorithm of discretized-localized SCRMs. We assume that the feasible region of a QOP consists of a finite number of quadratic constraints. Then, the QOP can be written as follows;

$$\text{(QOP)} \quad \left| \begin{array}{ll} \max & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} & \gamma_\ell + 2\boldsymbol{q}_\ell^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_\ell\boldsymbol{x} \leq 0, \ \ell = 1,\ldots,m, \end{array} \right. \tag{6.1}$$

where m is some positive number, $\boldsymbol{c} \in R^n$, $\gamma_\ell \in R$, $\boldsymbol{q}_\ell \in R^n$ and $\boldsymbol{Q}_\ell \in R^{n\times n}$ $(\ell = 1,\ldots,m)$. Using a compact convex set C_0 such that $F \subseteq C_0$, we write a feasible region of (6.1) as

$$\begin{array}{rcl} F & \equiv & \{\boldsymbol{x} \in C_0 : qf(\boldsymbol{x};\gamma,\boldsymbol{q},\boldsymbol{Q}) \leq 0 \ \ (\forall qf(\cdot;\gamma,\boldsymbol{q},\boldsymbol{Q}) \in \mathcal{P}_F)\}, \\ qf(\boldsymbol{x};\gamma,\boldsymbol{q},\boldsymbol{Q}) & \equiv & \gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}, \ \forall \boldsymbol{x} \in R^n, \\ \mathcal{P}_F & \equiv & \{\gamma_\ell + 2\boldsymbol{q}_\ell^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_\ell\boldsymbol{x} : \ \ell = 1,\ldots,m\}. \end{array}$$

Algorithm 6.1.1. (serial implementation of discretized-localized SCRMs)

Step 0: Let $D_1 \subseteq \overline{D}$. Construct a function-set $\mathcal{P}^L(C_0, D_1)$ by computing

$$\alpha(C_0,\boldsymbol{d}) = \max\{\boldsymbol{d}^T\boldsymbol{x} : \boldsymbol{x} \in C_0\} \ \ (\forall \boldsymbol{d} \in D_1).$$

Let $k = 0$.

Step 1: If $C_k = \emptyset$, then stop. Compute an upper bound ζ_k for the maximum objective function value of QOP (6.1) by $\zeta_k = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in C_k\}$. If ζ_k satisfies the termination criteria, then stop.

Step 2: Let $D_2 = D(\theta)$ with some value θ. Compute

$$\alpha(C_k,\boldsymbol{d}) = \max\{\boldsymbol{d}^T\boldsymbol{x} : \boldsymbol{x} \in C_k\} \ \ (\forall \boldsymbol{d} \in D_2).$$

Step 3: Construct a set $\mathcal{P}_k = \mathcal{P}^L(C_0, D_1) \cup \mathcal{P}^2(C_k, D_1, D_2)$. The set $\mathcal{P}_k$ induces valid inequalities for C_k.

Step 4: Define

$$C_{k+1} = \begin{cases} \widehat{F}(C_0, \mathcal{P}_F \cup \mathcal{P}_k) & \text{for the SSDP relaxation method,} \\ \widehat{F}^L(C_0, \mathcal{P}_F \cup \mathcal{P}_k) & \text{for the SSILP relaxation method.} \end{cases}$$

Step 5: Let $k = k + 1$, and go to Step 1.

The termination criteria of Step 1 and the choice of θ for the set $D(\theta)$ follow from the rule of Algorithm 4.1.1. Algorithm 6.1.1 above lacks the definitions for the sets of vectors D_1 and D_2. In previous works [32, 33, 62, 64], SCRMs commonly utilize D_1 such as

$$D_1 \quad = \quad \pm I \equiv \{\boldsymbol{e}_1, \dots, \boldsymbol{e}_n, -\boldsymbol{e}_1, \dots, -\boldsymbol{e}_n\}. \tag{6.2}$$

Practical SCRMs, proposed in Chapter 4, construct $D_2 = D(\theta)$ as

$$D(\theta) \equiv \{\ \boldsymbol{b}_{i+}(\theta),\ \boldsymbol{b}_{i-}(\theta),\quad i = 1, 2, \dots, n\} \tag{6.3}$$

with a parameter $\theta \in (0, \pi/2]$. Here,

$$\nu_{i+}(\theta) = \|\boldsymbol{c}\cos\theta + \boldsymbol{e}_i \sin\theta\|, \quad \nu_{i-}(\theta) = \|\boldsymbol{c}\cos\theta - \boldsymbol{e}_i \sin\theta\|,$$

$$\boldsymbol{b}_{i+}(\theta) = \frac{\boldsymbol{c}\cos\theta + \boldsymbol{e}_i \sin\theta}{\nu_{i+}(\theta)}, \qquad \boldsymbol{b}_{i-}(\theta) = \frac{\boldsymbol{c}\cos\theta - \boldsymbol{e}_i \sin\theta}{\nu_{i-}(\theta)}.$$

In (6.3), the objective direction $\boldsymbol{c}$ is excluded from $D(\theta)$ defined in Chapter 4, for simplicity of following discussion. Therefore, $\mathcal{P}^2(C_k, D_1, D_2)$ with $D_1 = \pm I$ and $D_2 = D(\theta)$ is constructed as

$$\mathcal{P}^2(C_k, D_1, D_2) = \left\{ \begin{array}{ll} r2sf(\boldsymbol{x}; C_k, -\boldsymbol{e}_j, \boldsymbol{b}_{i+}(\theta)), & r2sf(\boldsymbol{x}; C_k, \boldsymbol{e}_j, \boldsymbol{b}_{i-}(\theta)) \\ r2sf(\boldsymbol{x}; C_k, \boldsymbol{e}_j, \boldsymbol{b}_{i+}(\theta)), & r2sf(\boldsymbol{x}; C_k, -\boldsymbol{e}_j, \boldsymbol{b}_{i-}(\theta)) \\ i = 1, \dots, n,\ j = 1, \dots, n & \end{array} \right\}. \tag{6.4}$$

In practical SCRMs, $\mathcal{P}^2(C_k, D_1, D_2)$ is constructed according to (6.4) in Step 3 of Algorithm 6.1.1. If the algorithm takes the SDP relaxation $\widehat{F}(C_0, \mathcal{P}_F \cup \mathcal{P}_k)$ as C_{k+1}, we call the algorithm *DLSSDP*, and if the algorithm takes the SILP relaxation $\widehat{F}^L(C_0, \mathcal{P}_F \cup \mathcal{P}_k)$ instead of $\widehat{F}(C_0, \mathcal{P}_F \cup \mathcal{P}_k)$, we call it *DLSSILP*.

When we take $\theta = \pi/2$ at the first iteration of Algorithm 6.1.1, the vectors $\boldsymbol{b}_{i+}(\theta)$ and $\boldsymbol{b}_{i-}(\theta)$ $(i = 1, \dots, n)$ of D_2 turn out to be the unit vectors $\boldsymbol{e}_i$ and $-\boldsymbol{e}_i$, respectively. Then, the values $\alpha(C_0, \boldsymbol{e}_i)$ and $-\alpha(C_0, -\boldsymbol{e}_i)$ correspond upper and lower bounds for the variable x_i, respectively. In this case, the set $\mathcal{P}^2(C_0, D_1, D_2)$ consists of the pairwise products of lower and upper bounding constraints for variables x_i $(i = 1, 2, \dots, n)$. These constraints correspond to underestimators and overestimators of quadratic terms $x_i x_j$ $(i, j = 1, 2, \dots, n)$, which were introduced in [37] and have been used in some lower (or upper) bounding procedure of branch-and-bound methods (for instance, see [47, 71]). We also see that both $\boldsymbol{b}_{i+}(\theta)$

and $\boldsymbol{b}_{i-}(\theta)$ $(i = 1, \ldots, n)$ approach to the objective direction $\boldsymbol{c}$ as $\theta \to 0$. See Chapter 4 for more details.

Here we will show that each quadratic function $\gamma_\ell + 2\boldsymbol{q}_\ell^T \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{Q}_\ell \boldsymbol{x}$ $(\ell = 1, \ldots, m)$ of $\mathcal{P}_F$ can be convexified (or linearized) in the relaxation $\widehat{F}(C_0, \mathcal{P}_F \cup \mathcal{P}_k)$ (or $\widehat{F}^L(C_0, \mathcal{P}_F \cup \mathcal{P}_k)$) through the use of quadratic functions of $\mathcal{P}^2(C_k, D_1, D_2) \subset \mathcal{P}_k$ defined above. To show that, we use Lemma 2.2.3 of Chapter 2:

(i) $\widehat{F}(C_0, \mathcal{P}_F \cup \mathcal{P}_k) = \{\boldsymbol{x} \in C_0 : p(\boldsymbol{x}) \leq 0 \ \ (\forall p(\cdot) \in \text{c.cone}(\mathcal{P}_F \cup \mathcal{P}_k) \cap \mathcal{Q}_+)\}$,

(ii) $\widehat{F}^L(C_0, \mathcal{P}_F \cup \mathcal{P}_k) = \{\boldsymbol{x} \in C_0 : p(\boldsymbol{x}) \leq 0 \ \ (\forall p(\cdot) \in \text{c.cone}(\mathcal{P}_F \cup \mathcal{P}_k) \cap \mathcal{L})\}$.

