2019

Solution Techniques For Non-convex Optimization Problems

Wei Xia
Lehigh University, wex213@lehigh.edu

Solution Techniques For Non-convex Optimization Problems

by

Wei Xia

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Doctor of Philosophy

in

Industrial and Systems Engineering

Lehigh University

January 2019

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

Date

Dissertation Advisor

Committee Members:

Luis F. Zuluaga, Committee Chair

Frank E. Curtis

Martin Takáč

Juan C. Vera

Acknowledgements

First of all, I would like to express my sincere gratitude to my advisor Professor Luis F. Zuluaga, who has always been patient and supportive through my PhD. He gave me invaluable advices on my research as well as on life. For me, he is not only my advisor, but also a mentor in life. Besides my advisor, I would like to thank the rest of my thesis committee: Professor Frank E. Curtis, Professor Martin Takáč, and Professor Juan C. Vera, for their insightful comments and hearty encouragement that made the thesis what it is today. I would like to thank all the Professors in the ISE department for providing excellent courses for students, and the staffs for creating a wonderful environment. I am grateful to my Professors in my undergraduate studies: Professor Gexin Yu, Professor Junping Shi, Professor Ryan Vinroot, and Professor Vladimir Bolotnikov for encouraging me to pursue a PhD degree. And I would like to thank my friends at Lehigh, who has been nothing but supportive. Yuhai Hu, Xi He, Xiaolong Kuang, Chenxin Ma, Jie Liu, Shu Tu, Choat Inthawongse, Matt Menickelly, Hiva Ghanbari and Mohammadreza Samadi have been great friends. I particularly enjoyed the discussions with Yuhai Hu, Xi He, Chenxin Ma and Xin Shi. Finally, I would like to thank my family for their unconditional support and love.

Contents

List of Tables

List of Figures

Abstract

This thesis focuses on solution techniques for non-convex optimization problems. The first part of the dissertation presents a generalization of the completely positive reformulation of quadratically constrained quadratic programs (QCQPs) to polynomial optimization problems. We show that by explicitly handling the linear constraints in the formulation of the POP, one obtains a refinement of the condition introduced in [6] on QCQPs, where the refined theorem only requires nonnegativity of polynomial constraints over the feasible set of the linear constraints. The second part of the thesis is concerned with globally solving non-convex quadratic programs (QPs) using integer programming techniques. More specifically, we reformulate non-convex QP as a mixed-integer linear problem (MILP) by incorporating the KKT condition of the QP to obtain a linear complementary problem, then use binary variables and big-M constraints to model the complementary constraints. We show how to impose bounds on the dual variables without eliminating all the (globally) optimal primal solutions; using some fundamental results on the solution of perturbed linear systems. The solution approach is implemented and labeled as quadprogIP, where computational results are presented in comparison with quadprogBB, BARON and CPLEX. The third part of the thesis involves the formulation and solution approach of a problem that arises from an on-demand aviation transportation network. A multi-commodity network flows (MCNF) model with side constraints is proposed to analyze and improve the efficiency of the on-demand aviation network, where the electric vertical-takeoff-and-landing (eVTOLs) transportation vehicles and passengers can be viewed as commodities, and routing them is equivalent to finding the optimal flow of each commodity through the network. The side constraints capture the decisions involved in the limited battery capacity for each eVTOL. We propose two heuristics that are efficient in generating integer feasible solutions that are feasible to the exponential number of battery side constraints. The last part of the thesis discusses a solution approach for copositive programs using linear semi-infinite optimization techniques. A copositive program can be reformulated as a linear semi-infinite program, which can be solved using the cutting plane approach, where each cutting plane is generated by solving a standard quadratic subproblem. Numerical results on QP-reformulated copositive programs are presented in comparison to the approximation hierarchy approach in [22] and [2].

Introduction

In this thesis, we consider different problems in the general area of non-convex optimization. In Chapter 1, we consider a very general class of nonlinear programs, namely polynomial problems with linear constraints. In Chapter 2, we consider the class of general non-convex quadratic programs. In Chapter 3, an application of mixed linear integer programing techniques is studied. In particular, we study a variation of the well-known vehicle routing problem. In Chapter 4, we consider a class of copositive programs obtained from non-convex quadratic programs.

The materials presented in Chapter 1 have been published in

Wei Xia and Luis F Zuluaga. Completely positive reformulations of polynomial optimization problems with linear constraints. *Optimization Letters*, pages 1–13, 2017,

and the materials presented in Chapter 2 is to appear in

Wei Xia, Juan Vera, and Luis F Zuluaga. Globally solving non-convex quadratic programs via linear integer programming techniques. *to appear in INFORMS Journel of Computing*, 2015.

0.1 Completely Positive Reformulations of Polynomial Optimization Problems with Linear Constraints

A polynomial optimization problem (POP) is an optimization problem in which both the objective and constraints can be written in terms of polynomials on the decision variables. A POP can be viewed as a generalization of a quadratically constrained quadratic program (QCQP) to higher order polynomials. Nonconvex QCQPs are known to be NP hard (see, e.g., [78]), and as a result, POPs are also NP hard in nature. The difficulties of solving

POPs are due to their potential nonconvexity in both the objective and the feasible set, which makes obtaining global optimal solutions of POPs challenging. One existing approach for solving POPs makes use of sum of squares approximations and positive semidefinite moment matrices ([67, 68, 69, 70]). Alternatively, one line of research in this area looks at completely positive (CP) relaxations or reformulations for quadratic POPs (i.e., QCQPs). For example, in [24], it was shown that linearly constrained quadratic programs (LCQPs) with binary variables can be reformulated as a completely positive program (CPP). [26] extends this result to cases in which the feasible region belongs to a general convex set, with some assumptions on the quadratic coefficient matrix of the objective function over the recession cone of the feasible region. Other articles in the same line of work include [4, 6, 16, 17, 39, 42]. For POPs involving polynomials of degree larger than 2, the completely positive reformulation requires the use of *completely positive tensors* that are a natural extension of the completely positive matrices (cf., [31, 39, 71, 84]). For example, consider the work of [80], [5]. Recently, it has been shown that under appropriate assumptions, POPs can be reformulated as conic problems over the cone of completely positive tensors; which generalize the set of completely positive matrices. In Chapter 2, we show that by explicitly handling the linear constraints in the formulation of the POP, one obtains a generalization of the completely positive reformulation of quadratically constrained quadratic programs recently introduced by [6].

0.2 Globally solving Non-Convex QPs via MILP techniques

Quadratic programming (QP) is a well-studied fundamental NP-hard optimization problem which optimizes a quadratic objective over a set of linear constraints. In Chapter 3, we reformulate QPs as a mixed-integer linear problem (MILP). This is done via the reformulation of QP as a linear complementary problem, and the use of binary variables and big-M constrains. To obtain such reformulation, we show how to impose bounds on the dual variables without eliminating all the (globally) optimal primal solutions; using some fundamental results on the solution of perturbed linear systems, to model the complementary constraints.

We also illustrate the performance of our solution approach by comparing our solver with the current benchmark global QP solver `quadprogBB`, as well as with `BARON`, one of the leading non-linear programming (NLP) solvers, on a large variety of QP test instances. In practice, this approach is shown to typically outperform by orders of magnitude `quadprogBB`. This approach also outperforms `BARON`, on standard quadratic programming (SQP) and on randomly generated QP test instances considered in the related literature. Also, our approach has a comparable performance to `BARON` on box constrained QPs and more general QP instances from the literature. The `MATLAB` code, called `quadprogIP`, and the instances used to perform these numerical experiments are publicly available at `https://github.com/xiawei918/quadprogIP`.

0.3 Multi-Commodity Network Flow Problems with Resource Constraints

A recent advance in the technology of electric Vertical Take-off and Landing aircrafts (eVTOLs) has made on-demand aviation transportation a practical solution to improve urban mobility. eVTOLs, which are similar to helicopters, have several advantages that make them the ideal vehicle for an on-demand aviation network. A basic on-demand aviation transportation network (or eVTOL network) consists of the hubs, the eVTOLs, and the passengers. The development of such an on-demand aviation transportation network involves strategic decisions which are vital for the success of the operation. The efficiency of the network, in other words, the passenger throughput and time savings, are some of the most important aspect to consider when operating the network, and the routing of the eVTOLs in the network is the key to an efficient operation.

In chapter 3, we consider a multi-commodity network flows model (MCNF) model to help determine optimal routes of eVTOLs on an on-demand aviation network. The eVTOLs and passengers can be viewed as commodities in the network, and routing them is equivalent to finding the optimal flow of each commodity through the network. However, the flow of passengers between hubs is constrained by the availability of the eVTOLs, and the flow of eVTOLs is constrained by the remaining battery level of the eVTOLs. The optimal flow of both passengers and eVTOLs is the flow that transport most passengers to their respective destination on time. A heuristic is proposed based on Dijkstra's algorithm which generates good quality initial solutions in short time, and the heuristic is also used to reconstruct battery feasible incumbent solutions from battery infeasible solutions that violated the battery constraints. According to the experiments, the heuristic improves the efficiency of the solver by finding better incumbent solutions, thus improving the global upper bound. It may also be used to find good quality solutions when an optimal solution is not required.

0.4 Solving Copositive Programs via Semi-infinite Programming Approach

The theory of copositive programming and completely positive programming are closely related to the field of combinatorial and quadratic optimization problems, as they provide convex reformulations for problems that arise from these fields. It has been shown in [18] that the problem of maximizing a quadratic form over the simplex can be reformulated as an equivalent copositive program. Burer showed in [29] that any quadratic program with a mix of binary and continuous variables has an equivalent copositive program reformulation. More recent advancements on the topic of copositive programs and completely positive programs can be found in [43],[30] and [17].

In this chapter, we will focus on general copositive program of the form (COP). It is well known that copositive programs are NP-hard, despite the fact that they are convex optimization problems. Many approximation hierarchy has been proposed for the cone of copositive matrices and successfully used in the literature for solving copositive programs. In this chapter, we propose a cutting-plane algorithm for solving copositive programs. More specifically, we reformulate the copositive program as an equivalent linear semi-infinite program, which is then solved using a cutting-plane algorithm. The cutting-plane algorithm involves a pair of master problem and subproblem, where the master problem, which is an LP, generates solutions feasible to the current set of cuts, and the subproblem, which is a standard quadratic program, generates the most violated cut with respect to the current solution. Our approach exploit the efficiency of the solver in [95] on SQPs to generate strong inequalities which improves the tightness of the bounds obtained from the master problem. The preliminary experiments are conducted on a set of copositive programs obtained from reformulating the QPs.

Chapter 1

Completely Positive Reformulations of Polynomial Optimization Problems with Linear Constraints

1.1 Introduction

A polynomial optimization problem (POP) is an optimization problem that has both polynomial objective and constraints. It can be viewed as a generalization of a quadratically constrained quadratic program (QCQP) to higher order polynomials. Nonconvex QCQPs are known to be NP hard (see, e.g., [78]), and as a result, POPs are also NP hard in nature. The difficulties of solving POPs are due to their potential nonconvexity in both objective and feasible set, which makes obtaining global optimal solutions of POPs challenging. One existing approach for solving POPs makes use of sum of squares approximations and positive semidefinite moment matrices. For more details we refer the readers to [67, 68, 69, 70]. Alternatively, one line of research in this area looks at completely positive (CP) relaxations or reformulations for quadratic POPs (i.e., QCQPs). For example, in [24], it was shown that linearly constrained quadratic programs (LCQPs) with binary variables can be reformulated as a completely positive program (CPP). [26] extends this result to cases in which the feasible region belongs to a general convex set, with some assumptions on the quadratic coefficient matrix of the objective function over the recession cone of the feasible region. Other articles in the same line of work include [4, 6, 16, 17, 39, 42]. For POPs involving polynomials of degree larger than 2, the completely positive reformulation requires the use of *completely positive tensors* that are a natural extension of the completely positive matrices (cf., [31, 39, 71, 84]). For example, consider the work of [80], [5]. In this paper, instead of studying the POPs and CPPs over general convex cones, we concentrate on $\mathbb{R}^n_+$, in which the desired convex reformulations can be written in terms of the well studied CP matrices and tensors. For more details on CP matrices and tensors, we refer our readers to [16, 31, 39, 42, 71].

Here, we propose a convex reformulation result of POPs which considers the set of linear constraints explicitly. This result can be viewed as a refinement of [80, Thm. 5], and is shown to be a generalization of [6, Thm. 4]. Instead of assuming the polynomial constraints to be nonnegative over $\mathbb{R}^n_+$, the refined theorem only requires nonnegativity of polynomial constraints over the feasible set of the linear constraints.

The rest of the paper is organized as follows: Section 2 introduces the notation. Section 3 gives a brief summary on CP relaxations of QCQPs. Section 4 presents the main result and its relationship with related results for QCQPs and POPs in general. Section 5 concludes with some final remarks.

1.2 Preliminaries

We first introduce the notation that is used throughout the article. Let $\mathbb{R}$ denote the set of real numbers, and $\mathbb{R}_+$ denote the set of nonnegative real numbers. Let $\mathcal{S}^n$ denote the set of symmetric matrices in $\mathbb{R}^{n\times n}$, and $\mathcal{S}^n_d$ denotes the set of symmetric tensors of dimension n and order d. Let $\mathbb{R}[x]$ denote the set of n-variate polynomials with real coefficients. Similarly, $\mathbb{R}_d[x]$ denotes the polynomials in $\mathbb{R}[x]$ of degree at most d. For any $A \in \mathcal{S}^n$, $\operatorname{diag}(A)$ denotes the vector of elements on the diagonal of the matrix A, and for any $a, b \in \mathbb{R}^n$, let $a \circ b$ denote the Hadamard product of vectors a and b . For any $A, B \in \mathcal{S}^n$, let $\langle A, B\rangle$ denote the trace of the product of matrices A and B, where $\operatorname{trace}(A) = \sum_{i=1}^n A_{ii}$. A polynomial of degree d is represented as

$$h(x) = \sum_{\alpha\in\mathbb{Z}^n_+ : \|\alpha\|_1 \le d} h_\alpha x^\alpha,$$

where $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}$. For any polynomial $h(x)$, $\tilde{h}(x)$ denotes the homogeneous term of highest degree. Let $\deg(h)$ denote the degree of a polynomial $h(x)$. Similar to [80], we let $M_d : \mathbb{R}^n \to \mathcal{S}_d^n$ be the map defined by

$$M_d(x) = \underbrace{x \otimes x \otimes \ldots \otimes x}_{d},$$

where $\otimes$ denotes the tensor product. In the case of $d = 2$, the tensor $M_2(x)$ is the rank-one matrix $M_2(x) = xx^T$. Also, the cone of completely positive tensors is defined as

$$\mathcal{C}_{n,d} = \operatorname{conv}(M_d(\mathbb{R}_+^n)),$$

where $\operatorname{conv}(\cdot)$ denotes the convex hull. Notice that $\mathcal{C}_{n,2}$ is the cone of completely positive matrices (cf., [42])

$$\mathcal{C}_n = \Big\{ \sum_{i=1}^{k} x_i x_i^T : x_i \in \mathbb{R}_+^n \text{ for } i \in \{1, \ldots, k\}, k \in \mathbb{N} \Big\}.$$

For a given set $U \subseteq \mathbb{R}^n$, let $\operatorname{conic}(U) = \{\sum_{i=1}^k \lambda_k u_k : \lambda_k \in \mathbb{R}_+, u_k \in U, k > 0\}$ be the conic hull of U. Also, the tensor map for polynomials $C_d : \mathbb{R}_d[x] \to \mathcal{S}_d^{n+1}$ is defined by

$$C_d\Big(\sum_{\alpha \in \mathbb{Z}_+^n : \|\alpha\|_1 \le d} p_\alpha x^\alpha \Big)_{i_1,\ldots,i_d} = \frac{\alpha_1! \ldots \alpha_n!}{\|\alpha\|_1!} p_\alpha$$

where α is the exponent such that $x_1^{\alpha_1} \cdots x_n^{\alpha_n} = x_{i_1} \cdots x_{i_d}$. The tensor mapping C_d allows one to express the value of a polynomial $p \in \mathbb{R}[x]$ at $a \in \mathbb{R}^n$ as the following inner product

$$p(a) = \langle C_d(p), M_d(1, a) \rangle_{n,d}.$$

Example 1.2.1. *Let* $p(x_1, x_2) = 2x_2^3 - x_1^2 x_2 + 8x_1 x_2^2 + x_1 x_2 - x_1 - x_2^2 + 3$. *For any* $a =$

$(a_1, a_2)^T \in \mathbb{R}^2$, *$p(a) = \langle C_3(p), M_3(1, a)\rangle$, where*

$$C_{(0,\cdot,\cdot)} = \begin{pmatrix} 3 & -\frac{1}{3} & 0 \\ -\frac{1}{3} & 0 & \frac{1}{6} \\ 0 & \frac{1}{6} & -\frac{1}{3} \end{pmatrix}, \quad M_{(0,\cdot,\cdot)} = \begin{pmatrix} 1 & a_1 & a_2 \\ a_1 & a_1^2 & a_1a_2 \\ a_2 & a_1a_2 & a_2^2 \end{pmatrix},$$
$$C_{(1,\cdot,\cdot)} = \begin{pmatrix} -\frac{1}{3} & 0 & \frac{1}{6} \\ 0 & 0 & -\frac{1}{3} \\ \frac{1}{6} & -\frac{1}{3} & \frac{8}{3} \end{pmatrix}, \quad M_{(1,\cdot,\cdot)} = \begin{pmatrix} a_1 & a_1^2 & a_1a_2 \\ a_1^2 & a_1^3 & a_1^2a_2 \\ a_1a_2 & a_1^2a_2 & a_1a_2^2 \end{pmatrix},$$
$$C_{(2,\cdot,\cdot)} = \begin{pmatrix} 0 & \frac{1}{6} & -\frac{1}{3} \\ \frac{1}{6} & -\frac{1}{3} & \frac{8}{3} \\ -\frac{1}{3} & \frac{8}{3} & 2 \end{pmatrix}, \quad M_{(2,\cdot,\cdot)} = \begin{pmatrix} a_2 & a_1a_2 & a_2^2 \\ a_1a_2 & a_1^2a_2 & a_1a_2^2 \\ a_2^2 & a_1a_2^2 & a_2^3 \end{pmatrix}.$$

Following [88], let the horizon cone of $S \subseteq \mathbb{R}^n$ be defined as

$$S^\infty = \{y \in \mathbb{R}^n : \text{ there exists } x^k \in S, \lambda^k \in \mathbb{R}_+, k = 1, 2, \ldots \text{ such that } \lambda^k \downarrow 0 \text{ and } \lambda^k x^k \to y\}.$$

In the case when S is empty, the horizon cone of S is the empty set. In other words, $\{\emptyset\}^\infty = \emptyset$. More properties of the horizon cone can be found in [80, Prop. 3].

Example 1.2.2. *Let*

$$Q = \{(x_1, x_2) \in \mathbb{R}^2 : x_2 = x_1^2\}.$$

To obtain the horizon cone of Q, observe that the elements of Q are of the form $\{(a, a^2) : a \in \mathbb{R}\}$. For $k \in \mathbb{Z}_+$, $b \in \mathbb{R}_+$, let $x^k = (k, k^2)$ and $\lambda^k = \frac{b}{k^2}$, then $\lambda^k x^k \downarrow (0, b)$. Thus $\{0\} \times \mathbb{R}_+ \subseteq Q^\infty$. Now let $(a, b) \in Q^\infty$, then $(a, b) = \lim_{k\to\infty} \lambda^k(x_1^k, x_2^k)$ with $\lambda^k \downarrow 0$, and $(x_1^k, x_2^k) \in Q$. Notice that $b = \lim_{k\to\infty} \lambda^k x_2^k = \lim_{k\to\infty} \lambda^k (x_1^k)^2 \geq 0$, and $a^2 = \lim_{k\to\infty}(\lambda^k x_1^k)^2 = \lim_{k\to\infty} \lambda^k(\lambda^k x_2^k) = \lim_{k\to\infty} \lambda^k b = 0$. Thus $\{0\} \times \mathbb{R}_+ \supseteq Q^\infty$ which shows that $Q^\infty = \{0\} \times \mathbb{R}_+$.

We denote the set of feasible solutions of a set of linear constraints $Ax = b$ with $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ as $L = \{x \in \mathbb{R}^n_+ : Ax = b\}$. Note that in the case of linear constraints, the horizon cone of the feasible set is equivalent to the recession cone of the feasible set (cf., [80, Prop. 3 (i)]). That is, $L^\infty = \text{recc}(L) = \{d \in \mathbb{R}^n_+ : Ad = 0\}$.

1.3 Completely positive relaxations on QCQPs

It is known that under appropriate conditions, a QCQP can be reformulated as a completely positive program (CPP), that is, a conic (linear) program over the cone of completely positive matrices (see, e.g., [6, 16, 27]). Next, we summarize the theorems that provide insights on the conditions that guarantee the equivalence of the QCQPs with their respective CPP relaxations.

Theorem 1.3.1. ***([24, Sec. 3.2])*** *Consider the following QCQP problem:*

$$\begin{aligned} \min_{x \in \mathbb{R}^n_+} \ & q(x) = \frac{1}{2}x^T H x + f^T x \\ s.t. \ & Ax = b \\ & h_1(x) = \frac{1}{2}x^T H_1 x + f_1^T x + c_1 = 0 \end{aligned} \tag{$QCQP_1$}$$

and its CPP relaxation

$$\begin{aligned} \min_{x,X} \ & \frac{1}{2}\langle H, X\rangle + f^T x \\ s.t. \ & Ax = b \\ & \text{diag}(AXA^T) = b \circ b \\ & \frac{1}{2}\langle H_1, X\rangle + f_1^T x + c_1 = 0 \\ & \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \in \mathcal{C}_{1+n} \end{aligned} \tag{CPP_1}$$

where $H, H_1 \in \mathcal{S}^n$, $B \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$,$f, f_1 \in \mathbb{R}^n$, and $c_1 \in \mathbb{R}$. If the following conditions are satisfied, then $(QCQP_1)$ is equivalent to (CPP_1).

1. *$x \in L \Rightarrow h_1(x) \geq 0$,*

2. *$d \in L_\infty \Rightarrow d_j = 0 \quad \forall j \in \bar{B}$,*

where $\bar{B} = \{j \in \{1, \dots, n\} : (H)_j \neq 0$ *or* $(f_1)_j \neq 0\}$*, and* $(H)_j$ *denotes the* j*th column of* H*.*

In [27], it is shown that Theorem 1 applies to QCQP's with no restrictions on the quadratic constraints, but under the assumption that the set $L = \{x \in \mathbb{R}^n_+ : Ax = b\}$ is bounded.

Note that [6] showed in Lemma 1 that if $f_i(x) \geq 0, \forall x \in \ S, i = 1, \dots, m$, we have

$$\{x \in S : f_i(x) \leq 0, i = 1, \dots, m\} = \{x \in S : f(x) \leq 0\},$$

where $f(x) = \sum_{i=1}^{m} f_i(x)$. Thus in case of multiple quadratic constraints that satisfy the nonegativity condition, the quadratic constraints can be aggregated into a single quadratic constraint to obtain an exact reformulation of the QCQP using Theorem 1.

In [26], Theorem 1 was extended to the case $x \in \mathcal{K}$ where $\mathcal{K} \subseteq \mathbb{R}^n$ is a closed, convex cone, defined by using a generalization of CP matrices. More specifically, using the matrices $CP(\mathcal{K}) = \{\sum_k x^k (x^k)^T : x^k \in \mathbb{R}_+ \times \mathcal{K}\}$. The equivalence holds under the assumption that $d^T H_1 d = 0$ for all d in the recession cone of $\mathcal{K} \cap L$ and the objective of the QCQP is bounded below on $\mathcal{K} \cap L$. Alternatively, [27] extended the result in [24] to QCQPs without requirements on the quadratic constraints, but with the assumption that $\mathcal{K} \cap L$ is bounded. In [6], the result of [27] is extended to any convex cone, despite its boundedness. One of the main results in [6] is the theorem below. Although this theorem is stated over a general closed convex cone $\mathcal{K}$, our interest is focused on $\mathbb{R}^n_+$, thus we state the theorem over $\mathbb{R}^n_+$.

Theorem 1.3.2. ***([6, Thm. 4])*** *Assume* $d^T H d \geq 0$ *for any* $d \in \mathbb{R}^n_+$ *such that* $Ad = 0$*,* $a^T H_1 d = 0$*, and* $\frac{1}{2} x^T H_1 x + f_1^T x + c_1 \geq 0$ *for all* $x \in L$*. The QCQP*

$$\begin{aligned} \min_{x \in \mathbb{R}^n_+} \ & \frac{1}{2} x^T H x + f^T x \\ s.t. \ & Ax = b \\ & h_1(x) = \frac{1}{2} x^T A x + f_1^T x + c_1 = 0 \end{aligned} \tag{$QCQP_2$}$$

and the CPP relaxation

$$\begin{aligned} \min_{x,X}\ & \frac{1}{2}\langle H_1, X\rangle + d^T x \\ s.t.\ & Ax = b \\ & \operatorname{diag}(AXA^T) = b \circ b \\ & \frac{1}{2}\langle H_1, X\rangle + f_1^T x + c_1 = 0 \\ & \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \in \mathcal{C}^*_{1+n}(\mathbb{R}^n_+) \end{aligned} \tag{CPP_2}$$

are equivalent.

Note that in [6], the quadratic constraints are inequality constraints with less than or equal signs. We took the liberty to replace the inequalities with equalities since all quadratic constraints are assumed to be nonnegative on L. Thus setting the constraints to be equal or less than or equal to is equivalent.

1.4 Main Result

In this section, we extend the result of Bai from QCQPs to POPs. To achieve this, we refine [80, Thm. 5] to POPs with explicit linear constraints. That is, consider the POP:

$$\begin{aligned} \inf\quad & q(x) \\ \text{s.t.}\quad & l_i(x) = 0, \quad i = 1, \dots, m_l \\ & h_j(x) = 0, \quad j = 1, \dots, m_n \\ & x \geq 0 \end{aligned} \tag{POP}$$

and its CPP tensor relaxation:

$$\begin{aligned} \inf\quad & \langle C_d(q), Y\rangle \\ \text{s.t.}\quad & \langle C_d(h_j), Y\rangle = 0, \quad j = 1, \dots, m_n \\ & \langle C_d(l_i^d), Y\rangle = 0, \quad i = 1, \dots, m_l \\ & \langle C_d(1), Y\rangle = 1 \\ & Y \in \mathcal{C}_{n+1,d}. \end{aligned} \tag{CPP}$$

where $l_i(x)$ is a linear polynomial of the form $a_i^T x - b_i = 0$ for $i = 1, \dots, m_l$ with $a_i \in \mathbb{R}^n$

and $b_i \in \mathbb{R}$. Without loss of generality, we assume the degrees of q and h_j for $j = 1, \dots, m_n$ are equal to $d \in N$ and $d = 2k, k \in \mathbb{N}$.

For polynomials that do not satisfy the assumption, one can convert it to a degree d polynomial where

$$d = 2\lceil \frac{\max\{\deg(q), \deg(h_i), i = 1, \dots, m_n\}}{2} \rceil,$$

by multiplying each polynomial by a nonzero polynomial to raise its power to the appropriate degree.

Next, we formally define when a POP and its completely positive reformulation are said to be equivalent:

Definition 1.4.1. *Problems* (POP) *and* (CPP) *are equivalent if the following holds:*

a. The optimal values of (POP) *and* (CPP) *are equivalent.*

b. The optimal value of (POP) *is attained if and only if the optimal value of* (CPP) *is attained.*

c. For any $Y \in \mathcal{C}_{n+1,d}$*, let* $\chi(Y) = (Y_{0,\dots,0}, Y_{0,\dots,1}, \dots, Y_{0,\dots,n})$*, then if* Y *is optimal for* (CPP)*, then* $\chi(Y)$ *is in the convex hull of the set of optimal solutions of* (POP)*.*

Next we provide a new theorem which extends Theorem 2 to POPs.

Theorem 1.4.2. *Consider problem* (POP) *and* (CPP)*. Let* $l_i(x) = a_i^T x - b_i$ *where* $i = 1, \dots, m_l$ *are linear polynomials on* $x \in \mathbb{R}^n$*. Let* $h_t(x)$ *where* $t = 1, \dots, m_n$ *denote polynomials on* $x \in \mathbb{R}^n$ *of degree* $d > 1$*, and* $L = \{x \in \mathbb{R}^n_+ : a_i^T x - b_i = 0, i = 1, \dots, m_l\}$*. If the conditions:*

(i) $h_t(x) \geq 0$ *for all* $x \in L$*, where* $t = 1, \dots, m_n$*,*

(ii) $\tilde{q}(x) \geq 0$ *for all* $x \in \{x \in \mathbb{R}^n_+ : \tilde{l}_i(x) = 0, \tilde{h}_t(x) = 0,$ *where* $i = 1, \dots, m_l, t \in 1, \dots, m_n\}$

are satisfied, then (POP) *and* (CPP) *are equivalent.*

Before we prove the theorem, we restate two Lemmas for reference.