Adding some two functions of $\mathcal{P}^2(C_k, D_1, D_2) \subset \mathcal{P}_k$ induces quadratic functions with only one quadratic term $x_i x_j$ such that

$$\left.\begin{array}{rcl} qf_{ij}(\boldsymbol{x}) & \equiv & \dfrac{\nu_{i+}(\theta)}{2\sin\theta}\, r2sf(\boldsymbol{x}; C_k, -\boldsymbol{e}_j, \boldsymbol{b}_{i+}(\theta)) + \dfrac{\nu_{i-}(\theta)}{2\sin\theta}\, r2sf(\boldsymbol{x}; C_k, \boldsymbol{e}_j, \boldsymbol{b}_{i-}(\theta)) \\ & = & \boldsymbol{x}^T \boldsymbol{e}_i \boldsymbol{e}_j^T \boldsymbol{x} + (\boldsymbol{a}_{ij}^T \boldsymbol{x} + b_{ij}) \quad (i, j = 1, 2, \ldots, n). \end{array}\right\} \tag{6.5}$$

The above $\boldsymbol{a}_{ij} \in R^n$ and $b_{ij} \in R$ are expressed as

$$\left.\begin{array}{rl} \boldsymbol{a}_{ij} = & \frac{1}{2\tan\theta}\ \{\alpha(C_0, -\boldsymbol{e}_j) + \alpha(C_0, \boldsymbol{e}_j)\}\ \boldsymbol{c} \\ & \quad + \frac{1}{2}\{\alpha(C_0, -\boldsymbol{e}_j) - \alpha(C_0, \boldsymbol{e}_j)\}\ \boldsymbol{e}_i \\ & \qquad + \frac{1}{2\sin\theta}\{\nu_{i-}(\theta)\alpha(C_k, \boldsymbol{b}_{i-}(\theta)) - \nu_{i+}(\theta)\alpha(C_k, \boldsymbol{b}_{i+}(\theta))\}\ \boldsymbol{e}_j, \\ & \\ b_{ij} = & -\frac{1}{2\sin\theta}\{\nu_{i+}(\theta)\alpha(C_0, -\boldsymbol{e}_j)\alpha(C_k, \boldsymbol{b}_{i+}(\theta)) + \nu_{i-}(\theta)\alpha(C_0, \boldsymbol{e}_j)\alpha(C_k, \boldsymbol{b}_{i-}(\theta))\}. \end{array}\right\} \tag{6.6}$$

Adding some two functions of $\mathcal{P}^2(C_k, D_1, D_2)$ also generates such quadratic functions with only one quadratic term $-x_i x_j$ that

$$\left.\begin{array}{rcl} qf'_{ij}(\boldsymbol{x}) & \equiv & \dfrac{\nu_{i+}(\theta)}{2\sin\theta}\, r2sf(\boldsymbol{x}; C_k, \boldsymbol{e}_j, \boldsymbol{b}_{i+}(\theta)) + \dfrac{\nu_{i-}(\theta)}{2\sin\theta}\, r2sf(\boldsymbol{x}; C_k, -\boldsymbol{e}_j, \boldsymbol{b}_{i-}(\theta)) \\ & = & -\boldsymbol{x}^T \boldsymbol{e}_i \boldsymbol{e}_j^T \boldsymbol{x} + (\boldsymbol{a}'^T_{ij} \boldsymbol{x} + b'_{ij}) \quad (i, j = 1, 2, \ldots, n). \end{array}\right\} \tag{6.7}$$

The above $\boldsymbol{a}'_{ij} \in R^n$ and $b'_{ij} \in R$ of $qf'_{ij}(\boldsymbol{x})$ have similar expressions to $\boldsymbol{a}_{ij}$ and b_{ij} of (6.6), respectively. We omit the detailed description for them. The definitions of $qf_{ij}(\boldsymbol{x})$ and $qf'_{ij}(\boldsymbol{x})$, shown as (6.5) and (6.7) respectively, indicate that $qf_{ij}(\boldsymbol{x}),\ qf'_{ij}(\boldsymbol{x}) \in \text{c.cone}(\mathcal{P}_F \cup \mathcal{P}_k)$ for $i, j = 1, \ldots, n$. Hence, from Lemma 2.2.3 (ii), we see that $\widehat{F}^L(C_0, \mathcal{P}_F \cup \mathcal{P}_k)$ includes the following linear function $g_\ell(\boldsymbol{x})$, $\ell = 1, \ldots, m$.

$$\left.\begin{array}{rcl} g_\ell(\boldsymbol{x}) \equiv & & (\gamma_\ell + 2\boldsymbol{q}_\ell^T \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{Q}_\ell \boldsymbol{x}) + \sum\limits_{(i,j)} Q_{\ell+}^{(i,j)}\ qf'_{ij}(\boldsymbol{x}) - \sum\limits_{(i,j)} Q_{\ell-}^{(i,j)}\ qf_{ij}(\boldsymbol{x}) \\ & = & (\gamma_\ell + \sum\limits_{(i,j)} Q_{\ell+}^{(i,j)} b'_{ij} - \sum\limits_{(i,j)} Q_{\ell-}^{(i,j)} b_{ij}) + (2\boldsymbol{q}_\ell + \sum\limits_{(i,j)} Q_{\ell+}^{(i,j)}\ \boldsymbol{a}'_{ij} - \sum\limits_{(i,j)} Q_{\ell-}^{(i,j)}\ \boldsymbol{a}_{ij})^T \boldsymbol{x}, \end{array}\right\} \tag{6.8}$$

where the (i,j)th element of the matrix $\boldsymbol{Q}_\ell$ is $Q_\ell^{(i,j)}$ and

$$Q_{\ell+}^{(i,j)} = \begin{cases} Q_\ell^{(i,j)} & \text{if } Q_\ell^{(i,j)} > 0 \\ 0 & \text{otherwise,} \end{cases} \qquad Q_{\ell-}^{(i,j)} = \begin{cases} Q_\ell^{(i,j)} & \text{if } Q_\ell^{(i,j)} < 0 \\ 0 & \text{otherwise.} \end{cases}$$

We can regard that $g_\ell(\boldsymbol{x}) \leq 0$ gives a linear relaxation for a quadratic constraint $\gamma_\ell + 2\boldsymbol{q}_\ell^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_\ell\boldsymbol{x} \leq 0$ of (6.1). Moreover, we expect that $\mathcal{P}^2(C_k, D_1, D_2)$ generates much tightly relaxed linear constraints than $g_\ell(\boldsymbol{x}) \leq 0$ for the constraint $\gamma_\ell + 2\boldsymbol{q}_\ell^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_\ell\boldsymbol{x} \leq 0$ in the SILP relaxation $\widehat{F}^L(C_0, \mathcal{P}_F \cup \mathcal{P}_k)$, and that it generates much tightly relaxed convex constraints than $g_\ell(\boldsymbol{x}) \leq 0$ in the SDP relaxation $\widehat{F}(C_0, \mathcal{P}_F \cup \mathcal{P}_k)$.

As shown above, all quadratic functions $\gamma_\ell + 2\boldsymbol{q}_\ell^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_\ell\boldsymbol{x}$ $(\ell = 1, \dots, m)$ of $\mathcal{P}_F$ are convexified or linearized in C_{k+1} through the use of quadratic functions of $\mathcal{P}^2(C_k, D_1, D_2) \subset \mathcal{P}_k$. The linear functions of $\mathcal{P}^L(C_0, D_1) \subset \mathcal{P}_k$ also play an important role to construct C_{k+1}. See Remark 2.3.6 of Chapter 2, for the role of $\mathcal{P}^L(C_0, D_1)$.

6.2 Parallel Successive Convex Relaxation Methods

In Step 2 of Algorithm 6.1.1, $2n$ SDPs or LPs including $4n^2$ additional quadratic constraints are generated for constructing $\mathcal{P}^2(C_k, D_1, D_2)$. It should be emphasized that these $2n$ problems are independent and we can process them in parallel. Thus, in this section we consider parallel computation for some SDPs or LPs.

Parallel computation is much help to reduce computational time drastically, while $4n^2$ constraints in each SDP or LP become an obstacle when we solve larger dimensional QOPs (6.1). In Section 6.2.1, we design new SCRMs so that each SDP or LP has fewer number of constraints than $4n^2$, though they might increase the number of problems to be solved at each iteration. Then, in Section 6.2.2, we will modify Algorithm 6.1.1 and present a parallel algorithm for the client-server based computing system.

6.2.1 An Effective Technique for Reducing Inequalities

At the kth iteration of Algorithm 6.1.1, SDPs or LPs with a feasible region C_k are constructed in Step 2. Note that C_{k+1} consists of $\mathcal{P}_k = \mathcal{P}^L(C_0, D_1) \cup \mathcal{P}^2(C_k, D_1, D_2)$ and $\mathcal{P}_F$. In order to reduce the number of constraints of each SDP or LP, we devise another candidate for D_1, $D_2 = D(\theta)$ and $\mathcal{P}^2(C_k, D_1, D_2)$.

Here we introduce new notation. Let $\lambda_1^\ell, \dots, \lambda_n^\ell$ $(\ell = 1, 2, \dots, m)$ denote n eigenvalues of matrices $\boldsymbol{Q}_\ell$ $(\ell = 1, 2, \dots, m)$ which appear in the quadratic constraints $\gamma_\ell + 2\boldsymbol{q}_\ell^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_\ell\boldsymbol{x} \leq 0$ $(\ell = 1, 2, \dots, m)$ of QOP (6.1). And, let $\boldsymbol{\Lambda}_\ell$ $(\ell = 1, 2, \dots, m)$ denote a diagonal matrix $diag(\lambda_1^\ell, \dots, \lambda_n^\ell)$. For each matrix $\boldsymbol{Q}_\ell$, there exists a real orthogonal matrix $\boldsymbol{P}_\ell$ such that $\boldsymbol{P}_\ell^T\boldsymbol{Q}_\ell\boldsymbol{P}_\ell = \boldsymbol{\Lambda}_\ell$. Using the above notation, we define the sets $I_+(\ell)$ and $I_+(\ell)$ $(\ell = 1, \dots, m)$

as

$$\begin{array}{rcl} I_+(\ell) & \equiv & \text{the set of indices corresponding to positive diagonal} \\ & & \text{elements of } \boldsymbol{\Lambda}_\ell \text{, that is, } \lambda_i^\ell > 0 \text{ for } \forall i \in I_+(\ell), \\ I_-(\ell) & \equiv & \text{the set of indices corresponding to negative diagonal} \\ & & \text{elements of } \boldsymbol{\Lambda}_\ell \text{, that is, } \lambda_j^\ell < 0 \text{ for } \forall j \in I_-(\ell). \end{array}$$

From the definition, we see that $I_+(\ell),\ I_-(\ell) \subseteq \{1, 2, \ldots, n\}$ and $I_+(\ell) \cap I_-(\ell) = \emptyset$.