Lemma 1.4.3. ***([80, Lem. 1])*** *If* $q \in \mathbb{R}[x]$ *is bounded below in* $S \subseteq \mathbb{R}^n$*, then* $\tilde{q}(x) \geq 0$ *for all* $x \in S^\infty$*.*

Lemma 1.4.4. ***([80, Lem. 2])*** *For any* $d > 0$ *and* $n > 0$*,* $\mathcal{C}_{n+1,d} = conic(M_d(\{0,1\} \times \mathbb{R}^n_+))$*.*

of Theorem 3. This proof borrows from the proof of [80, Thm. 5]. Define the following sets

$$\mathcal{F}_{POP} = \{x \in \mathbb{R}^n_+ : x \text{ is a feasible solution to (POP)}\}$$
$$\mathcal{O}_{POP} = \{x \in \mathbb{R}^n_+ : x \text{ is an optimal solution to (POP)}\}$$
$$\mathcal{F}_{CPP} = \{Y \in \mathcal{S}^{n+1}_d : Y \text{ is a feasible solution to (CPP)}\}$$
$$\mathcal{O}_{CPP} = \{Y \in \mathcal{S}^{n+1}_d : Y \text{ is an optimal solution to (CPP)}\}$$

Let $\nu^* = \inf\{q(x) : x \in \mathcal{F}_{POP}\}$. Since CPP is a relaxation of POP, clearly we have

$$\nu^* \geq \inf\{\langle C_d(q), Y\rangle : Y \in \mathcal{F}_{CPP}\}.$$

In particular, the statement of the theorem holds when $\nu^* = -\infty$. Now, assume $q(x)$ is bounded below in $\mathcal{F}_{POP}$. By Lemma 1 we obtain

$$\tilde{q}(x) \geq 0 \text{ for any } x \in \mathcal{F}^{\infty}_{POP}.$$

By Lemma 2, for any $Y \in \mathcal{C}_{n+1,d}$

$$Y = \sum_{k=1}^{n_1} \lambda_k M_d(1, u_k) + \sum_{j=1}^{n_0} \mu_j M_d(0, v_j) \tag{1.1}$$

for some $n_0, n_1 \in \mathbb{N}$, $\lambda_k, \mu_j > 0$, and $u_k, v_j \in \mathbb{R}^n_+$.
If $Y \in \mathcal{F}_{CPP}$,

$$1 = \langle C_d(1), Y\rangle = \sum_{i=1}^{n_1} \lambda_k \tag{1.2}$$

also, for any $i = 1, \ldots, m_l$

$$0 = \langle C_d(l_i^d), Y\rangle = \sum_{k=1}^{n_1} \lambda_k l_i^d(u_k) + \sum_{j=1}^{n_0} \mu_j \tilde{l}_i^d(v_j) \tag{1.3}$$

and for any $t = 1, \ldots, m_n$

$$0 = \langle C_d(h_t), Y\rangle = \sum_{k=1}^{n_1} \lambda_k h_t(u_k) + \sum_{j=1}^{n_0} \mu_j \tilde{h}_t(v_j). \tag{1.4}$$

Since $l_i^d(x) \geq 0$ for any $x \in \mathbb{R}_+^n$, $i = 1, \dots, m_l$, then by Lemma 1 we have $\tilde{l}_i^d(x) \geq 0$ for any $x \in \mathbb{R}_+^n$, $i = 1, \dots, m_l$. This together with (1.3) implies that

$$u_k \in \bigcap_{i=1}^{m_l} \{x \in \mathbb{R}_+^n : l_i^d(x) = 0\} = L \tag{1.5}$$

and

$$v_j \in \bigcap_{i=1}^{m_l} \{x \in \mathbb{R}_+^n : \tilde{l}_i^d(x) = 0\} = L^\infty.$$

Since $h_t(x) \geq 0$ for all $x \in L, t = 1, \dots, m_n$. Then, by Lemma 1 we have $\tilde{h}_t(x) \geq 0$ for any $x \in L^\infty, t = 1, \dots, m_n$. Thus from (1.4) we get

$$h_t(u_k) = 0 \text{ for any } k = 1, \dots, n_0, t = 1, \dots, m_n \tag{1.6}$$

and

$$\tilde{h}_t(v_j) = 0 \text{ for any } j = 1, \dots, n_1, t = 1, \dots, m_n. \tag{1.7}$$

Thus for any $j = 1, \dots, n_0$ we have

$$v_j \in \{x \in \mathbb{R}_+^n : \tilde{l}_i^d(x) = 0 \text{ for } i = 1, \dots, m_l, \tilde{h}_t(x) = 0, t = 1, \dots, m_n\}.$$

By condition (2) of Theorem 3

$$\tilde{q}(v_j) \geq 0 \text{ for } j = 1, \dots, n_0.$$

Therefore

$$\langle C_d(q), Y\rangle = \sum_{k=1}^{n_1} \lambda_k q(u_k) + \sum_{j=1}^{n_0} \mu_j \tilde{q}(v_j) \geq \sum_{k=1}^{n_1} \lambda_k q(u_k) \geq \nu^*, \tag{1.8}$$

where the last inequality follows from (1.2), (1.5), and (1.6).

Thus part (a) of Definition 1 holds. To prove part (b), notice that for $x^* \in \mathcal{O}_{POP}$, we have $M_d(x^*) \in \mathcal{F}_{CPP}$ and $\langle C_d(q), M_d(x^*)\rangle = q(x^*) = v^*$. If $Y^* \in \mathcal{O}_{CPP}$, by (1.8) we know that each u_k in the decomposition (1.1) is in $\mathcal{O}_{POP}$. To prove part (c), we have

$$\chi(Y^*) = \sum_{k=1}^{n_1} \lambda_k \chi(M_d(1, u_k)) + \sum_{j=1}^{n_0} \mu_j \chi(M_d(0, v_j)) = \sum_{i=1}^{n_1} \lambda_k v_k \in \mathcal{O}_{POP}$$

since $\sum_{i=1}^{n_1} \lambda_k = 1$ from (1.2).

□

It can be seen that the two conditions of Theorem 3 are generalizations of the assumptions of Theorem 2. The first assumption of Theorem 2, where the quadratic constraints have to satisfy $h_1(x) = \frac{1}{2}x^T H_1 x + f_1^T x + c_1 \geq 0$ for all $x \in L$ is a special case of the first condition of Theorem 3, which requires all polynomial constraints to be nonnegative over the feasible region defined by the linear constraints. The other assumption of Theorem 2, $d^T H d \geq 0$ for $d \in \mathbb{R}^n_+$ such that $Ad = 0$ and $d^T H_1 d = 0$, is also a special case of the second condition of Theorem 3; that is, $\tilde{q}(d) \geq 0$ for all $d \in \{d \in \mathbb{R}^n_+ : \tilde{l}_i^d(d) = 0 \text{ for } i = 1, \ldots, m_l, \tilde{h}_t(d) = 0, t = 1, \ldots, m_n\}$, since $\{d \in \mathbb{R}^n_+ : \tilde{l}_i^d(d) = 0 \text{ for } i = 1, \ldots, m_l, \tilde{h}_1(d) = 0\} = \{d \in \mathbb{R}^n_+ : (a_i^T d)^d = 0, i = 1, \ldots, m_l, d^T H_1 d = 0\} = \{d \in \mathbb{R}^n_+ : Ad = 0, d^T H_1 d = 0\}$. Therefore, Theorem 3 is a generalization for POPs of higher than quadratic degree. Note that under Theorem 2, any QCQP can be converted to $(QCQP_2)$, which only contains linear constraints and one nonnegative quadratic constraint (cf. [6]).

In Theorem 3, the polynomial constraints and the linear constraints that are raised to an even power degree can also be aggregated into a single polynomial constraint. This is equivalent to writing each constraint individually as done above. One difference from Theorem 2 is how the linear constraints are reformulated. Raising the linear constraints to the dth power as it is done in Theorem 3 is not the only way to handle the linear constraints. One may carry the linear constraints to the relaxation by adding the redundant constraints $Ax = b$. Note that in the case of $d = 2$, the reformulation of the linear constraints $Ax = b$ by Theorem 2

$$\langle C_d((a_i^T x - b_i)^{2k}), Y \rangle = 0, i = 1, \ldots, m_l, \tag{1.9}$$

and the reformulation of the linear constraints in Theorem 3

$$Ay = b, \operatorname{diag}(AYA^T) = b \circ b \tag{1.10}$$

are equivalent. To see this, notice that (1.9) is equivalent to

$$\langle C_d((a_i^T x)^{2k} - 2(a_i^T x)^k b_i^k + (b_i)^{2k}), Y\rangle = 0, i = 1, \dots, m_l,$$

which is equivalent to

$$\langle C_1(a_i^T x - b_i), Y\rangle = 0 \tag{1.11}$$

$$\langle C_d((a_i^T x)^{2k} - b_i^{2k})), Y\rangle = 0 \tag{1.12}$$

where a redundant linear constraint (1.11) is added so the equivalence holds. For $k = 1$ and letting $Y = \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix}$, (1.11) and (1.12) are equivalent to (1.10). Both reformulations are equivalent to the corresponding POP, but for consistency and simplicity, we adopt reformulation (1.9).

Another way to handle the linear constraints is to multiply the linear constraint by any strictly positive polynomials of degree $d-1$, therefore obtaining a polynomial of degree d that is equivalent to the linear constraint. The effect of these different way of reformulating the linear constraints may vary. On the other hand, simply enforcing the linear constraint by multiplying a positive polynomial of degree $d-1$ might maintain sparsity. Notice that if $d > 2$, one might choose different ways to represent the linear constraints. However, to better compare Theorem 3 with both Theorem 2 and [80, Thm. 5], it is simpler to raise the polynomials defining the linear constraints to the dth power.

Consider POPs of the form:

$$\begin{aligned} \inf \quad & q(x) \\ \text{s.t.} \quad & h_j(x) = 0, \quad j = 1, \dots, m \\ & x \geq 0 \end{aligned} \tag{POPNL}$$

and its CPP tensor relaxation:

$$\begin{array}{rl} \inf & \langle C_d(q), Y\rangle \\ \text{s.t.} & \langle C_d(h_j), Y\rangle = 0, \quad j = 1,\dots,m \\ & \langle C_d(1), Y\rangle = 1 \\ & Y \in \mathcal{C}_{n+1,d}. \end{array} \tag{CPPNL}$$

Above, we use “NL" for nonlinear to emphasize the fact that the potential linearity of any of the $h_j(x), j = 1,\dots,m$ is not explicitly used in the related results stated below.

Theorem 1.4.5. ***([80, Thm. 4])*** *Let $q, h_1, \cdots, h_m \in \mathbb{R}_d[x]$ in* (POPNL) *be such that for $i = 1, \cdots, m$*

(i) deg$(h_i) = d, h_i(x) \geq 0$ for all $x \in S_{i-1}$, and

(ii) $\{x \in S_{i-1}^{\infty} : \tilde{h}_i(x) = 0\} \subseteq S_i^{\infty}$

where $S_0 = \mathbb{R}^n_+$ and $S_i = \{x \in S_{i-1} : h_i(x) = 0\}, i = 1, \cdots, m$. Then (POPNL) *and* (CPPNL) *are equivalent.*

Theorem 1.4.6. ***([80, Thm. 5])*** *Let $q, h_1, \cdots, h_m \in \mathbb{R}_d[x]$ in* (POP) *be such that for $i = 1, \cdots, m$*

(i) deg$(h_i) = d, h_i(x) \geq 0$ for all $x \in \mathbb{R}^n_+$, and

(ii) $\tilde{q}(x) \geq 0$ for all $x \in \{x \in \mathbb{R}^n_+ : \tilde{h}_i(x) = 0, i = 1, \cdots, m\}$

Then (POPNL) *and* (CPPNL) *are equivalent.*

Clearly, Theorem 3 is a refinement of [80, Thm. 5] since it does not require the non-linear polynomials to be nonnegative over $\mathbb{R}^n_+$, but only over L.

Next we show that neither [80, Thm. 4] or Theorem 3 here is more general than the other. For that purpose, an example which satisfies the conditions of [80, Thm. 4(i)] but not the conditions of Theorem 3(i) can be easily constructed. In particular, consider the following problem

$$
\begin{aligned}
\min \quad & x_1^4 \\
& l_1(x_1, x_2) = x_1 - x_2 = 0 \\
& h_2(x_1, x_2) = (x_2 + 1)^2 (x_1 - 2)^2 = 0 \\
& h_3(x_1, x_2) = (x_1^2 + 1)(x_2^2 - 4) = 0 \\
& x_1, x_2 \in \mathbb{R}_+^2
\end{aligned}
$$

We can see that for $(x_1, x_2) = (1, 1)$, the constraint $l_1(1, 1) = 0$ is satisfied, but after reformulating $l_1(x_1, x_2)$ as $h_1(x_1, x_2) = (x_1 - x_2)^4$, we have that $h_3(1, 1) = -6 < 0$, so Theorem 3(i) is not satisfied. On the other hand, $h_2(x_1, x_2) \geq 0$ for any $(x_1, x_2) \in \mathbb{R}^2$, and $h_3(x_1, x_2) = 0 \geq 0$ for $\{(x_1, x_2) \in \mathbb{R}_+^2 : h_1(x_1, x_2) = 0, h_2(x_1, x_2) = 0\} = \{(2, 2)\}$. Thus Theorem 4(i) is satisfied. Also,

$$
\begin{aligned}
&\{(x_1, x_2) \in S_0^\infty : \tilde{h}_1(x_1, x_2) = 0\} = \{x \in \mathbb{R}_+^2 : x_1 = x_2\} = S_1^\infty \\
&\{(x_1, x_2) \in S_1^\infty : \tilde{h}_2(x_1, x_2) = 0\} = \{(0, 0)\} = S_2^\infty \\
&\{(x_1, x_2) \in S_2^\infty : \tilde{h}_3(x_1, x_2) = 0\} = \{(0, 0)\} = S_3^\infty
\end{aligned}
$$

Therefore Theorem 4(ii) is satisfied.

Similarly, one can easily construct an example that satisfies the conditions of Theorem 3 but not those of Theorem 4. In particular, consider the following problem

$$
\begin{aligned}
\min \quad & x_1^4 \\
& l_1(x_1, x_2) = x_1 - 1 = 0 \\
& h_2(x_1, x_2) = (x_1^2 - x_2)^2 = 0 \\
& (x_1, x_2) \in \mathbb{R}_+^2
\end{aligned}
\tag{1.13}
$$

Clearly, after reformulating $l_1(x_1, x_2)$ as $h_1(x_1, x_2) = (x_1 - 1)^4$, problem (1.13) satisfies $\deg(q) = \deg(h_i)$ for $i = 1, 2$. Also, $h_2(x_1, x_2) \geq 0$ for all $(x_1, x_2) \in \mathbb{R}^2$ since it is a sum of squares. Next, we have

$$\{(x_1,x_2) \in \mathbb{R}^2_+ : \tilde{l}_1(x_1,x_2) = 0\} = \{(0,x_2) : x_2 \geq 0\}$$
$$\{(x_1,x_2) \in \mathbb{R}^2_+ : \tilde{l}_1(x_1,x_2) = 0, \tilde{h}_2(x_1,x_2) = 0\} = \{(0,x_2) : x_2 \geq 0\}$$

so that $\tilde{q}(x_1,x_2) = x_1^4 = 0 \geq 0$ for $(x_1,x_2) \in \{(x_1,x_2) \in \mathbb{R}^2_+ : \tilde{l}_1(x_1,x_2) = 0, \tilde{h}_2(x_1,x_2) = 0\}$, and the conditions of Theorem 3 are satisfied.

Also, if one checks Theorem 4(ii), we have

$$\begin{aligned} S_1 &= \{(x_1,x_2) \in \mathbb{R}^2_+ : l_1(x_1,x_2) = 0\} = \{(1,x_2) : x_2 \geq 0\} \\ S_2 &= \{(x_1,x_2) \in \mathbb{R}^2_+ : l_1(x_1,x_2) = h_2(x_1,x_2) = 0\} = \{(1,1)\} \\ S_1^\infty &= \{(0,x_2) : x_2 \geq 0\} \end{aligned}$$

and

$$\begin{aligned} \{(x_1,x_2) \in S_1^\infty : \tilde{h}_2(x_1,x_2) = 0\} = \{(0,x_2) : x_2 \geq 0\} &\not\subseteq S_2^\infty \\ &= \{(1,1)\}^\infty = \{(0,0)\} \end{aligned}$$

Thus the conditions of Theorem 4 are not satisfied. We conclude that Theorem 3 and Theorem 4 are not equivalent.

1.5 Conclusion

In this paper, we extended [6, Theorem 4] on QCQPs to POPs with linear constraints. This new theorem is a refinement of [80, Thm. 5]. The main difference introduced by the new theorem is a weaker requirement on the nonnegativity of the nonlinear constraints than [80, Thm. 5]. The relations among an alternative reformulation of POPs to CPs of [80, Thm. 4] and Theorem 3 are studied here by providing appropriate examples.

An interesting direction of future work is to find other CP reformulation results for QCQPs that could be potentially generalized to apply for POPs. Also interesting is to investigate whether a set of weaker conditions can be imposed on the POP while still being able to obtain a CP reformulation of the POP.

Chapter 2

Globally solving Non-Convex Quadratic Programs via Linear Integer Programming techniques

2.1 Introduction

Quadratic programmming (QP), is a fundamental optimization problem with a quadratic objective and linear constraints. QP is NP-hard [see, e.g., 78, and the references therein], however, when the objective is convex, QP can be globally solved (within a predetermined precision $\epsilon > 0$) in polynomial time via *interior-point methods* [see, e.g., 86]. Here, the focus is on obtaining global solutions for non-convex QP. QP is arguably the most basic instance of a (non-convex), *non-linear program* (NLP). At a fundamental level, the complexity of globally solving QP lies in the fact that multiple of its local optimal solutions may not necessarily be global optimal solutions [see, e.g. 12].