Next, we define new vectors with a parameter $\theta \in (0, \pi/2]$:

$$\nu_{i+}^\ell(\theta) = \|\boldsymbol{c}\cos\theta + (\boldsymbol{P}_\ell\boldsymbol{e}_i)\sin\theta\|, \quad \nu_{i-}^\ell(\theta) = \|\boldsymbol{c}\cos\theta - (\boldsymbol{P}_\ell\boldsymbol{e}_i)\sin\theta\|,$$

$$\boldsymbol{b}_{i+}^\ell(\theta) = \frac{\boldsymbol{c}\cos\theta + (\boldsymbol{P}_\ell\boldsymbol{e}_i)\sin\theta}{\nu_{i+}^\ell(\theta)}, \qquad \boldsymbol{b}_{i-}^\ell(\theta) = \frac{\boldsymbol{c}\cos\theta - (\boldsymbol{P}_\ell\boldsymbol{e}_i)\sin\theta}{\nu_{i-}^\ell(\theta)},$$

and propose different constructions of D_1, $D_2 = D(\theta)$ and $\mathcal{P}^2(C_k, D_1, D_2)$ for the SDP relaxation as

$$\left.\begin{array}{rcl} D_1^S & \equiv & \{\boldsymbol{P}_\ell\boldsymbol{e}_i, -\boldsymbol{P}_\ell\boldsymbol{e}_i,\ i \in I_-(\ell),\ \ell = 1, \ldots, m\}, \\ D_2^S & \equiv & \{\boldsymbol{b}_{i+}^\ell(\theta), \boldsymbol{b}_{i-}^\ell(\theta),\ i \in I_-(\ell),\ \ell = 1, \ldots, m\}, \\ \mathcal{P}_S^2(C_k, D_1^S, D_2^S) & = & \left\{\begin{array}{l} r2sf(\boldsymbol{x}; C_k, -\boldsymbol{P}_\ell\boldsymbol{e}_i, \boldsymbol{b}_{i+}^\ell(\theta)), \quad r2sf(\boldsymbol{x}; C_k, \boldsymbol{P}_\ell\boldsymbol{e}_i, \boldsymbol{b}_{i-}^\ell(\theta)) \\ i \in I_-(\ell),\ \ell = 1, \ldots, m \end{array}\right\}, \end{array}\right\} \tag{6.9}$$

and for the SILP relaxation as

$$\left.\begin{array}{rcl} D_1^L & \equiv & \{\boldsymbol{P}_\ell\boldsymbol{e}_i, -\boldsymbol{P}_\ell\boldsymbol{e}_i,\ i \in I_-(\ell) \cup I_+(\ell),\ \ell = 1, \ldots, m\}, \\ D_2^L & \equiv & \{\boldsymbol{b}_{i+}^\ell(\theta), \boldsymbol{b}_{i-}^\ell(\theta),\ i \in I_-(\ell) \cup I_+(\ell),\ \ell = 1, \ldots, m\}, \\ \mathcal{P}_L^2(C_k, D_1^L, D_2^L) & = & \left\{\begin{array}{l} r2sf(\boldsymbol{x}; C_k, -\boldsymbol{P}_\ell\boldsymbol{e}_i, \boldsymbol{b}_{i+}^\ell(\theta)), \quad r2sf(\boldsymbol{x}; C_k, \boldsymbol{P}_\ell\boldsymbol{e}_i, \boldsymbol{b}_{i-}^\ell(\theta)) \\ r2sf(\boldsymbol{x}; C_k, \boldsymbol{P}_\ell\boldsymbol{e}_j, \boldsymbol{b}_{j+}^\ell(\theta)), \ r2sf(\boldsymbol{x}; C_k, -\boldsymbol{P}_\ell\boldsymbol{e}_j, \boldsymbol{b}_{j-}^\ell(\theta)) \\ i \in I_-(\ell),\ j \in I_+(\ell),\ \ell = 1, \ldots, m \end{array}\right\}. \end{array}\right\} \tag{6.10}$$

We designate the SDP relaxation method which takes $\mathcal{P}_S^2(C_k, D_1^S, D_2^S)$ for $\mathcal{P}^2(C_k, D_1, D_2)$ as *DLSSDP-diag*, and the SILP relaxation method which takes $\mathcal{P}_L^2(C_k, D_1^L, D_2^L)$ for $\mathcal{P}^2(C_k, D_1, D_2)$ as *DLSSILP-diag*.

We will show that each quadratic function $\gamma_\ell + 2\boldsymbol{q}_\ell^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_\ell\boldsymbol{x}$ $(\ell = 1, \ldots, m)$ of $\mathcal{P}_F$ can be convexified in the relaxation $\widehat{F}(C_0, \mathcal{P}_F \cup \mathcal{P}_k)$ through the use of quadratic functions of $\mathcal{P}_S^2(C_k, D_1^S, D_2^S) \subset \mathcal{P}_k$, and also, that each quadratic function of $\mathcal{P}_F$ can be linearized in the relaxation $\widehat{F}^L(C_0, \mathcal{P}_F \cup \mathcal{P}_k)$ through the use of quadratic functions of $\mathcal{P}_L^2(C_k, D_1^L, D_2^L) \subset \mathcal{P}_k$.

Firstly, we consider DLSSDP-diag. Here note that all matrices $\boldsymbol{Q}_\ell$ $(\ell = 1, \ldots, m)$ of quadratic constraints can be expressed as $\boldsymbol{Q}_\ell = \boldsymbol{Q}_\ell^+ + \boldsymbol{Q}_\ell^-$ using positive definite matrices $\boldsymbol{Q}_\ell^+$ $(\ell = 1, \ldots, m)$ and negative definite matrices $\boldsymbol{Q}_\ell^-$ $(\ell = 1, \ldots, m)$;

$$\boldsymbol{Q}_\ell^+ \equiv \sum_{i \in I_+(\ell)} \lambda_i^\ell (\boldsymbol{P}_\ell\boldsymbol{e}_i)(\boldsymbol{P}_\ell\boldsymbol{e}_i)^T, \quad \text{and} \quad \boldsymbol{Q}_\ell^- \equiv \sum_{i \in I_-(\ell)} \lambda_i^\ell (\boldsymbol{P}_\ell\boldsymbol{e}_i)(\boldsymbol{P}_\ell\boldsymbol{e}_i)^T.$$

Adding some two functions of $\mathcal{P}^2_S(C_k, D^S_1, D^S_2)$ induces such quadratic functions that

$$\begin{aligned} qf^\ell_i(\boldsymbol{x}) &\equiv \frac{\nu^\ell_{i+}(\theta)}{2\sin\theta}\, r2sf(\boldsymbol{x}; C_k, -\boldsymbol{P}_\ell \boldsymbol{e}_i, \boldsymbol{b}^\ell_{i+}(\theta)) + \frac{\nu^\ell_{i-}(\theta)}{2\sin\theta}\, r2sf(\boldsymbol{x}; C_k, \boldsymbol{P}_\ell \boldsymbol{e}_i, \boldsymbol{b}^\ell_{i-}(\theta)) \\ &= \boldsymbol{x}^T(\boldsymbol{P}_\ell \boldsymbol{e}_i)(\boldsymbol{P}_\ell \boldsymbol{e}_i)^T\boldsymbol{x} + (\boldsymbol{a}^{\ell T}_i \boldsymbol{x} + b^\ell_i) \quad (\ell = 1,\ldots,m,\ i \in I_-(\ell)). \end{aligned}$$

For simplicity, we omit the precise description for a vector $\boldsymbol{a}^\ell_i \in R^n$ and a scalar $b^\ell_i \in R$ included in $qf^\ell_i(\boldsymbol{x})$. Then, using the quadratic functions $qf^\ell_i(\boldsymbol{x})$ $(\ell = 1,\ldots,m,\ i \in I_-(\ell))$, any quadratic function $\gamma_\ell + 2\boldsymbol{q}^T_\ell \boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_\ell \boldsymbol{x}$ of the constraints in QOP (6.1) can be made convex as follows.

$$\left.\begin{aligned} &(\gamma_\ell + 2\boldsymbol{q}^T_\ell \boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_\ell \boldsymbol{x}) - \sum_{i\in I_-(\ell)} \lambda^\ell_i\, qf^\ell_i(\boldsymbol{x}) \\ = \;& (\gamma_\ell - \sum_{i\in I_-(\ell)} \lambda^\ell_i\, b^\ell_i) + (2\boldsymbol{q}_\ell - \sum_{i\in I_-(\ell)} \lambda^\ell_i\, \boldsymbol{a}^\ell_i)^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}^+_\ell \boldsymbol{x}, \\ & \qquad\qquad\qquad\qquad (\ell = 1,\ldots,m). \end{aligned}\right\} \tag{6.11}$$

Hence, to construct the convex relaxation C_{k+1}, the quadratic functions $\mathcal{P}^2_S(C_k, D^S_1, D^S_2)$ relax all nonconvex quadratic constraints of QOP (6.1) into convex ones.

Secondly, consider DLSSILP-diag. By summing up other two functions of $\mathcal{P}^2_L(C_k, D^L_1, D^L_2)$ such that

$$\begin{aligned} qf'^\ell_i(\boldsymbol{x}) &\equiv \frac{\nu^\ell_{i+}(\theta)}{2\sin\theta}\, r2sf(\boldsymbol{x}; C_k, \boldsymbol{P}_\ell \boldsymbol{e}_i, \boldsymbol{b}^\ell_{i+}(\theta)) + \frac{\nu^\ell_{i-}(\theta)}{2\sin\theta}\, r2sf(\boldsymbol{x}; C_k, -\boldsymbol{P}_\ell \boldsymbol{e}_i, \boldsymbol{b}^\ell_{i-}(\theta)) \\ &= -\boldsymbol{x}^T(\boldsymbol{P}_\ell \boldsymbol{e}_i)(\boldsymbol{P}_\ell \boldsymbol{e}_i)^T\boldsymbol{x} + (\boldsymbol{a}'^{\ell T}_i \boldsymbol{x} + b'^\ell_i) \quad (\ell = 1,\ldots,m,\ i \in I_+(\ell)), \end{aligned}$$

where $\boldsymbol{a}'^\ell_i \in R^n$ and $b'^\ell_i \in R$, and by adding the above quadratic functions to (6.11), we get linear functions such that

$$\left.\begin{aligned} &(\gamma_\ell + 2\boldsymbol{q}^T_\ell \boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_\ell \boldsymbol{x}) + \sum_{i\in I_+(\ell)} \lambda^\ell_i\, qf'^\ell_i(\boldsymbol{x}) - \sum_{i\in I_-(\ell)} \lambda^\ell_i\, qf^\ell_i(\boldsymbol{x}) \\ = \;& (\gamma_\ell + \sum_{i\in I_+(\ell)} \lambda^\ell_i\, b'^\ell_i - \sum_{i\in I_-(\ell)} \lambda^\ell_i\, b^\ell_i) + (2\boldsymbol{q}_\ell + \sum_{i\in I_+(\ell)} \lambda^\ell_i\, \boldsymbol{a}'^\ell_i - \sum_{i\in I_-(\ell)} \lambda^\ell_i\, \boldsymbol{a}^\ell_i)^T\boldsymbol{x}, \\ & \qquad\qquad\qquad\qquad (\ell = 1,\ldots,m). \end{aligned}\right\} \tag{6.12}$$

The linear functions of the set c.cone$(\mathcal{P}_F \cup \mathcal{P}_k) \cap \mathcal{L}$ provide linear relaxations for all quadratic functions of $\mathcal{P}_F$. Therefore, the definitions of $\mathcal{P}^2_S(C_k, D^S_1, D^S_2)$ and $\mathcal{P}^2_L(C_k, D^L_1, D^L_2)$ are reasonable to make the convex relaxation C_{k+1} of the nonconvex feasible region F of QOP (6.1).