QPs commonly arise in applications in engineering, pure and social sciences, finance, and economics [see, e.g., 62]. As a result, there has been extensive work on studying how to obtain global solutions of QPs using both NLP techniques [see, 47, 54, for surveys in this area], and convex optimization techniques [consider, e.g., 32, 64, 65, 76, among many others].

In this paper, we reformulate QPs as a mixed-integer linear problem (MILP). This provides an advantageous way to obtain global solutions for QPs, as it allows the use of current state-of-the-art MILP solvers. Moreover, the numerical experiments of Section 2.3 show that a basic implementation of this MILP based solution approach, which we refer to as `quadprogIP`, typically outperforms by orders of magnitude `quadprogBB`, `BARON`, and `CPLEX` on standard QPs. In most of the general QP instances, `quadprogIP` outperforms `quadprogBB` and `BARON`, but it is outperformed by `CPLEX`. For box-constrained QPs, `quadprogIP` has a comparable performance to `quadprogBB` and `BARON` in small- to medium-scale instances, but it is outperformed by these solvers on large-scale instances, and by `CPLEX` in all box-constrained QPs.

Unlike `quadprogBB`, the solution approach proposed here is able to solve QP instances whose dual feasible set is unbounded. The `MATLAB` code and the instances used to perform these numerical experiments are available at `https://github.com/xiawei918/quadprogIP`.

To obtain the proposed MILP-reformulation (see Sec. 2.2.1), the QP's KKT conditions are used to reformulate the QP as a *linear complementarity problem* (LCP). In this reformulation, the complexity of the problem is captured by the complementarity constraints. The *KKT-branching approach* [28], which consist on branching on this complementarity constraints, is not useful on this reformulation of the problem, as the underlying linear relaxations at the root node of the KKT-branching tree are (under mild assumptions) unbounded [28, Cor. 2.3]. Another alternative, namely, reformulating the complementarity constraints using binary variables and big-M constraints, requires the knowledge of bounds on the problem's KKT multipliers, which in general are unbounded [cf., 63, Sec. 6.1 and 6.2]. To directly use MILP solvers for the solution of the QP, we overcome this requirement by restricting our attention to a subset of optimal KKT points. We show (Theorem 1) that it is possible to impose bounds on the dual variables without eliminating any of the (globally) optimal primal solutions. Our results are based on fundamental results on the approximate solution of systems of linear equations [e.g., 55, 72]. One advantage of the proposed methodology is that unlike previous related work, the convergence of the MILP-based approach to the QP's global optimal solution in finite time follows in straightforward fashion (see Sec. 2.3.1). Also, the methodology can be applied to QPs without the need for assumptions on the relative interior of its feasible set (see Sec. 2.2.3 for details).

Before stating the results described above, we end this section with a short review of both NLP and convex optimization techniques for the global solution of QP's. Using NLP techniques, [94] proposed an interior-point algorithm for NLPs (thus, applicable for QPs), which is an extension of the interior-point methods for linear and convex optimization problems [cf., 86]. [46] proposed an algorithm which globally solves certain classes of NLPs by decomposing the problem based on an appropriate partition of its decision variables. The work of [8] and [93] on the use of *relaxation and linearization techniques* [cf., 91], in combination with *spatial branching techniques* [cf., 93], has lead to the development of the two well-known global solvers `Couenne` [7] and `BARON` [89] for NLPs. Another solver that combines these type of techniques, together with techniques to exploit the problems structure is `GloMIQO`, developed by [73] for the solution of more general quadratically constrained quadratic programs with integer variables. More recently, specialized solution approaches have been developed for special classes of QP. In particular, [20] develop a special branch-and-cut algorithm for box constrained QPs based on using cuts derived from the boolean quadric polytope. Also, [19] develop new specialized cuts that are used within a spatial branch and bound algorithm to solve standard QPs. For further review of numerical and theoretical results on the solution of QPs using NLP techniques, we refer the reader to [54] and [47].

Besides NLP techniques, convex optimization techniques [cf., 9, 86] have also been used to address the solution of QPs. For example, [76] and later [64, 65], explored the use of semidefinite programming (SDP) as well as second-order cone relaxations to approximately or globally solve a QP.

More recently, [28] proposed a SDP-based branch and bound approach to globally solve box-constrained QPs; they reformulate a QP by adding the QP's corresponding Karush-Kuhn-Tucker (KKT) conditions as redundant constraints. Let us refer to this quadratically constrained quadratic program (QCQP) as $\mathrm{QP}_{\mathrm{KKT}}$. To solve $\mathrm{QP}_{\mathrm{KKT}}$, [28] construct a finite *KKT-branching* tree by branching on the resulting problem's complementarity constraints. SDP relaxations of $\mathrm{QP}_{\mathrm{KKT}}$ are used to obtain lower bounds at each node of the KKT-branching tree. On the other hand, to obtain upper bounds a (local) QP-solver based on NLP techniques is used. [32] improved the solution methodology of [28] by obtaining tighter lower bounds at each node of the KKT-branching tree. For that purpose, the *double non-negative* (DNN) relaxation of the *completely positive* reformulation [24] of $\mathrm{QP}_{\mathrm{KKT}}$ at each node of the KKT-branching tree is used. [32] provide a `MATLAB` implementation of their approach called `quadprogBB`. In this implementation, the `MATLAB` (local) QP solver `quadprog` is used to obtain the upper bounds while the algorithm proposed by [25] is used to obtain lower bounds, at each node of the KKT-branching tree. [32] show that this solution approach typically outperforms the solver `Couenne` and the approach proposed by [28] on a test bed of publicly available QP instances. This makes the solver `quadprogBB` a current benchmark for the global solution of QP problems.

The rest of the paper is organized as follows. In Section 2, we formally introduce the QP problem and present the theoretical results that serve as the foundation for the proposed solution approach. In Section 3, we illustrate the effectiveness of this approach by presenting relevant numerical results on test instances of the QP problem. To conclude, in Section 4, we provide conclusions and directions for future work.

2.2 Solution Approach for non-convex QPs

We consider the following quadratic programming problem

$$\begin{aligned} \text{QP}: \qquad\qquad \min \quad & \tfrac{1}{2}x^{\intercal}Hx + f^{\intercal}x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0, \end{aligned} \tag{2.1}$$

where $f \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and $H \in \mathbb{R}^{n\times n}$ is a symmetric matrix. Note that there is no assumption on the matrix H being positive semidefinite; that is, QP is in general a non-convex optimization problem [cf., 12].

Similar to [28] and [32], we assume that the feasible set of QP is nonempty and bounded. However, in what follows, no further assumption is made about the feasible set of QP.

2.2.1 Mixed-integer linear programming reformulation

After introducing the Lagrange multipliers $\mu \in \mathbb{R}^m$ for its equality constraints and $\lambda \in \mathbb{R}^n$ for its non-negativity constraints, the KKT conditions for QP are given by

$$Hx + f + A^{\intercal}\mu - \lambda = 0 \tag{2.2a}$$

$$x^{\intercal}\lambda = 0 \tag{2.2b}$$

$$Ax = b \tag{2.2c}$$

$$x \geq 0, \lambda \geq 0. \tag{2.2d}$$

In what follows, we will refer to the set

$$\Lambda_{\text{KKT}} = \{(x, \mu, \lambda) \in \mathbb{R}^{2n+m} : (x, \mu, \lambda) \text{ satisfy } (2.2\text{a}) - (2.2\text{d})\} \tag{2.3}$$

as the *KKT points* of QP.

Note that because the feasible set of QP (2.1) is a polyhedron, the KKT conditions (2.2) are first order necessary conditions for the optimal solutions of QP [see, e.g. 44, Thm. 3.3]. Thus, one can add these KKT conditions as redundant constraints in QP to obtain the following equivalent formulation of QP,

$$\begin{aligned} \min \quad & \tfrac{1}{2} x^\intercal H x + f^\intercal x \\ \text{s.t.} \quad & Hx + f + A^\intercal \mu - \lambda \ = 0 \\ & x^\intercal \lambda = 0 \\ & Ax = b \\ & x \geq 0, \lambda \geq 0. \end{aligned} \tag{2.4}$$

As shown by [49, Thm. 2.4], one can use the KKT conditions (2.2a)–(2.2c) to linearize the objective of (2.4). Namely, for any feasible solution $x \in \mathbb{R}^n$ of (2.4), we have

$$\frac{1}{2} x^\intercal H x + f^\intercal x = \frac{1}{2}(f^\intercal x - x^\intercal A^\intercal \mu + x^\intercal \lambda) = \frac{1}{2}(f^\intercal x - b^\intercal \mu).$$

As a result, problem (2.4) is equivalent to the following problem with a linear (instead of quadratic) objective.

$$\begin{aligned} \tfrac{1}{2} \min \quad & f^\intercal x - b^\intercal \mu \\ \text{s.t.} \quad & Hx + f + A^\intercal \mu - \lambda \ = 0 \\ & x^\intercal \lambda = 0 \\ & Ax = b \\ & x \geq 0, \lambda \geq 0. \end{aligned} \tag{2.5}$$

Notice that in (2.5), the complexity of QP is captured in the complementary constraints $x^\intercal \lambda = 0$. Next, we address the complementary constraints in (2.5) by using *Big-M* constraints. For that purpose, in Section 2.2.2, we derive upper bounds $U, V \in \mathbb{R}^n$ on the decision variables $x, \lambda \in \mathbb{R}^n$ of (2.5) such that there are (globally) optimal KKT points $(x, \mu, \lambda) \in \mathbb{R}^{2n+m}$ of QP satisfying $x \leq U$, $\lambda \leq V$. Using these upper bounds, one can show (see, Theorem 1) that a global optimal solution of QP can be obtained by solving the following MILP

$$\begin{aligned} \text{IQP}: \qquad\qquad \tfrac{1}{2}\min\quad & f^{\intercal}x - b^{\intercal}\mu \\ \text{s.t.}\quad & Hx + f + A^{\intercal}\mu - \lambda = 0 \\ & Ax = b \\ & 0 \le x_j \le z_j U_j \qquad\qquad j = 1,\dots,m \\ & 0 \le \lambda_j \le (1 - z_j)V_j \qquad j = 1,\dots,m \\ & z_j \in \{0,1\} \qquad\qquad\quad j = 1,\dots,m. \end{aligned} \tag{2.6}$$

Specifically, problem IQP is a MILP with the same optimal value as QP whose optimal solutions are optimal solutions of QP.

2.2.2 Bounding the primal and dual variables

As mentioned earlier, the first step in obtaining problem IQP is to derive explicit upper bounds $U, V \in \mathbb{R}^n$ such that there are optimal KKT points $(x, \mu, \lambda) \in \mathbb{R}^{2n+m}$ of QP satisfying $x \le U$, $\lambda \le V$.

Similar to [32], using the assumption that the feasible set of QP is non-empty and bounded, one can compute the upper bounds $U \in \mathbb{R}^n_+$ on the primal variables $x \in \mathbb{R}^n$ by setting:

$$U_j := \max\{x_j : Ax = b,\, x \ge 0\}, \tag{2.7}$$

for every $j = 1, \dots, n$.

Using assumptions stronger than ours, [32] show that Λ_{KKT}, the set of KKT points, is bounded. As the following example illustrates, under our weaker assumptions, the set Λ_{KKT} could be unbounded.

Example 2.2.1. Consider the instance of QP defined by setting

$$H = \begin{bmatrix} 2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad f = \begin{bmatrix} 2 \\ 4 \\ 3 \end{bmatrix} \qquad A = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Note that in this case the feasible region of QP is $\{[0, 1-t, t]^{\intercal} : 0 \le t \le 1\}$, which is bounded and non-empty. However, the set of KKT points Λ_{KKT} (2.3) is unbounded. Specifically, notice that for any $v \geq 1$ the following is a KKT point for QP :

$$x = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad \mu = \begin{bmatrix} v \\ -3-v \end{bmatrix} \qquad \lambda = \begin{bmatrix} -1+v \\ 0 \\ 0 \end{bmatrix}.$$

Thus, to handle the complementarity contraints in (2.5) using Big-M constraints, we do not try to obtain a bound for the value of the entries of $\lambda \in \mathbb{R}^m$ for all KKT points. Instead, in Theorem 1, we prove that there exist a bound that we can impose in the dual variables, without discarding all (globally) optimal KKT points of QP. For this purpose, we make use of fundamental results on the approximate solution of systems of linear equations [e.g., 55, 72].

Let us first define a particular instance of the well-known Hoffman bound [60], closely following the notation in [55].

Definition 2.2.2. *Fix the norm $\|\cdot\|_\alpha$ on $\mathbb{R}^n$ and the norm $\|\cdot\|_\beta$ on $\mathbb{R}^m$. Given $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, let $F := \{x \in \mathbb{R}^n_+ : Ax = b\}$. Let $\mathcal{H}_{A,b} \in \mathbb{R}$ be the smallest constant satisfying:*

$$\text{For all } y \in \mathbb{R}^n \text{ such that } Ay = b, \text{ there is } x \in F \text{ such that } \|x - y\|_\alpha \leq \mathcal{H}_{A,b}\|y^-\|_\beta. \quad (2.8)$$

Above, for any $y \in \mathbb{R}^n$, $y^- \in \mathbb{R}^n$ is the vector difined by $y_i^- = \max\{0, -y_i\}$, $i = 1, \ldots, n$. That is, Definition 2.2.2 corresponds to the Hoffman bound obtained when looking at perturbations of only the non-negative constraints of the polyhedron $F := \{x \in \mathbb{R}^n_+ : Ax = b\}$.

In what follows we will use the following notation to denote the dual norm associated to a given norm.

Definition 2.2.3. Given a norm $\|\cdot\|$ on $\mathbb{R}^n$, its associated *dual norm* on $\mathbb{R}^n$, denoted $\|\cdot\|^*$ is defined as:

$$\|x\|^* = \sup\{x^\intercal z : z \in \mathbb{R}^n, \|z\| \leq 1\}.$$

In particular, for any $x \in \mathbb{R}^n$, $\|x\|^*_\infty = \|x\|_1$.

Using Definition 2.2.2, we provide in Theorem 1 below, the desired bound to be used on the dual variables of QP in the IQP formulation (2.6).

Theorem 2.2.4. *Let $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$ be such that the set $F := \{x \in \mathbb{R}^n_+ : Ax = b\}$ is non-empty and bounded. Let $\mathcal{H}_{A,b}$ be defined by (2.8), and $\kappa := \max\{\|Hx\|^*_\alpha : Ax = b, x \geq 0\}$. Then, for each globally optimal solution x^* of QP, there exists $(\mu^*, \lambda^*) \in \mathbb{R}^m \times \mathbb{R}^n_+$ such that (x^*, μ^*, λ^*) is a KKT point of QP and $e^\intercal \lambda^* \leq (\kappa + \|f\|^*_\alpha)\mathcal{H}_{A,b}\|e\|_\beta$.*

Proof. Proof. Fix $M > (\kappa + \|f\|^*_\alpha)\,\mathcal{H}_{A,b}\|e\|_\beta$ and consider the following perturbed version of QP:

$$\begin{aligned} \min \quad & \tfrac{1}{2}x^\intercal Hx + f^\intercal x + Mt \\ \text{s.t.} \quad & Ax = b \\ & x \geq -te \\ & 0 \leq t \leq \delta, \end{aligned} \tag{2.9}$$

where e is the vector of all ones, and $\delta > 0$. Notice that the feasible set of (2.9) is a closed subset of $\{x \in \mathbb{R}^n : Ax = b,\ x \geq -\delta e\} \times [0, \delta]$ which is non-empty and bounded as F is non-empty, bounded, and the recession cone of $\{x \in \mathbb{R}^n : Ax = b,\ x \geq -\delta e\}$ is equal to the recession cone of F. Thus, the optimal value of (2.9) exists and it is attained.

Let (x^*, t^*) be an optimal solution of (2.9). Then, there exists $(\mu^*, \lambda^*, \rho^*, \omega^*) \in \mathbb{R}^{n+m+2}$ such that $(x^*, t^*, \mu^*, \lambda^*, \rho^*, \omega^*)$ satisfies the KKT conditions associated with problem (2.9)

$$\begin{aligned} Hx^* + f + A^\intercal\mu^* - \lambda^* \quad &= 0 \\ M - e^\intercal\lambda^* - \rho^* + \omega^* \quad &= 0 \\ (x^* - t^*e)^\intercal\lambda^* \quad &= 0 \\ t^*\rho^* \quad &= 0 \\ (\delta - t^*)\omega^* \quad &= 0 \\ Ax^* \quad &= b \\ x^* + t^*e \quad &\geq 0 \\ 0 \leq t^* \quad &\leq \delta \\ \lambda^*, \rho^*, \omega^* \quad &\geq 0, \end{aligned} \tag{2.10}$$

where $\mu^* \in \mathbb{R}^m$, $\lambda^* \in \mathbb{R}^n$, $\rho^* \in \mathbb{R}$, $\omega^* \in \mathbb{R}$, are respectively the Lagrangian multipliers of problem (2.9) associated with the linear constraints, lower bounds in the decision variables $x \in \mathbb{R}^n$, and lower and upper bounds on the decision variable $t \in \mathbb{R}$.

Now we claim that $t^* = 0$. In that case, notice that the set of optimal solutions for (2.9) is $\{(x^*, 0) \in \mathbb{R}^{n+1} : x^* \text{ is optimal for QP}\}$. Furthermore, the complementarity constraint

$(\delta - t^*)\omega^* = 0$ in (2.10) implies $\omega^* = 0$ and thus, from the equation $M - e^\intercal\lambda^* - \rho^* + \omega^* = 0$ in (2.10) and the fact that $\rho^* \geq 0$, it follows that (x^*, μ^*, λ^*) is a KKT point of QP with $e^\intercal\lambda^* \leq M$.

To show that $t^* = 0$, note that from Definition 2.2.2, it follows that there exists $x' \in F$ such that

$$\|x' - x^*\|_\alpha \leq \mathcal{H}_{A,b} t^* \|e\|_\beta. \tag{2.11}$$

In problem (2.9), $(x', 0)$ is a feasible solution and thus the objective value of $(x', 0)$ is no smaller than the objective value of (x^*, t^*). That is,

$$\frac{1}{2}x^{*\intercal}Hx^* + f^\intercal x^* + Mt^* \leq \frac{1}{2}x'^\intercal Hx' + f^\intercal x'.$$

Therefore,

$$\begin{aligned}
Mt^* &\leq \frac{1}{2}(x'^\intercal Hx' - x^{*\intercal}Hx^*) + f^\intercal(x' - x^*) \\
&= \frac{1}{2}(x' + x^*)^\intercal H(x' - x^*) + f^\intercal(x' - x^*) \\
&\leq \frac{1}{2}\|H(x' + x^*)\|_\alpha^* \|x' - x^*\|_\alpha + \|f\|_\alpha^* \|x' - x^*\|_\alpha \\
&\leq \left(\frac{1}{2}(\|Hx'\|_\alpha^* + \|Hx^*\|_\alpha^*) + \|f\|_\alpha^*\right)\|x' - x^*\|_\alpha.
\end{aligned}$$

Thus, using (2.11) we have

$$Mt^* \leq \left(\frac{1}{2}(\kappa + \kappa_\delta) + \|f\|_\alpha^*\right)\mathcal{H}_{A,b}\|e\|_\beta t^* \tag{2.12}$$

where $\kappa_\delta := \max\{\|Hx\|_\alpha^* : Ax = b, x \geq -\delta e, x \in \mathbb{R}^n\}$.

Since $\kappa_\delta \downarrow \kappa$ when $\delta \downarrow 0$, taking $\delta > 0$ small enough we have that

$$M > \left(\frac{1}{2}(\kappa + \kappa_\delta) + \|f\|_\alpha^*\right)\mathcal{H}_{A,b}\|e\|_\beta.$$

Thus, (2.12) implies that $t^* = 0$.

To finish the proof, fix x^*, an optimal solution of QP and let $M' = (\kappa + \|f\|_\alpha^*)\,\mathcal{H}_{A,b}\|e\|_\beta$. Let $S = \{\lambda \in \mathbb{R}^n_+ : \exists\ \mu \in \mathbb{R}^m \text{ such that } (x^*, \mu, \lambda) \in \Lambda_{\text{KKT}}\}$. From ((2.3)) S is a polyhedron, and we have proven that for all $M > M'$ there exists $\lambda' \in S$ satisfying $e^\intercal\lambda' \leq M$. We claim that this implies there exists $\lambda \in S$ satisfying $e^\intercal\lambda \leq M$. For the sake of contradiction, let

$P = \{\lambda \in \mathbb{R}^n : e^\intercal \lambda \leq M'\}$ and assume $S \cap P = \emptyset$. Since both $S, P \subseteq \mathbb{R}^n$ are polyhedrons, it follows from the Strong Separation Lemma that there exists $c \in \mathbb{R}^n, d \in \mathbb{R}$ with $c \neq 0$, such that: (i) $c^\intercal \lambda < d$ for all $\lambda \in P$, and (ii) $c^\intercal \lambda > d$ for all $\lambda \in S$. From (i) and P being a half-space, it follows that $c = se$, $d > sM'$ for some $s > 0$. Thus, $\frac{d}{s} > M'$, and therefore there exists $\lambda' \in S$ such that $e^\intercal \lambda' \leq \frac{d}{s}$, or equivalently $c^\intercal \lambda' \leq d$, contradicting (ii).

□

Remark 2.2.5. From Theorem 1, the constraint $e^\intercal \lambda \leq (\kappa + \|f\|_\alpha^*)\mathcal{H}_{A,b}\|e\|_\beta$ could be added in the IQP formulation of QP (2.6). Adding this constraint however has not led to improved solution times of IQP. Thus, this constraint is not used in the implementation of our proposed solution approach for QP.

If one fixes the norm $\|\cdot\|_\beta$ in Definition 2.2.2 to be the 1-norm $\|\cdot\|_1$, one can directly impose bounds on each of the dual variables associated with the non-negativity constraints in QP.

Proposition 1. *Let $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$ be such that the set $F := \{x \in \mathbb{R}^n_+ : Ax = b\}$ is non-empty and bounded. Let $\widetilde{\mathcal{H}}_{A,b}$ be defined by* (2.8) *with the norm $\|\cdot\|_\beta$ being the 1-norm $\|\cdot\|_1$, and $\kappa := \max\{\|Hx\|_\alpha^* : Ax = b, x \geq 0\}$. Then, for all globally optimal solutions x^* of QP, there exists $(\mu^*, \lambda^*) \in \mathbb{R}^m \times \mathbb{R}^n_+$ such that (x^*, μ^*, λ^*) is a KKT point of QP and $\lambda_i^* \leq (\kappa + \|f\|_\alpha^*)\widetilde{\mathcal{H}}_{A,b}$.*

Proof. Proof. The proof is analogous to that of Theorem 1, after considering (instead of (2.9)), the perturbed version of QP given by $\min\{\frac{1}{2}x^\intercal Hx + f^\intercal x + \widetilde{M}(e^\intercal y) : Ax = b, x \geq -y, 0 \leq y \leq \delta e\}$. □

However, as it will be shown in Section 2.2.3, the freedom to choose both the norms $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$ in Definition 2.2.2, is important in obtaining the best possible bounds for dual variables associated with the non-negativity constraints in QP.

2.2.3 Computation of the dual bounds

Theorem 1 provides the bounds needed for the reformulation of QP as IQP (2.6) in terms of the Hoffman constant $\mathcal{H}_{A,b}$ introduced in Definition 2.2.2. Next, we discuss how this constant can be obtained in closed-form for important special classes of QP, as well as how it can be computed for general classes of QP.

Standard Quadratic Programming.

Consider the *standard quadratic program* (SQP):

$$\text{SQP}: \qquad \min_{x \in \Delta} \ \tfrac{1}{2} x^\intercal H x + f^\intercal x \tag{2.13}$$

where $\Delta = \{x \in \mathbb{R}^n : \sum_{i=1}^n x_i = 1, x \geq 0\}$, is the standard simplex. The SQP problem is fundamental in optimization and arises in many applications [see, e.g., 14]. Next, we show that in this case, the Hoffman bound $\mathcal{H}_{A,b}$ introduced in Definition 2.2.2 can be computed in closed-form for suitable choices of the norm $\|\cdot\|_\alpha$ on $\mathbb{R}^n$ and the norm $\|\cdot\|_\beta$ on $\mathbb{R}^m$.

Proposition 2. *Consider the norm $\|\cdot\|_\alpha = \|\cdot\|_1$ on $\mathbb{R}^n$ and the norm $\|\cdot\|_\beta = \|\cdot\|_1$ on $\mathbb{R}^m$. Let $A = e^\intercal$ and $b = 1$. Then $\mathcal{H}_{A,b} = 2$.*

Proof. Proof. Let $y \in \mathbb{R}^n$ such that $e^\intercal y = 1$ be given. Let $I = \{i \in \{1, \ldots, n\} : y_i \geq 0\}$ and $I^c = \{1, \ldots, n\} \setminus I$, and consider the case $I^c \neq \emptyset$ (otherwise, the statement follows by letting $x = y$ in Definition 2.2.2). Note that $e^\intercal y = 1$ implies $I \neq \emptyset$ and that $\sum_{i \in I} y_i = 1 - \sum_{i \in I^c} y_i = 1 + \|y^-\|_1$. Let $x \in \mathbb{R}^n$ be defined by setting $x_i = 0$ for all $i \in I^c$, and $x_i = \frac{1}{1+\|y^-\|_1} y_i$ for all $i \in I$. Clearly, $x \in \Delta$. Furthermore, for any $i \in I^c$, $|x_i - y_i| = -y_i$. Also for any $i \in I$, we have $|x_i - y_i| = \frac{\|y^-\|_1}{1+\|y^-\|_1} y_i$. Thus, $\|x - y\|_1 = -\sum_{i \in I^c} y_i + \frac{\|y^-\|_1}{1+\|y^-\|_1} \sum_{i \in I} y_i = 2\|y^-\|_1$. That is, $\mathcal{H}_{A,b} \leq 2$. To show $\mathcal{H}_{A,b} \geq 2$, consider $y = (n, -1, \ldots, -1)$. For any $x \in \Delta$, it follows that $\|x - y\|_1 = |x_1 - n| + \sum_{i=2}^n |x_i + 1| = 2n - 1 - x_1 + \sum_{i=2}^n x_i = 2(n - x_1) \geq 2(n-1) = 2\|y^-\|_1$. $\square$

Thus, (2.13) can be reformulated as IQP by letting:

$$U = e \text{ and } V \geq Me, \tag{2.14}$$

with

$$M = 2n \left(\|H\|_{\infty,\infty} + \|f\|_\infty\right), \tag{2.15}$$

where we have used that

$$\kappa = \max\{\|Hx\|_\infty : e^\intercal x = 1, x \in \mathbb{R}^n_+\} \leq \max_{i,j \in \{1,\ldots,n\}} |H_{ij}| =: \|H\|_{\infty,\infty}.$$

Remark 2.2.6. *It is worth mentioning that a proof similar to the one given in Proposition 2* *shows that if* $\|\cdot\|_1$ *is replaced with* $\|\cdot\|_\infty$ *in Proposition 2, the corresponding Hoffman constant would be equal to* $n-1$*. However, this leads to a weaker bound* V *than the one given in* (2.14).

Quadratic Programming with Box Constraints.

Now, consider the *box-constrained QP* (BoxQP)

$$\text{BoxQP}: \qquad \begin{aligned} \min \quad & \tfrac{1}{2}x^\intercal Hx + f^\intercal x \\ \text{s.t.} \quad & l \le x \le u, \end{aligned} \tag{2.16}$$

where $l, u \in \mathbb{R}^n$ are given bounds on the primal variables of BoxQP satisfying (w.l.o.g.) $l < u$ (component-wise). Problem BoxQP is equivalent to the following QP problem:

$$\begin{aligned} \min \quad & \tfrac{1}{2}x^\intercal Hx + (Hl+f)^\intercal x \\ \text{s.t.} \quad & x + s = u - l \\ & x \ge 0, s \ge 0. \end{aligned} \tag{2.17}$$

Next, we show that in this case, the Hoffman bound $\mathcal{H}_{A,b}$ introduced in Definition 2.2.2 can be computed in closed-form for a suitable choice of the norm $\|\cdot\|_\alpha$ on $\mathbb{R}^n$ and the norm $\|\cdot\|_\beta$ on $\mathbb{R}^m$.

Proposition 3. *Consider the norm* $\|\cdot\|_\alpha = \|\cdot\|_\infty$ *on* $\mathbb{R}^n$ *and the norm* $\|\cdot\|_\beta = \|\cdot\|_\infty$ *on* $\mathbb{R}^m$*. Let* I *denote the identity matrix in* $\mathbb{R}^{n\times n}$*,* $b \in \mathbb{R}^n_+$*, and* $A = [I, I]$*. Then* $\mathcal{H}_{A,b} = 1$*.*

Proof. Proof. Let $(y, z) \in \mathbb{R}^{2n}$ such that $y + z = b$ be given. Define $x = y^+ - z^-$ and $s = z^+ - y^-$. We claim $(x, s) \in F = \{(x, s) \in \mathbb{R}^{2n}_+ : x + s = b\}$. To show this notice first that $x + s = y^+ - z^- + z^+ - y^- = y + z = b$. Now let $i \in \{1, \dots, n\}$. If $z_i^- = 0$ then $x_i = y_i^+ \ge 0$. Thus assume $z_i^- > 0$. Then $z_i^+ = 0$ and $x_i = b_i - s_i = b_i + y_i^- \ge 0$. Thus $x \ge 0$. Similarly $s \ge 0$. To finish, notice that $\|(y,z) - (x,s)\|_\infty = \|(-y^- + z^-, -z^- + y^-)\|_\infty = \|(y^-, z^-)\|_\infty = \|(y,z)^-\|_\infty$. This shows that $\mathcal{H}_{A,b} \le 1$. To show $\mathcal{H}_{A,b} \ge 1$, let $y = -e$ and $z = b + e$. For any $(x,s) \in F$, it follows that $\|(x,s) - (y,z)\|_\infty \ge |x_1 + 1| \ge 1 = \|(y,z)^-\|_\infty$. □

Using Proposition 3, we obtain that (2.17) can be reformulated as IQP by letting:

$$U = \begin{bmatrix} u - l \\ u - l \end{bmatrix} \text{ and } V \ge Me, \tag{2.18}$$

with

$$M = \min(n\|H\|_{\infty,\infty}\|u-l\|_1, \|H\|_{1,1}\|u-l\|_\infty) + \|f + Hl\|_1, \tag{2.19}$$

where we have used that

$$\kappa = \max\{\|Hx\|_1 : x + s = u - l, x, s \in \mathbb{R}^n_+\} \le \min(n\|H\|_{\infty,\infty}\|u-l\|_1, \|H\|_{1,1}\|u-l\|_\infty),$$

where $\|H\|_{1,1} := \sum_{i \neq j \in \{1,\dots,n\}} |H_{ij}|$.

General Quadratic Programming.