Table 6.1 shows the number of SDPs or LPs to be solved at every iteration, and the number of constraints each problem has, comparing four versions of SCRMs. $|P|$ denotes the number of elements belonging to the set P. The entries of Table 6.1 are computed as

$$\left\{\begin{array}{lcl} \#\text{SDP} & = & |D_2|, \\ \#\text{LP} & = & |D_2|, \\ \#\text{const.} & = & |\mathcal{P}^2(C_k, D_1, D_2)|. \end{array}\right.$$

Table 6.1: Comparison among four SCRM methods

methods	#SDP	#LP	#const.
DLSSDP	$2n$		$4n^2$
DLSSDP-diag	$2\sum_{\ell=1}^{m}\lvert I_-(\ell)\rvert$		$2\sum_{\ell=1}^{m}\lvert I_-(\ell)\rvert$
DLSSILP		$2n$	$4n^2$
DLSSILP-diag		$2\sum_{\ell=1}^{m}(\lvert I_+(\ell)\rvert + \lvert I_-(\ell)\rvert)$	$2\sum_{\ell=1}^{m}(\lvert I_+(\ell)\rvert + \lvert I_-(\ell)\rvert)$

It should be noted that $\sum_{\ell=1}^{m}|I_-(\ell)| \leq \sum_{\ell=1}^{m}(|I_+(\ell)| + |I_-(\ell)|) \leq mn$. Hence, if $m \leq 2n$ holds between the number of constraints m and the number of variables n for QOP (6.1), "#const." of DLSSDP-diag (or DLSSILP-diag) is fewer than that of DLSSDP (or DLSSILP). For the entries in the columns "#SDP" and "#LP", we don't know whether $2\sum_{\ell=1}^{m}|I_-(\ell)|$ and $2\sum_{\ell=1}^{m}(|I_+(\ell)| + |I_-(\ell)|)$ are larger than $2n$. Concerning our test problems of (6.1) used in numerical experiments, the number of SDPs (or LPs) to be solved in DLSSDP-diag (or DLSSILP-diag) is larger than that in DLSSDP (or DLSSILP), while each problem generated in the former has much fewer constraints than that in the latter. We can confirm this by Tables 6.4 and 6.5 of Section 6.3.2.

Remark 6.2.1. In this section, we have introduced DLSSDP-diag and DLSSILP-diag, to decrease the number of quadratic constraints in SDPs and LPs generated by Algorithm 6.1.1 (Steps 0 and 2). There is still plenty of room for improvement. To save the amount of work at each iteration of Algorithm 6.1.1, we can choose less quadratic functions for $\mathcal{P}^2(C_k, D_1, D_2)$; for example, we take

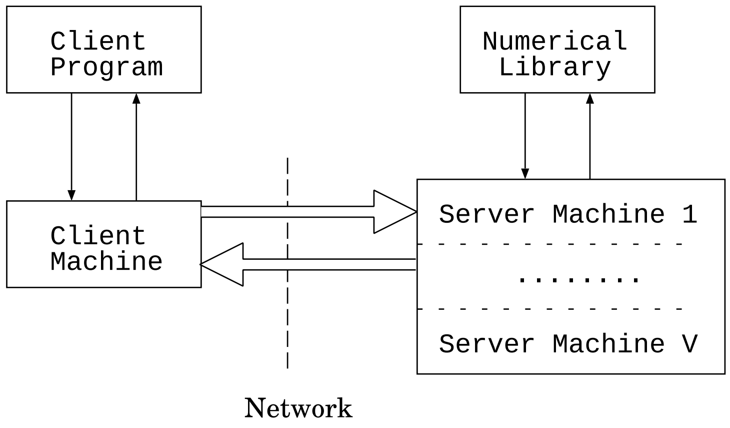
$$\begin{aligned}
\widetilde{\mathcal{P}}^2_S(C_k, D_1^S, D_2^S) &= \left\{ \begin{array}{l} r2sf(\boldsymbol{x}; C_k, -\boldsymbol{P}_\ell \boldsymbol{e}_i, \boldsymbol{b}^\ell_{i+}(\theta)) + r2sf(\boldsymbol{x}; C_k, \boldsymbol{P}_\ell \boldsymbol{e}_i, \boldsymbol{b}^\ell_{i-}(\theta)) \\ i \in I_-(\ell),\ \ell = 1, \ldots, m \end{array} \right\}, \\
\widetilde{\mathcal{P}}^2_L(C_k, D_1^L, D_2^L) &= \left\{ \begin{array}{l} r2sf(\boldsymbol{x}; C_k, -\boldsymbol{P}_\ell \boldsymbol{e}_i, \boldsymbol{b}^\ell_{i+}(\theta)) + r2sf(\boldsymbol{x}; C_k, \boldsymbol{P}_\ell \boldsymbol{e}_i, \boldsymbol{b}^\ell_{i-}(\theta)) \\ r2sf(\boldsymbol{x}; C_k, \boldsymbol{P}_\ell \boldsymbol{e}_j, \boldsymbol{b}^\ell_{j+}(\theta)) + r2sf(\boldsymbol{x}; C_k, -\boldsymbol{P}_\ell \boldsymbol{e}_j, \boldsymbol{b}^\ell_{j-}(\theta)) \\ i \in I_+(\ell),\ j \in I_-(\ell),\ \ell = 1, \ldots, m \end{array} \right\},
\end{aligned}$$

instead of $\mathcal{P}^2_S(C_k, D_1^S, D_2^S)$ defined by (6.9) and $\mathcal{P}^2_L(C_k, D_1^L, D_2^L)$ defined by (6.10), respectively. The number of quadratic functions in $\widetilde{\mathcal{P}}^2_S(C_k, D_1^S, D_2^S)$ is $\sum_{\ell=1}^{m}|I_-(\ell)|$, which is half of that in $\mathcal{P}^2_S(C_k, D_1^S, D_2^S)$. Similarly, $|\widetilde{\mathcal{P}}^2_L(C_k, D_1^L, D_2^L)| = \frac{1}{2}|\mathcal{P}^2_L(C_k, D_1^L, D_2^L)|$ holds. However, reducing constraints in SDPs (or LPs) might rather weaken the accuracy of upper bounds. Consequently, the amount of computation and the accuracy of the relaxation should be balanced properly.

6.2.2 A Parallel Algorithm

Algorithm 6.1.1 generates a large number of SDPs or LPs simultaneously at each iteration and solves them sequentially. Therefore, by implementing SCRMs on a parallel computing

Figure 6.1: The client-server based computing system



system, we can solve some SDPs or LPs simultaneously. Here we will modify Algorithm 6.1.1 for parallel computing on a Ninf (network based information library for high performance computing) system [50, 52]. The basic Ninf system supports client-server based computing as Figure 6.1 shows, and provides a global network-wide computing infrastructure developed for high-performance numerical computation services. It intends not only to exploit high performance in global network parallel computing, but also to provide a simple programming interface similar to conventional function calls in existing languages. We employ the SDPA [20] as an SDP solver, and also discuss parallel execution of the SDPA on the Ninf. The SDPA on the Ninf enjoys the following features: 1. Solving many SDPs simultaneously without complicated implementation. 2. Computational resources including hardware, software and scientific data distributed across a wide area network are available.

Here we suppose that we have a computer system consisting of V $(\leq |D_2|)$ processors; if we have more than $|D_2|$ processors available, we only use the first $|D_2|$. The kth iteration of new parallel algorithm constructs SDPs or LPs as

$$\alpha(C_k, \boldsymbol{d}) = \max\{\boldsymbol{d}^T\boldsymbol{x} : \boldsymbol{x} \in C_k\} \ (\forall \boldsymbol{d} \in D_2),$$

and allocate them to V processors. Then, each processor handles roughly $|D_2|/V$ problems, which are designated by the client machine.

We will provide parallel implementation of Algorithm 6.1.1. In the following algorithm, the work of the client machine and that of each server machine are described together, and each step of Algorithm 6.2.2 is discriminated by the sign (C) or (S); (C) stands for the client machine and (S) for server machines.

Algorithm 6.2.2. (parallel implementation of discretized-localized SCRMs)

Step 0 (C): Define D_1 and $D_2 = D(\theta)$ with some value θ. Assign each $\boldsymbol{d} \in D_1 \cup D_2$ to an idle processor among V processors, and send data of $\boldsymbol{d}$ and C_0 to the processor.

Step 1 (S): Compute $\alpha(C_0, \boldsymbol{d}) = \max\{\boldsymbol{d}^T\boldsymbol{x} : \boldsymbol{x} \in C_0\}$ for some $\boldsymbol{d} \in D_1 \cup D_2$ designated by the client machine. Return $\alpha(C_0, \boldsymbol{d})$ to the client.

Let $k = 0$.

Step 2 (C): If $C_k = \emptyset$ then stop. Compute an upper bound ζ_k for the maximum objective function value of QOP (6.1) by $\zeta_k = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in C_k\}$. If ζ_k satisfies the termination criteria, then stop.

Step 3 (C): Define the set $D(\theta)$ with some value θ. Allocate each $\boldsymbol{d} \in D(\theta)$ to an idle processor, and send the data of $\boldsymbol{d}$, C_0, D_1, $\alpha(C_0, \boldsymbol{d})$ $(\forall \boldsymbol{d} \in D_1)$, D_2 and $\alpha(C_k, \boldsymbol{d})$ $(\forall \boldsymbol{d} \in D_2)$ to it.

Step 4 (S): Generate $\mathcal{P}_k = \mathcal{P}^L(C_0, D_1) \cup \mathcal{P}^2(C_k, D_1, D_2)$, and define C_{k+1}.

Step 5 (S): Compute $\alpha(C_{k+1}, \boldsymbol{d}) = \max\{\boldsymbol{d}^T\boldsymbol{x} : \boldsymbol{x} \in C_{k+1}\}$. Return the value $\alpha(C_{k+1}, \boldsymbol{d})$ to the client.

Step 6 (C): Set $D_2 = D(\theta)$. Let $k = k + 1$ and go to Step 2 (C).

In Step 4 (S), the same set $\mathcal{P}_k$ of quadratic functions is generated in each server machine. This redundant work is to reduce the transmitting time between the client machine and each server machine. From our numerical results, we found that after the construction of $\mathcal{P}_k$ on a client machine, sending data of $\mathcal{P}_k$ from the client machine to each server machine took extremely much communication time. Therefore, it is better to reduce the amount of data to be transmitted through the network as much as possible.

Remark 6.2.3. In numerical experiments, we add the objective direction $\boldsymbol{c}$ to the set D_2 and solve $\alpha(C_k, \boldsymbol{c}) = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in C_k\}$ in some server machine. Then, in Step 2 (C), we find $\alpha(C_k, \boldsymbol{c})$ among $\alpha(C_k, \boldsymbol{d})$ with $\forall \boldsymbol{d} \in D_2$, and set $\zeta_k = \alpha(C_k, \boldsymbol{c})$. Therefore, the work of the client machine is only to assign each SDP (or SILP) to one of the V processors, and to check the termination criteria. The client machine can avoid the computation for not only solving a bounding problem but constructing C_{k+1}.

Remark 6.2.4. If we have enough processors to handle $|D_2|$ problems, it is better to solve all $(|D_1| + |D_2|)$ problems;

$$\alpha(C_k, \boldsymbol{d}) = \max\{\boldsymbol{d}^T\boldsymbol{x} : \boldsymbol{x} \in C_k\} \quad (\forall \boldsymbol{d} \in D_1 \cup D_2)$$

at every kth iteration, and construct C_{k+1} using $\alpha(C_k, \boldsymbol{d})$ instead of $\alpha(C_0, \boldsymbol{d})$ for $\forall \boldsymbol{d} \in D_1$. Then we can obtain the tighter relaxation C_{k+1} of the nonconvex feasible region F of QOP (6.1).

6.3 Computational Experiments

In this section, we present our four kinds of test problems, describe some implementation details on Algorithms 6.1.1 and 6.2.2, and report some encouraging numerical results.

6.3.1 Test Problems

We summarize some properties of our test problems in Tables 6.2 and 6.3. They consist of four types of problems such as (a) 0-1 integer QOPs, (b) linearly constrained QOPs, (c) bilevel QOPs, and (d) fractional QOPs. In Section 2.1 of Chapter 2, we have shown how to transform the above four types of problems (a), (b), (c) and (d) into QOPs. In our numerical experiments, we applied some SCRMs to the transformed QOP (6.1). The type of each test problem is denoted in the second column of Tables 6.2 and 6.3. The columns n and m denote the number of variables and the number of constraints (not including box constraints) of the transformed QOP (6.1), respectively. The column "#QC" denotes the number of quadratic constraints among m constraints in QOP (6.1). The last column gives the number of local optima for some types of the test problems. We denote "?" for the case that the number of local optima is not available. We know optimum objective function values for all test problems of Tables 6.2 and 6.3 in advance.

We give more precise description for problems (a)-(d) as follows.

(a) 0-1 integer QOP is formulated as

$$\text{(0-1IQOP)} \quad \left| \begin{array}{ll} \min & \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} \\ \text{subject to} & \boldsymbol{x} \in \{0,1\}^n. \end{array} \right.$$

We used the code of Pardalos and Rodgers [42] to generate coefficient matrices $\boldsymbol{Q}$ of the test problems.

(b) Linearly constrained QOP can be expressed as

$$\text{(LCQOP)} \quad \left| \begin{array}{ll} \min & \gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \end{array} \right.$$

where $\boldsymbol{Q} \in R^{n\times n}$, $\boldsymbol{q} \in R^n$, $\boldsymbol{x} \in R^n$, $\boldsymbol{b} \in R^m$ and $\boldsymbol{A} \in R^{m\times n}$. We generate each test problem (LCQOP) by the code of Calamai, Vincente and Judice [13]. Their construction of (LCQOP) provides not only its optimum solution but the number of its local minima.

(c) Bilevel QOP is formulated as

$$\text{(BLQOP)} \quad \left| \begin{array}{lll} \min\limits_{x} & \gamma + 2\boldsymbol{q}^T\boldsymbol{z} + \boldsymbol{z}^T\boldsymbol{Q}\boldsymbol{z} & \\ \text{subject to} & & \\ & \min\limits_{y} & \boldsymbol{z}^T\boldsymbol{R}\boldsymbol{z} \\ & \text{subject to} & \boldsymbol{A}\boldsymbol{z} \leq \boldsymbol{b}, \quad \boldsymbol{z} = \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix}, \end{array} \right.$$

Table 6.2: The test problems (small size)

Problem	type	source	n	m	# QC	# local
01int20	0-1IQOP	[42]	21	21	21	?
01int30	0-1IQOP	[42]	31	31	31	?
LC30-36	LCQOP	[13]	31	46	1	36
LC30-162	LCQOP	[13]	31	46	1	162
LC40-6	LCQOP	[13]	41	61	1	6
LC40-72	LCQOP	[13]	41	61	1	72
LC50-1296	LCQOP	[13]	51	76	1	1296
BLevel3-6	BLQOP	[12]	19	25	10	4
BLevel8-3	BLQOP	[12]	21	22	10	4
Frac20-10	FQOP	–	21	12	1	?
Frac30-15	FQOP	–	31	17	1	?

Table 6.3: The test problems (large size)

Problem	type	source	n	m	# QC	# local
01int50	0-1IQOP	[42]	51	51	51	?
01int55	0-1IQOP	[42]	56	56	56	?
01int60	0-1IQOP	[42]	61	61	61	?
LC60-72	LCQOP	[13]	61	91	1	72
LC70-72	LCQOP	[13]	71	106	1	72
LC80-144	LCQOP	[13]	81	121	1	144
BLevel20-3	BLQOP	[12]	33	22	10	4
BLevel30-4	BLQOP	[12]	47	29	13	8
BLevel40-4	BLQOP	[12]	57	29	13	8
Frac50-20	FQOP	–	51	22	1	?
Frac60-20	FQOP	–	61	22	1	?
Frac70-25	FQOP	–	71	27	1	?

where $\boldsymbol{x} \in R^p$, $\boldsymbol{y} \in R^q$, $\boldsymbol{z} \in R^n$ with $n = p + q$, and $\boldsymbol{A} \in R^{m \times n}$. Let $\boldsymbol{Q}$ and $\boldsymbol{R}$ be symmetric positive semidefinite matrices, *i.e.*, $\boldsymbol{Q} \in \mathcal{S}^n_+$ and $\boldsymbol{R} \in \mathcal{S}^n_+$. We generate test problems of (BLQOP) by the code of Calamai and L.N.Vincente [12].

(d) Fractional QOP is generated as

$$\text{(FQOP)} \left| \begin{array}{ll} \text{min} & g(\boldsymbol{x}) = \dfrac{1/2\ \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}}{\boldsymbol{q}^T\boldsymbol{x} - \gamma} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{q}^T\boldsymbol{x} \geq 3/2\ \gamma. \end{array} \right.$$

Here $\boldsymbol{x} \in R^n$, $\boldsymbol{q} \in R^n$, $\boldsymbol{A} \in R^{m \times n}$, $\boldsymbol{Q} \in \mathcal{S}^n_+$ and $\gamma \equiv 1/2\ \boldsymbol{q}^T\boldsymbol{Q}^{-1}\boldsymbol{q}$. If we take a constant term $\boldsymbol{b} \in R^m$ so that $\boldsymbol{A}\boldsymbol{Q}^{-1}\boldsymbol{q} < \boldsymbol{b}$ holds, the above (FQOP) has an optimum value 1. Indeed, note that (FQOP) is equivalent to the problem finding $\lambda^* \geq 0$ such that $\pi(\lambda^*) = 0$, where

$$\left.\begin{array}{l} \pi(\lambda) = \min\{1/2\ \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} - \lambda\ (\boldsymbol{q}^T\boldsymbol{x} - \gamma) :\ \boldsymbol{x} \in X\}, \\ \text{where } X \equiv \{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b},\ \boldsymbol{q}^T\boldsymbol{x} \geq 3/2\ \gamma\}. \end{array}\right\} \tag{6.13}$$

We see that the following problem:

$$\min\ \{1/2\ \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{q}^T\boldsymbol{x} + \gamma\} \tag{6.14}$$

has an optimum solution $\boldsymbol{x}^* = \boldsymbol{Q}^{-1}\boldsymbol{q}$, since $\boldsymbol{Q}$ is a positive definite matrix. Then, the optimum solution $\boldsymbol{x}^*$ of (6.14) achieves $\pi(1) = 0$ for the problem (6.13), and hence, the problem (FQOP) generated by this technique has the optimum value $\lambda^* = g(\boldsymbol{x}^*) = 1$.

6.3.2 Numerical Results

To make Algorithms 6.1.1 and 6.2.2 implementable, it is necessary to clarify these issues: (a) the value θ for $D_2 = D(\theta)$, (b) termination criteria, and (c) SCRMs to be used.

(a) We start Algorithms 6.1.1 and 6.2.2 by constructing a set $D_2 = D(\theta)$ with $\theta = \pi/2$. If at the kth iteration of Algorithms 6.1.1 and 6.2.2, the decrease $(\zeta_k - \zeta_{k-1})$ of the upper bounds becomes little, we decrease the value θ and update D_2. Otherwise, we use the same set D_2 at the next iteration. Throughout the computational experiments, we use the following replacement rule:

Let $\ell = 0$, $\theta_0 = \pi/2$, $K = 3$ and $\{\sigma_j\}_{j=0}^K$ be a decreasing sequence such that $\{1, \frac{8}{9}, \frac{4}{9}, \frac{2}{9}\}$. If $\ell < K$ and the upper bound ζ_k generated at the kth iteration satisfies

$$\frac{\zeta_{k-1} - \zeta_k}{\max\{|\zeta_k|,\ 1.0\}} < 1.0^{-3} \times \sigma_\ell,$$

then set $\ell = \ell + 1$, and replace D_2 by $D_2 = D(\theta)$ with $\theta = \sigma_\ell\theta_0$.

(b) If $\ell = K$ and $\dfrac{\zeta_{k-1} - \zeta_k}{\max\{|\zeta_k|,\ 1.0\}} < 1.0^{-3} \times \sigma_K$, we terminate the algorithm. Then, ζ_k is the best upper bound found by Algorithm 6.1.1 or 6.2.2.

(c) We have chosen two SCRMs of DLSSILP and DLSSDP-diag, and implemented DLSSILP on a single processor and DLSSDP-diag on multiple processors. Note that DLSSILP is coincident with a practical SCRM ("DLSSILP") proposed in Chapter 4.

These programs of Algorithms 6.1.1 and 6.2.2 were coded in ANSI C++ language. We used SDPA Version 5.0 [21] as an SDP solver to compute $\alpha(C_0, \boldsymbol{d})$ for $\forall \boldsymbol{d} \in D_1$ and $\alpha(C_k, \boldsymbol{d})$ for $\forall \boldsymbol{d} \in D_2$ in Algorithm 6.2.2, and used CPLEX Version 6.5 as an LP solver to compute them in Algorithm 6.1.1.

Our experiments were conducted to see the following three factors: (i) comparison between DLSSILP and DLSSDP-diag with respect to the number of problems generated at every iteration and the size of each problem; (ii) the accuracy of upper bounds obtained by DLSSDP-diag, compared with the accuracy of those generated by DLSSILP; (iii) computational efficiency of parallel Algorithm 6.2.2 using 1, 2, 4, 8, 16, 32, 64 and 128 processors.