Note that to compute an appropriate M value in Theorem 1, it is enough to let $M > (\kappa + \|f\|_\alpha^*)\overline{\mathcal{H}}_A\|e\|_\beta$ for some constant $\overline{\mathcal{H}}_A \ge \mathcal{H}_{A,b}$. As shown bellow, $\overline{\mathcal{H}}_A$ can be computed in general using [55, Theorem 3.2].

Proposition 4. *Fix the norm $\|\cdot\|_\alpha$ on $\mathbb{R}^n$ and the norm $\|\cdot\|_\beta$ on $\mathbb{R}^m$. Let $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$ be such that the set $F := \{x \in \mathbb{R}^n_+ : Ax = b\} \neq \emptyset$. Also, let*

$$\bar{\sigma}(A) = \left\{(\mu^+, \mu^-, \lambda) \in \mathbb{R}^{m+n} : \|A^\intercal(\mu^+ - \mu^-) - \lambda\|_\alpha^* \le 1, \mu^+, \mu^- \in \mathbb{R}^m_+, \lambda \in \mathbb{R}^n_+\right\},$$

and

$$\overline{\mathcal{H}}_A = \max\{\|(\mu^+, \mu^-, \lambda)\|_\beta^* : (\mu^+, \mu^-, \lambda) \text{ is an extreme point of } \bar{\sigma}(A)\}. \tag{2.20}$$

Then, $\overline{\mathcal{H}}_A \ge \mathcal{H}_{A,b}$.

Proof. Proof. First notice that for all $y \in \mathbb{R}^n$,

$$\min_{x \in F} \|x - y\|_\alpha = \min\{\|x - y\|_\alpha : A'x \le b', x \in \mathbb{R}^n\}$$

where

$$A' = \begin{bmatrix} A \\ -A \\ -I \end{bmatrix}, b' = \begin{bmatrix} b \\ -b \\ 0 \end{bmatrix}.$$

Thus, it follows from [55, Theorem 3.2] that

$$\min_{x \in F} \|x - y\|_\alpha \leq \overline{\mathcal{H}}_A \left\| \begin{pmatrix} (Ay-b)^+ \\ (Ay-b)^- \\ y^- \end{pmatrix} \right\|_\beta , \tag{2.21}$$

after identifying $\overline{\mathcal{H}}_A = K_{\alpha\beta}(A')$, $\bar{\sigma}_\alpha(A) = \sigma_\alpha(A')$ [see, 55, Theorem 3.2].

From (2.21), it follows that for any $y \in \mathbb{R}^n$ such that $Ay = b$, then

$$\min_{x \in F} \|x - y\|_\alpha \leq \overline{\mathcal{H}}_A \left\| \begin{pmatrix} 0 \\ 0 \\ y^- \end{pmatrix} \right\|_\beta = \overline{\mathcal{H}}_A \left\| y^- \right\|_\beta . \tag{2.22}$$

Also, from (2.22), $\overline{\mathcal{H}}_A \geq \mathcal{H}_{A,b}$ follows from Definition 2.2.2, as $\mathcal{H}_{A,b}$ is the smallest constant satisfying (2.22). □

In order to use Proposition 4 to reformulate QP (2.1) as IQP (2.6), one can first, similar to [32], normalize the primal variables of QP to be between 0 and 1. Namely, under the boundedness assumption considered here, one has that QP is equivalent to:

$$\begin{aligned} \min \quad & \tfrac{1}{2} x^\intercal \tilde{H} x + \tilde{f}^\intercal x \\ \text{s.t.} \quad & \tilde{A} x = b \\ & x \geq 0, \end{aligned}$$

where $\tilde{H}_{ij} := H_{ij} U_i U_j$, $\tilde{f}_i := f_i U_i$, and $\tilde{A}_{ki} := A_{ki} U_i$ for all $i, j \in \{1, \ldots, n\}$, $k \in \{1, \ldots, m\}$, and $U \in \mathbb{R}^n_+$ is given by (2.7). Now, using Proposition 4, and choosing the norm $\|\cdot\|_\alpha = \|\cdot\|_\infty$ on $\mathbb{R}^n$ and the norm $\|\cdot\|_\beta = \|\cdot\|_\infty$ on $\mathbb{R}^m$, we obtain that QP (2.1) can be reformulated as IQP (2.6) by letting $H = \tilde{H}$, $f = \tilde{f}$, $A = \tilde{A}$,

$$U = e, \text{and } V \geq Me,$$

with

$$M = \left(\|\tilde{H}\|_{1,1} + \|\tilde{f}\|_1 \right) \overline{\mathcal{H}}_A, \tag{2.23}$$

where we have used that $\kappa = \max\{\|\tilde{H}x\|_1 : x \leq e, x \in \mathbb{R}^n_+\} \leq \|\tilde{H}\|_{1,1}$.

In the case that one chooses the norm $\|\cdot\|_\alpha = \|\cdot\|_1$ on $\mathbb{R}^n$ and the norm $\|\cdot\|_\beta = \|\cdot\|_1$ on $\mathbb{R}^m$, then $M = n(n\|\tilde{H}\|_{\infty,\infty} + \|\tilde{f}\|_\infty)\overline{\mathcal{H}}_A$. However, empirical results on test instances shows that this latter M is weaker than the bound obtained using (2.23).

Obtaining an efficient way to compute the constant $\overline{\mathcal{H}}_A$ in (2.20) is however still an open question [see, e.g., 55, 81, 97]. For illustrative purposes, in Section 2.2.3, we show the results of using an algorithm recently proposed in [81] to compute $\overline{\mathcal{H}}_A$, and the corresponding bound M in (2.23).

An alternative and efficient way to compute bounds on the dual variables of a general instance of QP, is to use the bounds on the dual variables proposed by [32, Proposition 3.1] which are valid for QP instances having a strictly non-negative feasible solution (i.e., a feasible solution satisfying $x > 0$), and can be computed by solving a LP [cf., 32, eq. (19)]. Specifically, notice that after obtaining the primal bounds $U \in \mathbb{R}^n$ using (2.7), problem QP is equivalent to

$$\begin{aligned} \min \quad & \tfrac{1}{2}x^\intercal Hx + f^\intercal x \\ \text{s.t.} \quad & Ax = b \\ & 0 \le x \le U. \end{aligned} \tag{2.24}$$

Following the notation used thus far and letting $\rho \in \mathbb{R}^n$ be the dual variables associated with the upper bound constraints on the variables $x \in \mathbb{R}^n$ in (2.24), it follows from the KKT conditions of (2.24) that any of its optimal solutions must satisfy:

$$Hx + f + A^\intercal\mu - \lambda + \rho = 0 \tag{2.25a}$$

$$x^\intercal\lambda = 0, (U - x)^\intercal\rho = 0 \tag{2.25b}$$

$$Ax = b \tag{2.25c}$$

$$x \ge 0, \lambda \ge 0, \rho \ge 0.$$

Also, after multiplying (2.25a) by a feasible solution $x \in \mathbb{R}^n$ of (2.24) and using (2.25b), (2.25c), it follows that any optimal solution of (2.24) also satisfies:

$$x^\intercal Hx + f^\intercal x + b^\intercal\mu + U^\intercal\rho = 0.$$

Then, if QP has a feasible solution $x \in \mathbb{R}^n$ satisfying $x_i > 0$, $i = 1, \ldots, m$, it follows from [32, Proposition 3.1] that bounds on the dual variables $V \in \mathbb{R}^n$ required for the MILP reformulation IQP of QP can be computed by solving the following LP:

$$V_j = \max \left\{ \lambda_j : \begin{array}{c} Hx + f + A^\intercal \mu - \lambda + \rho = 0 \\ H \bullet X + f^\intercal x + b^\intercal \mu + U^\intercal \rho = 0 \\ 0 \le X_{ij} \le U_i U_j, i, j = 1, \dots, n \\ 0 \le x \le U, \lambda \ge 0, \rho \ge 0, X \in \mathcal{S}^n \end{array} \right\}, \tag{2.26}$$

where $H \bullet X$ indicates the trace of the matrix HX, the matrix $X \in S^n$ represents the *linearization* of the matrix $xx^\intercal \in \mathcal{S}^n$, and $\mathcal{S}^n$ is the set of $n \times n$ real symmetric matrices.

Remark 2.2.7. In [32], eq. (18) is used to refine the dual variable bounds after scaling the problem so that its variables are between zero and one. However, this refinement of the bounds is not necessary to obtain their result in Proposition 3.1. The refined version of these bounds is however the one implemented in `quadprogIP`, the implementation of our solution approach for general instances of QP.

Bound comparison

In light of the bounds on the dual variables discussed here and the dual bounds proposed by [32, eq. (19)], it is natural to compare their values and computing times for different instances of QP. Before doing so, however, it is important to emphasize that the dual bounds proposed here can be imposed on QP, even if the dual feasible set of QP is unbounded.

Example 2.2.8 (Example 2.2.1 revisited)**.** Recall the problem discussed in Example 2.2.1, and consider the norm $\|\cdot\|_\alpha = \|\cdot\|_\infty$ on $\mathbb{R}^n$ and the norm $\|\cdot\|_\beta = \|\cdot\|_\infty$ on $\mathbb{R}^m$. It is not difficult to see that in this case $\mathcal{H}_{A,b} \le 1$, and $\kappa = 1$. Since $\|f\|_1 = 9$, and $\|e\|_\infty = 1$, it follows that $M = (\kappa + \|f\|_1)\mathcal{H}_{A,b} \le 10$. Thus, for this problem we can bound the dual variables with the constraint

$$[\lambda_1, \lambda_2, \lambda_3]^\intercal \le 11[1, 1, 1]^\intercal. \tag{2.27}$$

In fact, notice that the following optimal solution of the problem

$$x^* = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad \mu^* = \begin{bmatrix} 0 \\ -3 \end{bmatrix} \qquad \lambda^* = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

satisfies the dual bounds (2.27). On the other hand, from Example 2.2.1, it follows that

$$\max\{\lambda_1 : (x, \mu, \lambda) \in \Lambda_{\text{KKT}}\} = \infty.$$

Thus, dual bounds for this problem cannot be computed using [32, eq. (19)].

In Table 2.1, we compare the bounds obtained in Section 2.2.3 with the bounds obtained using [32, eq. (19)] for a number of randomly selected SQP instances. From Table 2.1, it is clear that the bounds obtained using Theorem 1, and specifically, eq. (2.15) are tighter than the ones obtained using [32, eq. (19)] (and labeled "RLT bounds" in Table 2.1 for the *reformulation linearization techniques* (RLT) used to derive them) on a random sample of SPQ instances. In fact, this is the case for all the SQP instances considered in Section 2.3.

	Thm. 1 bound		RLT bound	
SQP Instance	Value	Time (s)	Value	Time (s)
spar030-060-1.mat:	5,520	0.0000	10,628	0.4952
spar030-070-3.mat:	5,880	0.0000	16,448	0.4694
spar050-040-1.mat:	9,200	0.0000	18,925	1.4016
spar050-050-3.mat:	9,800	0.0000	29,719	1.8674
spar060-020-1.mat:	11,040	0.0000	13,255	1.8344
spar070-075-1.mat:	13,440	0.0000	90,316	17.0129
spar080-025-2.mat:	15,680	0.0001	31,792	13.8083
spar080-025-3.mat:	15,040	0.0001	38,115	13.8468
spar090-025-2.mat:	17,640	0.0001	44,646	22.8278
spar090-025-3.mat:	16,920	0.0001	49,459	23.3063
spar090-050-3.mat:	17,460	0.0001	95,726	37.7146
spar090-075-1.mat:	17,280	0.0001	148,416	49.4807
spar100-050-2.mat:	19,600	0.0001	115,192	67.8767
spar100-050-3.mat:	19,400	0.0001	119,634	67.6607

Table 2.1: Comparison of bounds on $e^{\intercal}\lambda$ obtained using (2.15) ("Thm. 1 bound" columns) vs. using [32, eq. (19)] ("RLT bound" columns), together with corresponding computation times for a random sample of SQP instances.

Similarly, in Table 2.2, we compare the bounds obtained in Section 2.2.3 with the bounds obtained using [32, eq. (19)] for a number of randomly selected SQP instances. From Table 2.2, it is clear that the bounds obtained using Theorem 1, and specifically, eq. (2.19) are tighter than the ones obtained using [32, eq. (19)] on a random sample of BoxPQ instances. In fact, this is the case for all the BoxQP instances considered in Section 2.3. Both in Table 2.1 and Table 2.2 the time differences are a result of the bounds resulting from Theorem 1 being computed from the closed-form formulas (2.15), (2.19), while the

RLP bounds are obtained by solving a linear program [32, eq. (19)].

	Thm. 1 bound		RLT bound	
BoxQP Instance	Value	Time (s)	Value	Time (s)
spar020-100-3.mat:	9,708	0.0001	73,497	0.2955
spar030-060-1.mat:	13,097	0.0001	152,505	0.5289
spar030-070-1.mat:	14,887	0.0002	172,151	0.5001
spar030-070-2.mat:	15,909	0.0001	176,093	0.5362
spar030-070-3.mat:	16,827	0.0001	184,397	0.4815
spar030-080-1.mat:	18,259	0.0001	219,338	0.4578
spar030-080-2.mat:	18,532	0.0001	205,091	0.4539
spar030-080-3.mat:	18,585	0.0001	202,502	0.5233
spar040-060-3.mat:	25,889	0.0001	388,914	0.7893
spar040-080-1.mat:	31,929	0.0001	524,796	0.9596
spar040-090-2.mat:	37,109	0.0001	589,155	1.1884
spar070-025-1.mat:	30,162	0.0182	819,461	7.1339

Table 2.2: Comparison of bounds on $e^\intercal \lambda$ obtained using (2.19) ("Thm. 1 bound" columns) vs. using [32, eq. (19)] ("RLT bound" columns), together with corresponding computation times for a random sample of BoxQP instances.

In Table 2.3, we compare the bounds obtained in Section 2.2.3 with the bounds obtained using [32, eq. (19)] for a number of randomly selected general QP instances. From Table 2.3, it is clear that the bounds obtained using Theorem 1, and specifically, eq. (2.23) are weaker than the ones obtained using [32, eq. (19)] on a random sample of general QP instances. In fact, this is the case for all the general QP instances considered in Section 2.3. In Table 2.3 the time differences are a result of the bounds resulting from Theorem 1 being computed using an algorithm whose complexity is exponential on the size of the constraint matrix of the problem [81], while the RLP bounds are obtained by solving a linear program [32, eq. (19)].

From the results in Table 2.1, Table 2.2, and Table 2.3, it is clear that using the bound of Theorem 1 can lead to tighter bounds on the QP dual variables $\lambda \in \mathbb{R}^n_+$ than the ones obtained using [32, eq. (19)] when a tight bound on the Hoffman constant $\mathcal{H}_{A,b}$ used in Theorem 1 can be computed efficiently.