(i) Tables 6.4 and 6.5 show the number of LPs (# LP $=|D_2|$) generated at each iteration of DLSSILP and the number of SDPs (# SDP $=|D_2|$) of DLSSDP-diag. Also, they show the number of constraints (# tot_const. $= |\mathcal{P}_F| + |\mathcal{P}^L(C_0, D_1)| + |\mathcal{P}^2(C_k, D_1, D_2)|$) in each problem. We see from these tables that SDPs of DLSSDP-diag have much less constraints than LPs of DLSSILP, though the number of SDPs generated by DLSSDP-diag is larger than that of LPs by DLSSILP.

(ii) In Tables 6.6 and 6.7, we summarize numerical results on a parallel implementation of DLSSDP-diag, and in Table 6.8, summarize those on a serial implementation of DLSSILP in terms of the following items:

$r.Err^1$	:	$r.Err$ at the first iteration;
$r.Err^*$	:	$r.Err$ at the last iteration;
iter.	:	the number of iterations each algorithm repeated;
R.time	:	the real time in second;
C.time	:	the cpu time in second.

Here $r.Err$ is the relative error of a solution, *i.e.*, $r.Err = \dfrac{|f_{\rm up} - f_{\rm opt}|}{\max\{|f_{\rm opt}|, 1.0\}}$, $f_{\rm opt}$ is the global optimum value of QOP (6.1) and $f_{\rm up}$ is the best upper bound found by each algorithm.

We ran Algorithm 6.2.2 using 128 processors of 64 server machines and 1 processor of a client machine. We slightly modified Algorithm 6.2.2 according to the suggestion of Remark 6.2.4, and hence, the modified algorithm generates $(|D_1| + |D_2|)$ SDPs at every iteration. Note that the number $(|D_1|+|D_2|)$ is almost twice of #SDP described in Tables 6.4 and 6.5. Tables 6.6 and 6.7 include not only solution information of DLSSDP-diag but time information such as C$\Rightarrow$S (total transmitting time from the client to each server), exec.time (total execution time on Ninf server machines), and

Table 6.4: Problems generated by DLSSILP and DLSSDP-diag for small-sized QOPs

	DLSSILP		DLSSDP-diag	
Problem	# LP	# tot_const.	# SDP	# tot_const.
01int20	40	1621	81	122
01int30	60	3631	117	178
LC30-36	60	3646	61	137
LC30-162	60	3646	61	137
LC40-6	80	6461	81	182
LC40-72	80	6461	81	182
BLevel3-6	36	1321	55	98
BLevel8-3	40	1622	59	101
Frac20-10	40	1612	43	76
Frac30-15	60	3617	63	110

Table 6.5: Problems generated by DLSSILP and DLSSDP-diag for large-sized QOPs

	DLSSILP		DLSSDP-diag	
Problem	# LP	# tot_const.	# SDP	# tot_const.
01int50	100	10051	201	302
01int55	110	12156	221	332
01int60	120	14461	241	362
LC60-72	120	14491	121	272
LC70-72	140	19706	141	317
LC80-144	160	25721	161	362
BLevel20-3	64	4118	83	137
BLevel30-4	92	8493	117	192
BLevel40-4	112	12573	137	222
Frac50-20	100	10022	103	175
Frac60-20	120	14422	123	205
Frac70-25	140	19627	143	240

Table 6.6: Numerical results of DLSSDP-diag on the PC cluster (small-sized QOPs)

Problem	DLSSDP-diag				Time Info (sec.)		
	$r.Err^1$	$r.Err^*$	iter.	R.time (sec.)	C⇒S	exec.time	S⇒C
01int20	8.34	6.23	6	24	0.07	1070	0.06
01int30	6.20	2.98	6	58	0.11	5811	0.10
LC30-36	100.00	5.30	9	28	0.09	1319	0.09
LC30-162	100.00	27.42	18	55	0.18	2790	0.20
LC40-6	100.00	0.89	8	43	0.12	3292	0.13
LC40-72	100.00	4.14	9	52	0.14	3783	0.14
BLevel3-6	6.53	2.44	13	18	0.11	847	0.10
BLevel8-3	6.53	2.45	13	28	0.12	1114	0.14
Frac20-10	89.36	0.92	27	54	0.22	3166	0.18
Frac30-15	89.58	0.88	26	345	0.37	25913	0.35

Table 6.7: Numerical results of DLSSDP-diag on the PC cluster (large-sized QOPs)

Problem	DLSSDP-diag				Time Info (sec.)		
	$r.Err^1$	$r.Err^*$	iter.	R.time (sec.)	C⇒S	exec.time	S⇒C
01int50	107.40	104.74	3	267	0.17	26127	0.12
01int55	100.15	75.37	5	905	0.31	86000	0.23
01int60	105.20	102.53	3	607	0.22	65560	0.15
LC60-72	100.00	3.13	8	171	0.22	12627	0.20
LC70-72	100.00	2.78	8	183	0.27	18601	0.24
LC80-144	123.70	2.94	8	406	0.44	35000	0.32
BLevel20-3	8.51	7.40	8	78	0.11	1939	0.10
BLevel30-4	12.41	8.75	13	167	0.29	11326	0.27
BLevel40-4	12.40	9.08	12	230	0.33	192970	0.27
Frac50-20	89.50	0.94	26	3200	0.75	305002	0.78
Frac60-20	89.53	0.97	26	6318	1.30	734037	0.98
Frac70-25	89.37	1.50	25	14196	1.72	1483764	1.21

Legend : DLSSDP-diag is executed on the PC cluster which is composed of 64 server machines. Each server machine has 2 processors (CPU Pentium III 800MHz) with 640MB memory.

Table 6.8: Numerical results of DLSSILP on a single processor (small-sized QOPs)

Problem	DLSSILP			
	$r.Err^1$	$r.Err^*$	iter.	C.time (sec.)
01int20	51.40	48.84	6	50.90
01int30	1.90	1.90	5	36.53
LC30-36	51.45	38.22	9	185.03
LC30-162	74.45	58.15	11	215.83
LC40-6	38.09	27.45	9	454.22
LC40-72	57.53	44.77	9	548.88
BLevel3-6	100.00	43.99	39	86.50
BLevel8-3	100.00	100.00	5	19.97
Frac20-10	100.00	100.00	5	18.58
Frac30-15	100.00	100.00	5	149.15

Legend : DLSSILP is implemented on one processor of DEC Alpha Workstation (CPU 600 MHz, 1GB memory)

Table 6.9: Computational efficiency by increasing the number of processors

#proc.	LC80-144		Frac50-20	
	R.time (sec.)	ratio	R.time (sec.)	ratio
1	33125	1.00	289259	1.00
2	16473	2.01	145980	1.98
4	8238	4.02	72343	3.99
8	4145	7.99	36272	7.97
16	2099	15.78	18595	15.56
32	1118	29.62	9424	30.69
64	624	53.08	4822	60.00
128	361	91.76	3200	90.39

Legend : DLSSDP-diag is executed on the PC cluster which is composed of 64 server machines. Each server machine has 2 processors (CPU Pentium III 800MHz) with 640MB memory.

S⇒C (total transmitting time from each server to the client). These time data was measured in real time. Since little data are transmitted through a network in the parallel algorithm, we can take no notice of transmitting time between two machines.

Table 6.8 presents our numerical results on a serial implementation of DLSSILP. DLSSILP cannot deal with the large-sized test problems of Table 6.5 except “BLevel20-3” due to the shortage of memory on our computational environment. Thus we show our numerical results of DLSSILP, restricted to the small-sized QOPs

From comparison between Table 6.6 and Table 6.8, we see that the upper bounds of DLSSDP-diag are more accurate than those of DLSSILP in most cases. Especially for fractional QOPs, DLSSDP-diag improves upper bounds significantly, compared with DLSSILP. Therefore we can expect DLSSDP-diag to be a practical bounding method for some difficult nonconvex QOPs, if multiple processors are available. On the other hand, DLSSILP has the merit that it attains an upper bound fast as Table 6.8 shows, though we cannot compare computational time between these two methods. We have no choice but to use different processors for numerical experiments of DLSSDP-diag and DLSSILP due to the commercial license of the CPLEX software.

(iii) We implemented DLSSDP-diag on the PC cluster which is composed of 128 processors. Table 6.9 shows computational efficiency in proportion to the number of processors. We compute “ratio“ which appears in the row of (#proc. $= k$) as $\frac{\text{R.time of (\#proc.=1)}}{\text{R.time of (\#proc.=}k\text{)}}$. The ratio indicates computational efficiency with the use of k processors. If the ratio is sufficiently close to k (=#proc.), we can regard Algorithm 6.2.2 as well paralleled. As Table 6.9 shows, the algorithm is well paralleled with relatively small number of #proc., since the number of SDPs are sufficiently large in comparison with #proc. so that such SDPs are allocated to server machines in balance. Then, computational time taken by each server machine is almost same, and good performance of parallel computation can be attained.

The numerical results demonstrate that compared with DLSSILP, DLSSDP-diag obtained better upper bounds in most test problems. Moreover, DLSSDP-diag made it possible to handle larger dimensional test problems which the existing SCRM (DLSSILP) had not attacked, by decreasing the number of constraints consisting of C_{k+1} considerably. As the first implementation for a parallel SCRM, our numerical experiments are quite satisfactory.

Chapter 7

Conclusions and Future Research

The most general class of QOPs (quadratic optimization problems) of the form (1.2) has received far less attention than linearly constrained quadratic problems (1.1), because of theoretical and practical difficulties in the process of solving (1.2). This thesis focused on the difficult problem (1.2), and proposed quite unconventional solution techniques based on SCRMs (successive convex relaxation methods). We briefly summarize the main results of the thesis, by listing four distinct subjects. Note that the classification below corresponds with the structure of the chapters.

- *Complexity analysis* [31] :
  Given an arbitrary positive number ϵ, we bound the number k of iterations which the conceptual SCRM requires to attain an ϵ-convex-relaxation. The number k is affected by some quantities which are extracted from an input problem (1.2); the diameter of a feasible region F in (1.2), the diameter of an initial relaxed region C_0 for F, a Lipschitz constant, a nonconvexity measure and a nonlinearity measure. The latter three quantities are obtained from quadratic constraints included in QOP (1.2).

- *Practical SCRMs* [62] :
  Practical versions of the discretized-localized SCRMs were implemented. Numerical results demonstrated that the methods could generate relatively good upper bounds for maximum objective function values in most test problems, compared with the reformulation-linearization technique (RLT) [55, 56, 54].

- *Special SCRMs to bilevel programs* [64] :
  A BQOP (bilevel quadratic optimization problem) results in one-level QOPs via two equivalent transformations. An exploitation of the special structure in the transformed QOPs accelerates the SCRM and generates tighter upper bounds for maximum objective function values.

- *Parallel SCRMs* [63] :
  A SCRM suitable to parallel computing was implemented on a PC cluster. In most

benchmark problems, it achieved tighter upper bounds than some other SCRMs proposed in Chapter 4. Moreover, it handled larger-sized problems which other SCRMs had not dealt with.

As we stated in Chapters 1 and 2, our problem (1.2) includes various classes of mathematical programs, and therefore, our SCRMs proposed in Chapters 4 and 6 are applicable to them. However, if such problems (1.2) have some nice structure, we should adapt the general SCRM to the structure, as we designed the special SCRM to BQOPs in Chapter 5.

Here we will sketch several directions for future research. See, Figure 7.1. The studies in two boxes were already completed. We pay attention to the following three studies; (i), (ii) and (iii).

(i) *Algorithm for an ϵ-approximate solution* :
The SCRM presented in the thesis generates an upper bound for QOP (1.2), since we have no theoretical result on the relations between $\epsilon > 0$, $\kappa > 0$ and $\delta > 0$ introduced in Theorem 2.3.3 of Chapter 2. However, by incorporating some SCRM into a branch-and-bound method, we can obtain an ϵ-approximate solution with very small error $\epsilon > 0$ even for difficult classes of nonconvex QOPs (1.2). Our SCRM consumes relatively much computational time in order to achieve a tighter upper bound for the maximum objective function value of (1.2), while the tight bound reduces the number of branching processes, *i.e.,* the number of subproblems to be solved. If we use the SCRM as an upper-bounding procedure in a branch-and-bound method, we propose stopping the SCRM within first several iterations. The upper bound provided by the SCRM with such a new stopping criterion might be tight enough. This is confirmed from Figure 4.1 of Chapter 4, and Figures 5.1 and 5.2 of Chapter 5, since in a sequence of upper bounds obtained by the SCRM, a drastic decrease occurs at an early stage of the execution of the algorithm.

(ii) *More general nonlinear programs* :
In connection with SCRMs, Kojima-Matsumoto-Shida [30] pointed out that a wide class of nonlinear programs can be reduced to nonconvex QOPs (1.2). See Example 2.1.4 of Section 2.1. Their technique makes it possible to extend the SCRMs discussed in the thesis, to a wide class of nonlinear programs. That is, for any SCRM, we could further weaken the assumption of quadratic functions to a wide class of nonlinear functions. Actually, the short note [22] reports preliminary numerical results on a SCRM applied to more general nonlinear programs.

(iii) *Complexity analysis for other SCRMs* :
Chapter 3 investigated computational complexity of the conceptual SCRMs. This study provides a new way of complexity analysis for other SCRMs such as

- conceptual SCRMs for general nonlinear programs,
- discretized-localized SCRMs for quadratic programs.

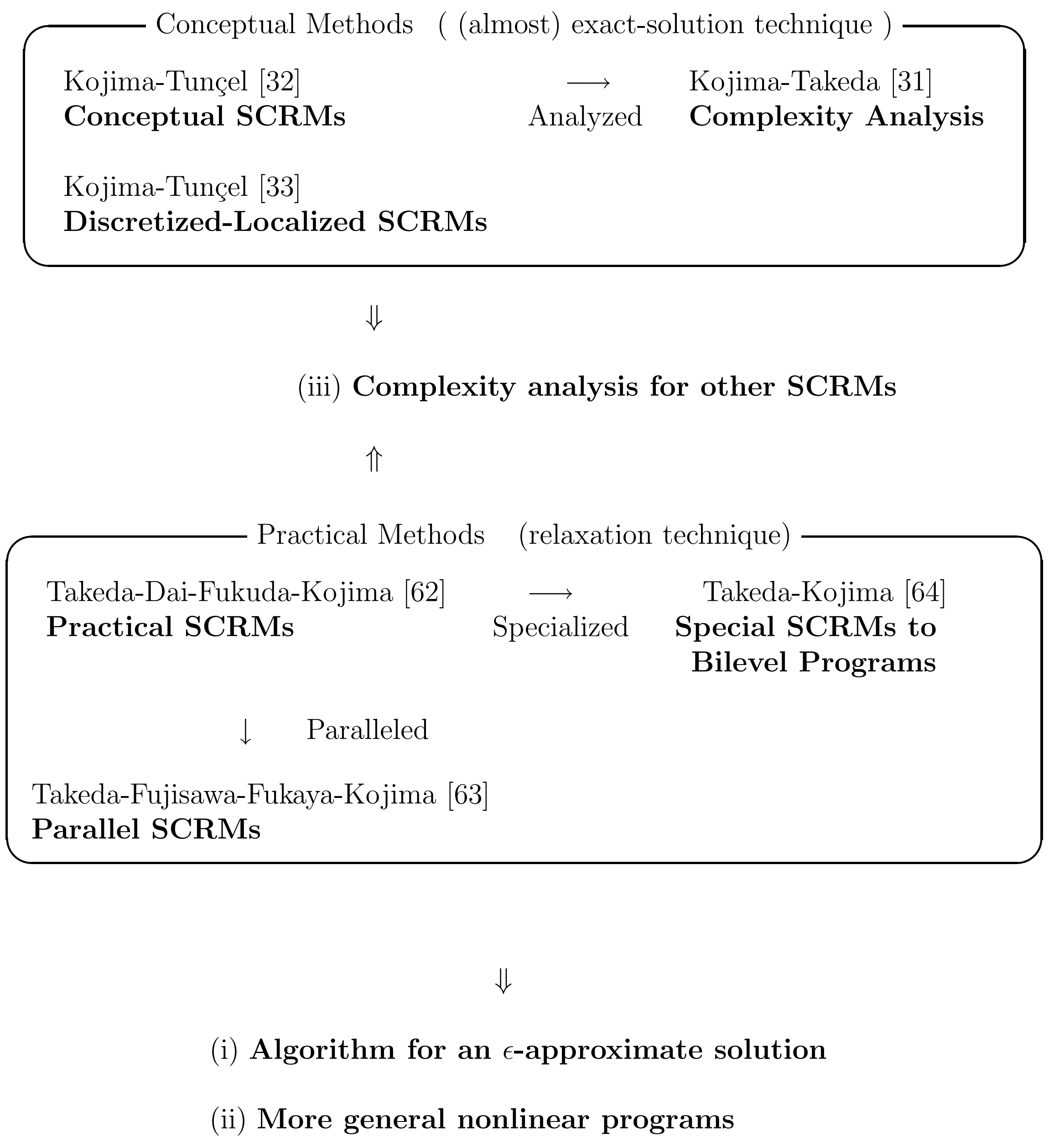

Conceptual Methods ((almost) exact-solution technique)
Kojima-Tunçel [32]
Conceptual SCRMs
Analyzed
Kojima-Takeda [31]
Complexity Analysis
Kojima-Tunçel [33]
Discretized-Localized SCRMs
(iii) Complexity analysis for other SCRMs
Practical Methods (relaxation technique)
Takeda-Dai-Fukuda-Kojima [62]
Practical SCRMs
Specialized
Takeda-Kojima [64]
Special SCRMs to Bilevel Programs
Paralleled
Takeda-Fujisawa-Fukaya-Kojima [63]
Parallel SCRMs
(i) Algorithm for an ϵ-approximate solution
(ii) More general nonlinear programs

Figure 7.1: Future work of SCRMs : (i), (ii) and (iii)

Bibliography

[1] E. Aiyoshi and K. Shimizu, "Hierarchical decentralized systems and its new solution by a barrier method," *IEEE Transactions on Systems, Man and Cybernetics* **SMC-11** (1981) 444–449.

[2] E. Aiyoshi and K. Shimizu, "A solution method for the static constrained Stackelberg problem via penalty method," *IEEE Transactions on Automatic Control* **AC-29** (1984) 1111–1114.

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

[4] F.A. AL-Khayyal and J.E. Falk, "Jointly constrained biconvex programming," *Mathematics of Operations Research* **8** (1983) 273-286.

[5] F.A. AL-Khayyal and C. Larsen, "Global optimization of a quadratic function subject to a bounded mixed integer constraint set," *Annals of Operations Research* **25** (1990) 169-180.

[6] C. Audet, P. Hansen, B. Jaumard and G. Savard, "A branch and cut algorithm for nonconvex quadratically constrained quadratic programming," *Mathematical Programming* **87** (2000) 131–152.

[7] F. Avram and L.M. Wein, "A product design problem in semiconductor manufacturing," *Operations Research* **40** (1992) 999–1017.

[8] J. Balakrishnan, F.R. Jacobs and M.A. Venkataramanan, "Solutions for the constrained dynamic facility layout problem," *European Journal of Operations Research* **57** (1992) 280–286.

[9] E. Balas, S. Ceria and G. Cornuéjols, "A lift-and-project cutting plane algorithm for mixed 0-1 programs," *Mathematical Programming* **58** (1993) 295–323.

[10] J.F. Bard, "Convex two-level optimization," *Mathematical Programming* **40** (1988) 15–27.

[11] R.W. Cottle, J.S. Pang and R.E. Stone, *The Linear Complementarity Problem* (Academic Press, New York, 1992).

[12] P.H. Calamai and L.N. Vicente, “Generating quadratic bilevel programming problems,” *ACM Transactions on Mathematical Software* **20** (1994) 103–122.

[13] P.H. Calamai, L.N. Vincente and J.J. Judice, “A new technique for generating quadratic programming test problems,” *Mathematical Programming* **61** (1993) 215–231.

[14] E.V. Denardo and C.S. Tang, “Linear control of a Markov production system,” *Operations Research* **40** (1992) 259–278.

[15] T.G.W. Epperly and R.E. Swaney, “Branch and bound for global NLP : Iterative LP algorithm and results”, in I.E. Grossmann (ed.), *Global Optimization in Engineering Design* (Kluwer Academic Publishers, Dordrecht, 1996).

[16] J.E. Falk and S.W. Palocsay, “Optimizing the sum of linear fractional functions”, in C.A. Floudas and P.M. Pardalos (eds.), *Recent Advances in Global Optimization* (Princeton University Press, Princeton, 1992).

[17] C.A. Floudas and P.M. Pardalos, *A Collection of Test Problems for Constrained Global Optimization Algorithms, Lecture Notes in Computing Science Vol. 455* (Springer-Verlag, Berlin, 1990).