As illustrated in Example 2.2.8, the dual QP bounds obtained from Theorem 1 can be used even if the dual feasible set of QP is unbounded. In such case, it is not possible to use the `quadprogBB` solution methodology proposed by [32] to solve the problem, as the methodology requires (through a condition on the primal QP problem) the dual feasible set of QP to be bounded. To illustrate this (see Table 2.4), we modify some general QP test

	Thm. 1 bound		RLT bound	
General QP Instance	Value	Time (s)	Value	Time (s)
st_e26.mat	49,200	0.0575	8,828	0.0742
st_fp4.mat	830,297	1.6961	594	0.1380
st_fp5.mat	47,263,557	125.8385	792	0.2807
st_glmp_kky.mat	83,410	22.4100	339	0.1013
st_glmp_ss1.mat	33,757	6.5787	429	0.1067
st_m1.mat	556,912,094	0.0655	19,060,333	0.3594
st_pan2.mat	6,017	0.0544	1,494	0.0939
st_jcbpaf2.mat:	-	-	96,969	0.2757
st_ph10.mat	1,320	0.5757	27	0.0679
st_ph2.mat	69,951	0.0601	8,043	0.1132
st_qpc_m0.mat	372	0.0573	35	0.0501
qp20_10_2_1.mat	69,846	0.0671	2,500	0.5206
qp30_15_1_4.mat	44,451	0.0621	1,661	0.7943
qp30_15_2_4.mat	41,625	0.0602	1,122	0.9512
qp40_20_1_4.mat	85,008	0.0548	8,700	1.3852
qp40_20_4_1.mat	523,052	0.0606	16,371	3.3573
qp50_25_1_2.mat	149,684	0.0668	12,476	2.4781
qp50_25_1_4.mat	169,868	0.0676	17,623	2.1617

Table 2.3: Comparison of bounds on $e^\intercal \lambda$ obtained using (2.23) ("Thm. 1 bound" columns) vs. using [32, eq. (19)] ("RLT bound" columns), together with corresponding computation times for a random sample of (general) QP instances. Dash "-" indicates that the time limit of 1800 sec has been reached without computing the bound.

instances in a simple way to make their dual feasible set unbounded. The modification we use is to pick the first variable x_1 of the instance and add the constraint $x_1 = x_1^*$, where x_1^* is the value of x_1 in an optimal solution of the problem (i.e., this results in a problem that likely violates the interior condition required by [see, 32, preceding Prop. 3.1]). As shown in Table 2.4, these modified instances can be correctly solved using the approach proposed here with the bounds (2.23), while `quadprogBB` of [32] is unable to solve them due to the unboundedness of some of the dual variables of the modified version of the problem. Specifically, Table 2.4 provides the name of the original instance (1st column), its optimal value (2nd column), the constraint that is added to the problem to make its dual feasible set unbounded while leaving its optimal value unchanged (3rd column), the value of the M bound (2.23) computed as a bound for the dual variables while still retaining at least an optimal solution (4th column), the optimal solution for the modified version of the instance obtained with `quadprogIP` (5th column), and the number of the dual variable of the modified version of the instance that `quadprogBB` detects to be unbounded which results in `quadprogBB` not being able to solve the modified version of the problem.

QP instance	Optimal Value	Fixed Variable	M bound (2.23)	`quadprogIP` Modified instance Optimal Value	`quadprogBB` detected unbounded dual
qp20_10_1_1.mat	-13.189	x_1=0.4660	2.07E+04	-13.189	43-th
qp20_10_1_2.mat	11.6662	x_1=1.0000	1.89E+04	11.6662	66-th
qp20_10_1_4.mat	-18.3137	x_1=0.0000	1.29E+04	-18.3137	45-th
qp20_10_2_1.mat	-3.2442	x_1=0.0000	6.98E+04	-3.2442	45-th
qp20_10_2_2.mat	8.5919	x_1=0.0000	1.27E+04	8.5919	45-th
qp20_10_2_4.mat	6.5794	x_1=0.0000	9.54E+03	6.5794	45-th
qp20_10_3_1.mat	-30.179	x_1=0.0000	7.10E+04	-30.179	45-th
qp20_10_3_2.mat	-15.0508	x_1=0.0000	4.70E+04	-15.0508	45-th
qp20_10_3_4.mat	-12.665	x_1=0.0000	1.49E+04	-12.665	45-th
qp30_15_1_1.mat	32.9577	x_1=0.0000	1.96E+05	32.9577	67-th
qp30_15_1_3.mat	0.525	x_1=0.0000	3.91E+04	0.525	67-th
qp30_15_1_4.mat	9.2296	x_1=0.0000	4.45E+04	9.2296	67-th
qp30_15_2_3.mat	-2.0693	x_1=1.0000	3.18E+04	-2.0693	98-th
qp30_15_2_4.mat	1.2862	x_1=0.0000	4.16E+04	1.2862	67-th
qp40_20_1_3.mat	-2.7293	x_1=0.0000	8.30E+04	-2.7293	89-th
qp50_25_1_4.mat	13.8442	x_1=0.0000	1.70E+05	13.8442	111-th
qp50_25_2_4.mat	-6.8577	x_1=0.0000	2.80E+05	-6.8577	111-th
qp50_25_3_2.mat	35.9871	x_1=0.0000	2.15E+05	35.9871	111-th

Table 2.4: Solution of QP test instances modified to have an unbounded dual feasible set using `quadprogIP`.

2.3 Computational results

In this section, we provide a detailed description of the implementation of the solution approach for QP problems described in the previous sections. Also, we illustrate the performance of the solution approach by presenting the results of numerical experiments on a diverse set of QP test problems.

2.3.1 Problem instances

To test the performance of the proposed solution approach for QP, we use the set of BoxQP (2.16), Globallib (`http://www.gamsworld.org/global/Globallib.htm`), CUTEr [54], and RandQP test problems used in [32, Section 4.2 and Table 1]. In addition to these test problems, we consider the following QP test instances:

- SQP. Standard quadratic programming instances (2.13) are created by replacing the constraints of each of the BoxQPs considered in [32, Section 4.2 and Table 1] by the constraint that the decision variables belong to the standard simplex of appropriate dimension.

- SQP30, SQP50 (see, `http://or.dei.unibo.it/library/msc`). A set of 300 SQP instances used for test purposes in [20].

- StableQP. These instances are particular SQPs resulting from the problem of computing the *stability number* of a graph [see, e.g., 74]. We use instances of this type arising from a class of graphs that have been used for testing purposes in the literature [see, e.g., 38, Section 4.2.2]. A more detailed description of these instances is presented in Section 2.3.1.

- Scozzari/Tardella (see, `http://or.dei.unibo.it/library/msc`). A set of 14 SQP instances used for test purposes in [20, 90].

- QPLIB2014 (see, `http://www.lamsade.dauphine.fr/QPlib2014/doku.php`). Nine nonconvex quadratic instances are selected from this test set. Four of the instances which are SQP instances are added to the SQP test set, and the other five instances are BoxQP instances, which are added to the BoxQP test set.

Similar to [32], Table 2.5 provides a summary of the basic information of all the test instances. In Table 2.5, n denotes the range of the number of decision variables required to formulate the corresponding problem instance using m_{ineq} inequality constraints, and m_{eq} equality constraints. Also, *density* denotes the corresponding density range for the matrix defining the quadratic problem's objective.

Type	# Instances	n	$m_{ineq} + m_{eq}$	density
StableQP	8	[5, 26]	[0,1]	[0.30, 0.60]
SQP	90	[20, 100]	[0, 90]	[0.19, 0.99]
BoxQP	90	[20, 100]	[0, 0]	[0.19, 0.99]
Globallib	83	[2, 100]	[1,52]	[0.01, 1]
CUTEr	6	[4, 12]	[0, 13]	[0.08, 1]
RandQP	64	[20, 50]	[14, 35]	[0.23, 1]
SQP30	150	[30, 30]	[0,1]	[1, 1]
SQP50	150	[50, 50]	[0,1]	[1, 1]
Scozzari/Tardella	14	[30, 1000]	[0,1]	[0.25, 1]

Table 2.5: Statistics of the test QP instances.

StableQP instances

For any graph G, the inverse of $\alpha(G)$, the *stability number* of G, can be computed by solving the following SQP [see, e.g., 74].

$$\frac{1}{\alpha(G)} = \min_{x \in \Delta} x^\intercal (A + I)x, \tag{2.28}$$

where $A \in \mathcal{S}^n$ is the adjacency matrix of the undirected graph $G(V, E)$ with number of vertices $\|V\| = n$, and set of edges $E \in V \times V$. Also I is the identity matrix of appropriate dimensions.

The StableQP instances are obtained by solving (2.28) for a class of graphs G_k, $k = 1, \dots$ introduced in [38] that have proven to be hard instances for approximation methods for $\alpha(G)$ proposed in [15, 22, 38, 40].

2.3.2 Implementation details

The solution approach for QP proposed here is implemented as follows. First, explicit upper and lower bounds for the instance's decision variables are obtained. Then, the problem instance is reformulated as QP by linearly shifting its decision variables, and adding slack variables to the problem as necessary (e.g., (2.17)). The upper bounds on the added slack variables are computed using (2.7) to obtain the primal variable upper bounds $U \in \mathbb{R}^n$. Upper bounds $V \in \mathbb{R}^n$ on the dual variable are calculated using the methods described in Section 2.2.3 (see (2.14), (2.18) and (2.26)). Finally, CPLEX 12.5.1 (cf., `http://www-eio.upc.edu/lceio/manuals/CPLEX-11/html/`) is used to solve IQP. The following parameter settings are used for CPLEX MILP solver:

- Max_time: This is the user specified maximum running time of the algorithm and is set to 10^4 seconds. Any problem taking longer than this value to be solved will be deemed as "out of time".

- Tol: The solver will stop when

$$\frac{|\text{bestnode} - \text{bestinteger}|}{1^{-10} + |\text{bestinteger}|} \le 10^{-6}.$$

  For the interested reader, the definition of the parameters *bestnode* and *bestinteger* can be found in [34]. Here, it suffices to say that this criteria is consistent with `quadprogBB` stopping criteria [cf., 32], which is

$$\frac{\text{Greatest upper bound} - \text{current lower bound}}{max\{1, |\text{Greatest upper bound}|\}} \leq 10^{-6}.$$

- Other parameters of the CPLEX MILP solver such as TolXInteger, Max_iter, BranchStrategy, Nodeselect, are set to their default values.

We refer to the procedure described in this section to solve QP as `quadprogIP`, which is coded using `Matlab R2014a`, and is publicly available at `https://github.com/xiawei918/quadprogIP`.

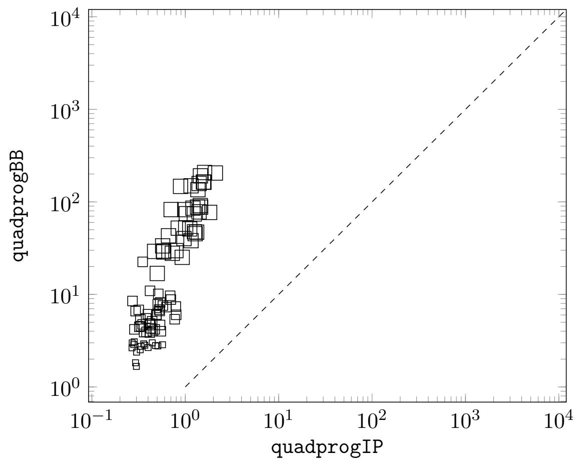
2.3.3 Numerical performance

In order to test the performance of the `quadprogIP` methodology proposed in Section 2.3.2, the QP test instances discussed in Section 2.3.1 are solved using `quadprogIP`, the `quadprogBB` solver introduced by [32], the NLP solver `BARON 17.8.9` of [89], and the `CPLEX 12.7.0.0` QP solver. All tests are done using `Matlab R2014b`, together with CPLEX 12.7.0., on a AMD Opteron 2.0 GHz machine with 32GB memory and 16 cores (each core is a 2.0 GHz. 64 bit architecture), from the COR@L laboratory (cf., `http://coral.ise.lehigh.edu/`).

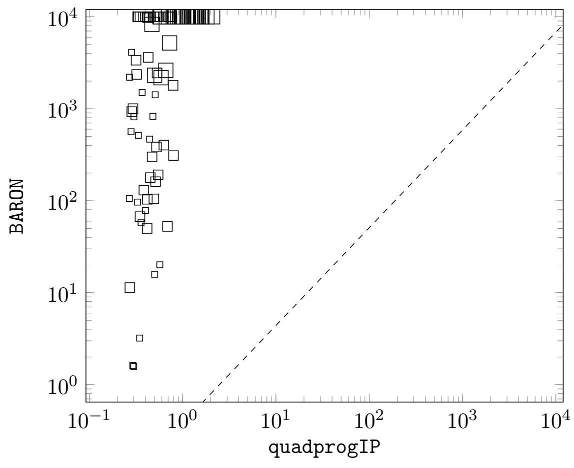
Similar to [32], to compare the performance between `quadprogIP` and `quadprogBB`, `quadprogIP` and `BARON`, and `quadprogIP` and `CPLEX`, we plot the solution time it takes to solve a particular QP instance with two of the solvers as a square in a 2D plane, where the y-axis denotes either `quadprogBB`'s, `BARON`'s, or `CPLEX`'s solution time and the x-axis denotes `quadprogIP` 's solution time. The dashed line in the plots indicates the $y = x$ line in the plane, that represents equal solution times. Thus, a square that is above the diagonal line indicates an instance for which it takes the solver represented on y-axis more solution time to solve than `quadprogIP`. Furthermore, the size of the square illustrates the size (number of decision variables) of the instance. That is, smaller squares represent "smaller" size instances while bigger squares represent "bigger" size instances. In the figures below, only instances in which at least one of the methodologies solves the problem within the maximum time allowed are displayed.

Results on SQP instances.

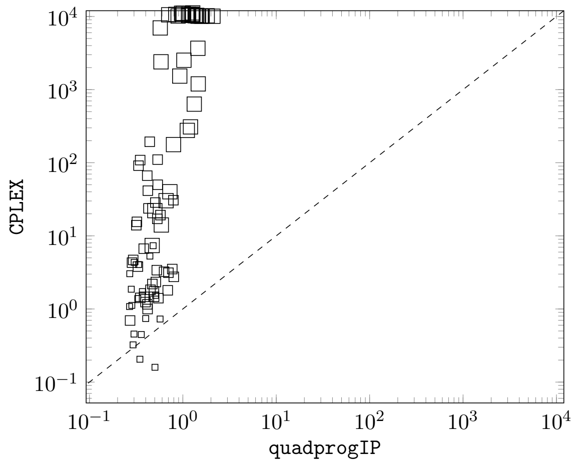
The results for the SQP test instances are shown in Figure 2.1. Note that a different scale is used in the axes of Figures 2.1a, 2.1b, and 2.1c.



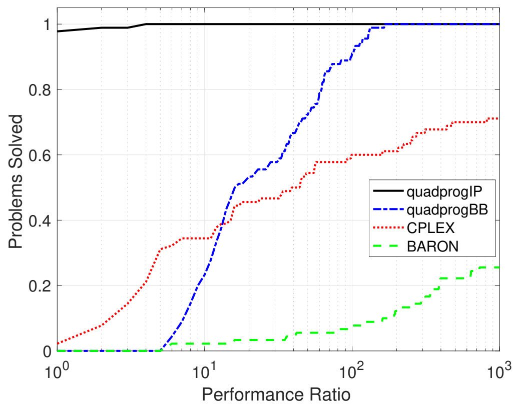
(a) `quadprogIP` vs `quadprogBB`.



(b) `quadprogIP` vs `BARON`.



(c) `quadprogIP` vs `CPLEX`.



(d) Performance profile for SQP instances.

Figure 2.1: Solution time in seconds of SQP instances. Size of squares illustrates size of the instance. A square at the maximum value of an axis represents an instances for which the solver in that axis reached maximum running time without solving it.

Figure 2.1a shows that `quadprogIP` clearly outperforms `quadprogBB` by solving all SQP instances in a time that is one to two orders of magnitude faster than `quadprogBB` and specially in the larger instances. Similarly, Figure 2.1b shows that `quadprogIP` clearly outperforms `BARON` by solving all SQP instances in a time that is one to two orders of magnitude faster than `BARON`, and specially in the larger instances. Although `CPLEX` solves two small-scale instances faster than `quadprogIP`, again, in general `quadprogIP` outperforms `CPLEX` by orders of magnitude in terms of solution time (see, Figure 2.1c). As Figures 2.1a, 2.1b, and 2.1c illustrate, the performance of `quadprogIP` against the other solvers improves as the SQP instance becomes larger. The performance profile in Figure 2.1d summarizes the clear advantages of solving the very important class of SQP instances with the proposed `quadprogIP` solution approach.