[18] C.A. Floudas and V. Visweswaran, “Quadratic optimization”, in R. Horst and P.M. Pardalos (eds.), *Handbook of Global Optimization* (Kluwer Academic Publishers, Dordrecht, 1995).

[19] T. Fujie and M. Kojima, “Semidefinite relaxation for nonconvex programs,” *Journal of Global Optimization* **10** (1997) 367–380.

[20] K. Fujisawa, M. Kojima and K. Nakata, “Exploiting sparsity in primal-dual interior-point methods for semidefinite programming,” *Mathematical Programming* **79** (1997) 235–253.

[21] K. Fujisawa, M. Kojima and K. Nakata, “SDPA (Semidefinite Programming Algorithm) – User’s Manual –,” Technical Report B-308, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Meguro, Tokyo, Japan, revised May 1999.

[22] M. Fukuda and M. Kojima, “Approximation of global optimal values of nonconvex programs using Successive Convex Relaxation Method,” *Continuous and Discrete Mathematics for Optimization*, Research Institute for Mathematical Sciences, RIMS Kokyuroku 1114, Kyoto, 1999.

[23] M.X. Goemans, “Semidefinite programming in combinatorial optimization,” *Mathematical Programming* **79** (1997) 143–161.

[24] K.C. Goh, M.G. Safonov and G.P. Papavassilopoulos, “Global optimization for the biaffine matrix inequality problem,” *Journal of Global Optimization* **7** (1995) 365-380.

[25] I.E. Grossmann and A. Kravanja, "Mixed-integer nonlinear programming : A survey of algorithms and applications," in L.T. Biegler, T.F. Coleman, A.R. Conn and F.N Santosa (eds.), *Large-Scale Optimization with Applications, Part II : Design and Control* (Springer-Verlag, Berlin, 1997).

[26] M. Grötschel, L. Lovász and A. Schrijver, *Geometric algorithms and combinatorial optimization* (Springer-Verlag, Berlin, 1988).

[27] R. Horst and H. Tuy, *Global Optimization, Second, Revised Edition* (Springer-Verlag, Berlin, 1992).

[28] S.H. Hum and R.K. Sarin, "Simultaneous product-mix planning, lot sizing and scheduling at bottleneck facilities," *Operations Research* **39** (1991) 296–307.

[29] H. Juel and R.F. Love, "The dual of a generalized minimax location problem," *Annals of Operations Research* **40** (1991) 261–264.

[30] M. Kojima, T. Matsumoto and M. Shida, "Moderate Nonconvexity = Convexity + Quadratic Concavity" Technical Report B-348, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Meguro, Tokyo, Japan, revised April 1999.

[31] M. Kojima and A. Takeda, "Complexity analysis of successive convex relaxation methods for nonconvex sets," Technical Report B-350, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Meguro, Tokyo, Japan, revised July 1999, to appear in *Mathematics of Operations Research.*

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

[33] M. Kojima and L. Tunçel, "Discretization and Localization in Successive Convex Relaxation Methods for Nonconvex Quadratic Optimization Problems," Technical Report B-341, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Meguro, Tokyo, Japan, July 1998, to appear in *Mathematical Programming A.*

[34] H. Konno, P.T. Thach and H. Tuy, *Optimization on low rank nonconvex structures* (Kluwer Academic Publishers, Dordrecht, 1997).

[35] L. Lovász and A. Schrijver, "Cones of matrices and set functions and 0-1 optimization," *SIAM Journal on Optimization* **1** (1991) 166–190.

[36] Z.Q. Luo, J.S. Pang and D. Ralph, *Mathematical Programming with Equilibrium Constraints* (Cambridge University Press, Cambridge, 1996).

[37] G.P. McCormick, "Computability of global solutions to factorable nonconvex programs: part I - convex underestimating problems," *Mathematical Programming* **10** (1976) 147–175.

[38] M. Mesbahi and G.P. Papavassilopoulos, "A cone programming approach to the bilinear matrix inequality problem and its geometry," *Mathematical Programming* **77** (1997) 247–272.

[39] Y.E. Nesterov, "Semidefinite relaxation and nonconvex quadratic optimization," CORE Discussion Paper, #9744, 1997.

[40] Y.E. Nesterov and A.S. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming* (SIAM, Philadelphia, 1994).

[41] M.G. Nicholls, "The application of non-linear bilevel programming to the aluminium industry," *Journal of Global Optimization* **8** (1996) 245–261.

[42] P.M. Pardalos and G. Rodgers, "Computational aspects of a branch and bound algorithm for quadratic 0-1 programming," *Computing* **45** (1990) 131–144.

[43] P.M. Pardalos and S.A. Vavasis, "Quadratic programming with one negative eigenvalue is NP-hard," *Journal of Global Optimization* **1** (1991) 843–855.

[44] G. Pataki and L. Tunçel, "On the generic properties of convex optimization problems in conic form," *Research Report* 97–16, Dept. of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada, 1997.

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

[46] M.V. Ramana, *An algorithmic analysis of multiquadratic and semidefinite programming problems,* PhD thesis, Johns Hopkins University, Baltimore, MD, 1993.

[47] H.S. Ryoo and N.V. Sahinidis, "A branch-and-reduce approach to global optimization," *Journal of Global Optimization* **8** (1996) 107–139.

[48] N.V. Sahinidis, "BARON : Branch and reduce optimization navigator, User's manual ver. 3.0", Dept. of Chemical Engineering, University of Illinois, Urbana, Illinois, 1998.

[49] N.V Sahinidis and I.E. Grossmann, "Reformulation of multiperiod MILP models for planning and scheduling of chemical processes", *Computers & Chemical Engineering* **15** (1991) 255-272.

[50] M. Sato, H. Nakada, S. Sekiguchi, S. Matsuoka, U. Nagashima and H. Takagi, "Ninf: a network based information library for a global world-wide computing infrastructure," in P. Sloot and B. Hertzberger (eds.), *High-Performance Computing and Networking, Lecture Notes in Computing Science Vol. 1225* (Springer-Verlag, Berlin, 1997).

[51] S. Schaible, "Fractional programming" in R. Horst and P.M. Pardalos (eds.), *Handbook of Global Optimization* (Kluwer Academic Publishers, Dordrecht, 1995).

[52] S. Sekiguchi, M. Sato, H. Nakada, S. Matsuoka and U. Nagashima , "– Ninf –: network based information library for globally high performance computing," in *Proc. of Parallel Object-Oriented Methods and Applications (POOMA'96),* February 1996.

[53] H.D. Sherali and W.P. Adams, "A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems," *SIAM Journal on Discrete Mathematics* **3** (1990) 411–430.

[54] H.D. Sherali and W.P. Adams, "A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems," *Discrete Applied Mathematics* **52** (1994) 83–106.

[55] H.D. Sherali and A. Alameddine, "A new reformulation-linearization technique for bilinear programming problems," *Journal of Global Optimization* **2** (1992) 379–410.

[56] H.D. Sherali and C.H. Tuncbilek, "A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique," *Journal of Global Optimization* **2** (1992) 101–112.

[57] H.D. Sherali and C.H. Tuncbilek, "A reformulation-convexification approach for solving nonconvex quadratic programming problems," *Journal of Global Optimization* **7** (1995) 1–31.

[58] K. Shimizu and E. Aiyoshi, "A new computational method for Stackelberg and min-max problems by use of a penalty method," *IEEE Transactions on Automatic Control* **AC-26** (1981) 460–466.

[59] N.Z. Shor, "Dual quadratic estimates in polynomial and boolean programming," *Annals of Operations Research* **25** (1990) 163-168.

[60] R.A. Stubbs and S. Mehrotra, "A branch-and-cut method for 0-1 mixed convex programming," Technical Report 96-01, Dept. of IE/MS, Northwestern University, Evanston, IL 60208, revised October 1997.

[61] R.E. Swaney, "Global solution of algebraic nonlinear programs," *AIChE Annual Meeting*, Chicago, 1990.

[62] A. Takeda, Y. Dai, M. Fukuda, and M. Kojima, "Towards the Implementation of Successive Convex Relaxation Method for Nonconvex Quadratic Optimization Problems," in P.M. Pardalos (ed.), *Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems* (Kluwer Academic Publishers, Dordrecht, 2000).

[63] A. Takeda, K. Fujisawa, Y. Fukaya, and M. Kojima, "'A Parallel Successive Convex Relaxation Algorithm for Quadratic Optimization Problems," to appear in *Optimization - Modeling and Algorithms*, The Institute of Statistical Mathematics Cooperative Research Report 126, Tokyo, 2001.

[64] A. Takeda and M. Kojima, "Successive Convex Relaxation Approach to Bilevel Quadratic Optimization Problems," Technical Report B-352, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Meguro, Tokyo, Japan, August 1999, to appear in M.C. Ferris, O.L. Mangasarian and J.S. Pang (eds.), *Applications and Algorithms of Complementarity* (Kluwer Academic Publishers, Dordrecht).

[65] A. Takeda, M. Kojima and K. Fujisawa, "A Combinatorial Problem Arising from Polyhedral Homotopies for Solving Polynomial Systems," *Continuous and Discrete Mathematics for Optimization*, Research Institute for Mathematical Sciences, RIMS Kokyuroku 1115, Kyoto, 2000.

[66] A. Takeda and H. Nishino, "On Measuring the Inefficiency with the Inner-Product Norm in Data Envelopment Analysis," Technical Report B-343, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Meguro, Tokyo, Japan, revised August 1999, to appear in *European Journal of Operational Research.*

[67] H. Tuy, *Convex analysis and Global Optimization* (Kluwer Academic Publishers, Dordrecht, 1998).

[68] L.N. Vicente and P.H. Calamai, "Bilevel and multilevel programming : A bibliography review," *Journal of Global Optimization* **5** (1994) 291–306.

[69] V. Visweswaran and C.A. Floudas, "A global optimization (GOP) for certain classes of nonconvex NLPs – II. Application of theory and test problems," *Computers and Chemical Engineering* **14** (1990) 1419–1434.

[70] V. Visweswaran, C.A. Floudas, M.G. Ierapetritou and E.N. Pistikopoulos, "A decomposition-based global optimization approach for solving bilevel linear and quadratic programs," in C.A. Floudas and P.M. Pardalos (eds.), *State of the Art in Global Optimization* (Kluwer Academic Publishers, Dordrecht, 1996).

[71] T.V. Voorhis and F. Al-khayyal, "Accelerating convergence of branch-and-bound algorithms for quadratically constrained optimization problems," in C.A. Floudas and P.M. Pardalos (eds.), *State of the Art in Global Optimization* (Kluwer Academic Publishers, Dordrecht, 1996).

[72] Y. Ye, "Approximating quadratic programming with quadratic constraints," Working paper, Dept. of Management Sciences, The University of Iowa, April 1997.