Results on SQP30 and SQP50 instances.

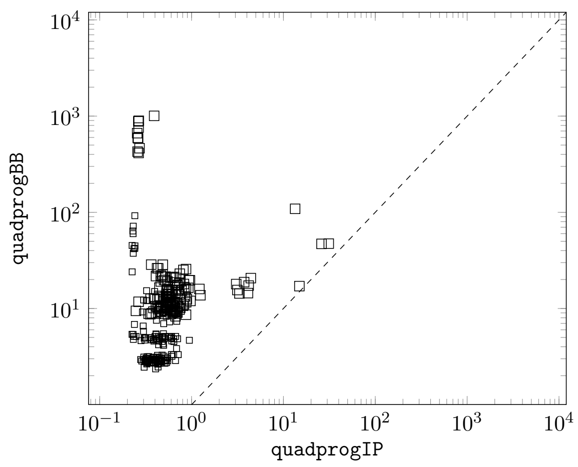
As Figure 2.2 shows, the results on the SQP instances SQP30 and SQP50 from [20] is very similar to the ones presented in Section 2.3.3. As with the set of SQP instances, only `CPLEX` is able to solve a few instances faster than `quadprogIP`; however, in general `quadprogIP` outperforms the other solvers by orders of magnitude in terms of solution time.

Results on StableQP instances.

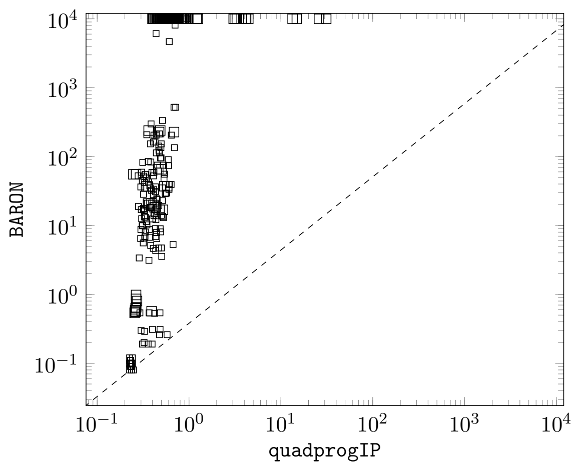
In line with the performance of `quadprogIP` on SQP, SQP30, and SQP50 instances, it is interesting to see in Table 2.6 that `quadprogIP` clearly outperforms `quadprogBB`, `BARON`, and `CPLEX` in the StableQP instances (see, Section 2.3.1). In fact, while `quadprogIP` solves each of the instances in less than a second, `quadprogBB`, and `CPLEX` are unable to solve the instances beyond $k \geq 4$ within the maximum allowed solution time of 10^4 seconds, while `BARON` is unable to solve the instances beyond $k \geq 3$ within the maximum allowed solution time.

	Solution Time (s.)			
k	`quadprogIP`	`quadprogBB`	`BARON`	`CPLEX`
1	0.34	3.67	8.93	0.39
2	0.25	6.28	2573.77	8.75
3	0.34	12.56	-	685.70
4	0.43	-	-	-
5	0.49	-	-	-
6	0.51	-	-	-
7	0.46	-	-	-
8	0.49	-	-	-

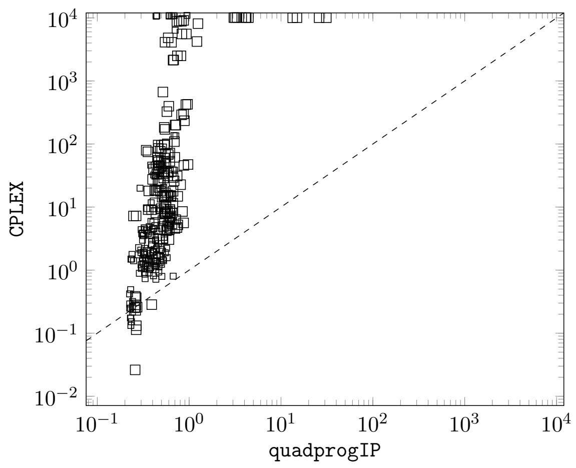
Table 2.6: Solution time in seconds for StableQP instances. Dash "-" indicates that the solver was unable to solve the instance within the maximum allowed time.



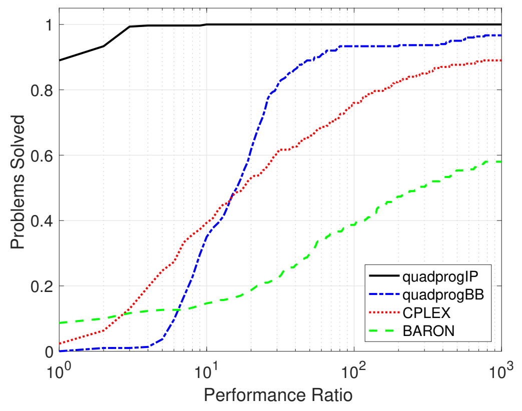
(a) `quadprogIP` vs `quadprogBB`.



(b) `quadprogIP` vs `BARON`.



(c) `quadprogIP` vs `CPLEX`.



(d) Performance profile for SQP30 and SQP50 instances.

Figure 2.2: Solution time in seconds of SQP30 and SQP50 instances. Size of squares illustrates size of the instance. A square at the maximum value of an axis represents an instances for which the solver in that axis reached maximum running time without solving it.

Results on Scozzari/Tardella instances.

The Scozzari/Tardella from [90] are composed of much larger-scale instances of SQP than the ones considered so far. Table 2.7, as with the previously discussed groups of standard QP instances, clearly shows that `quadprogIP` is able to solve these instances faster than the other solvers, and is able to solve more large-scale instances than the other solvers.

Results on BoxQP instances.

In Figure 2.3, we compare the performance of `quadprogIP` on the BoxQP instances against the other three selected solvers. It is clear from Figure 2.3a that while `quadprogIP` outper-

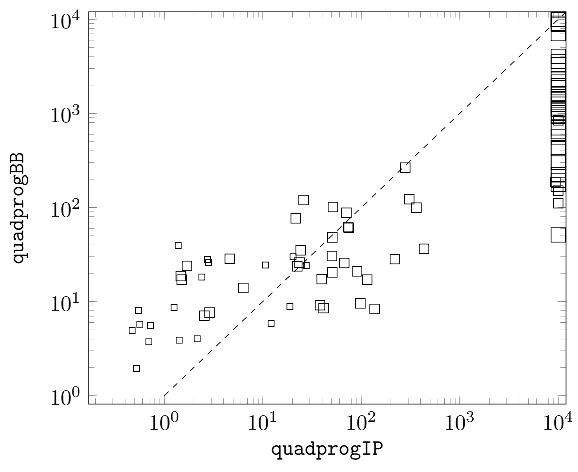
Scozzari/Tardella instance	Solution Time (s.)			
	`quadprogIP`	`quadprogBB`	`BARON`	`CPLEX`
Problem_30x30_0.75.mps.mat:	0.51	5.27	39.32	5.56
Problem_50x50_0.75.mps.mat:	11.78	48.29	-	2,162.13
Problem_100x100_0-1.mps.mat:	1.54	1,412.16	223.54	154.71
Problem_100x100_0.5.mps.mat:	6.71	319.82	-	-
Problem_100x100_0.75.mps.mat:	36.76	1,519.61	-	-
Problem_200x200_0-1.mps.mat:	36.86	-	-	9,995.67
Problem_200x200_0.5.mps.mat:	175.15	-	-	-
Problem_500x500_0-1.mps.mat:	240.09	-	-	-
Problem_500x500_0.25.mps.mat:	2,092.48	-	-	-
Problem_1000x1000_0.25.mps.mat:	-	-	-	-
Problem_Q30.mps.mat:	0.54	4.27	-	-
Problem_Q50.mps.bar.mat:	2.45	8,476.70	-	-
Problem_Q100.mps.bar.mat:	4.71	-	-	-
Problem_Q150.mps.mat:	20.89	-	-	-

Table 2.7: Solution time in seconds for Scozzari/Tardella instances. Dash "-" indicates that the solver was unable to solve the instance within the maximum allowed time.

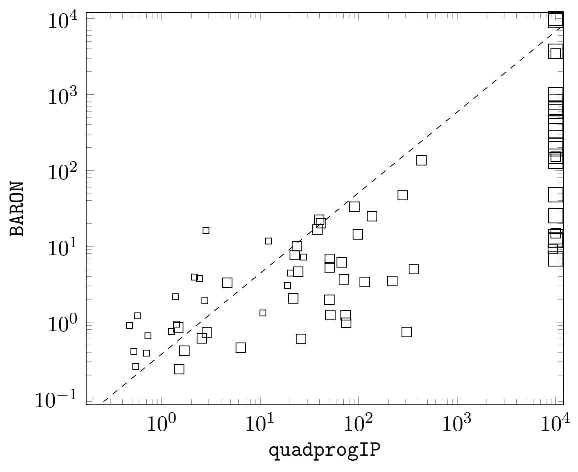
forms `quadprogBB` in the smaller BoxQP instances (ranging between 20–60 decision variables), `quadprogBB` outperforms `quadprogIP` for larger BoxQP instances (ranging between 60–100 decision variables), where `quadprogIP` is typically unable to solve the instance within the 10^4 maximum solution time.

Figure 2.3b shows the performance of `quadprogIP` and `BARON` on the BoxQP test set. It is clear that `BARON` outperforms `quadprogIP` in most BoxQP instances. Although for instances with less than 40 decision variables the solution time of `quadprogIP` is not significantly longer than that of `BARON`. Figure 2.3c shows that `CPLEX` performs much better than `quadprogIP` on all BoxQP instances. Figure 2.3d summarizes these results, where it is clear that `CPLEX` and `BARON` are the best solvers for these BoxQP instances.

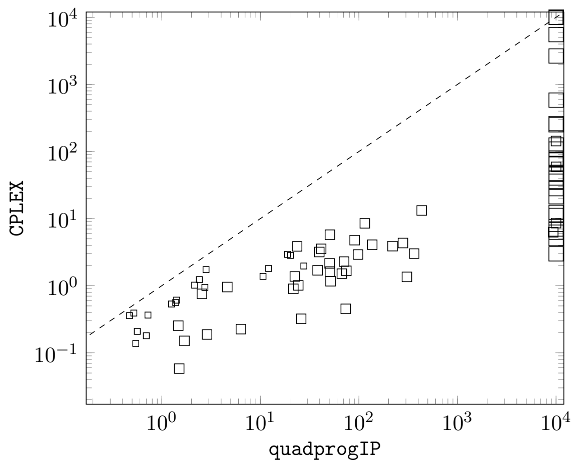
It is worth mentioning that the performance of `quadprogIP` on BoxQP instances can be improved by adding appropriate valid constraints to the IQP (2.6) formulation of the BoxQP. This valid constraints can be derived from [58, Prop. 1] [see also, 20, Lemma 4]. Specifically, notice that the IQP (2.6) corresponding to (2.17) can be written as:



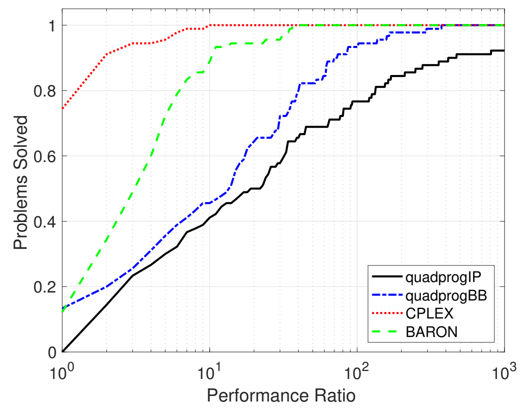
(a) `quadprogIP` vs `quadprogBB`.



(b) `quadprogIP` vs `BARON`.



(c) `quadprogIP` vs `CPLEX`.



(d) Performance profile for BoxQP instances.

Figure 2.3: Solution time in seconds of BoxQP instances. Size of squares illustrates size of the instance. A square at the maximum value of an axis represents an instances for which the solver in that axis reached maximum running time without solving it.

$$\begin{aligned}
\tfrac{1}{2}\min \quad & (Hl+f)^\intercal x-(u-l)^\intercal \mu \\
\text{s.t.} \quad & \begin{bmatrix} H & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ s \end{bmatrix}+\begin{bmatrix} f+Hl \\ 0 \end{bmatrix}+\begin{bmatrix} I \\ I \end{bmatrix}\mu-\begin{bmatrix} \lambda^x \\ \lambda^s \end{bmatrix}=0 \\
& x+s=u-l \\
& 0\le x_i\le z_i^x(u_i-l_i) && i=1,\dots,n \qquad (2.29)\\
& 0\le s_i\le z_i^s(u_i-l_i) && i=1,\dots,n \\
& 0\le \lambda_i^x\le (1-z_i^x)V_i && i=1,\dots,n \\
& 0\le \lambda_i^s\le (1-z_i^s)V_i && i=1,\dots,n \\
& z_i^x, z_i^s\in\{0,1\} && i=1,\dots,n.
\end{aligned}$$

Then, from [58, Prop. 1], it follows that the constraints

$$z_i^x + z_i^s = 1, \text{for all } i \in \{1, \dots, n\} \text{ such that } H_{ii} < 0, \tag{2.30}$$

are valid constraints for the optimal solutions of (2.29) when $V_i, i = 1, \dots, n$ is given by (2.18).

When added to (2.29), the valid constraints (2.30) improve the solution time of the approach proposed here to globally solve BoxQP problems. Although the `quadprogIP` code does not include the strengthening constraints (2.30) for BoxQPs, the results illustrated on Figure 2.4 show how adding the valid constraints (2.30) improves the solution time on a set of `spar` BoxQP instances ranging on size between 20-40 variables with density between 30-100. In particular, with the addition of these constraints, `quadprogIP` outperforms `quadprogBB` on these instances.

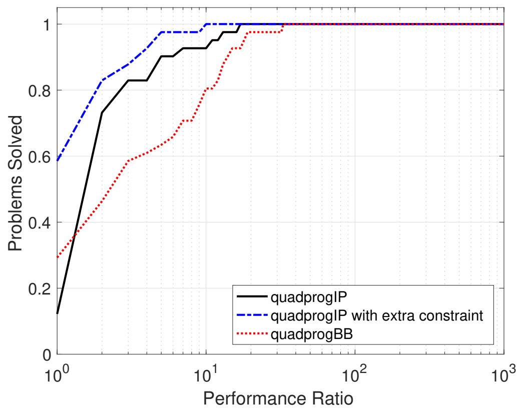
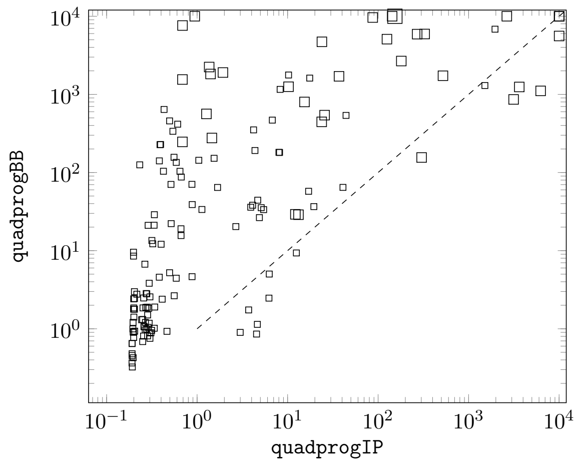


Figure 2.4: Performance profile for `spar` BoxQP instances anging on size between 20-40 variables with density between 30-100. Extra constraints refer to adding constraints (2.30) in the `quadprogIP` solver.

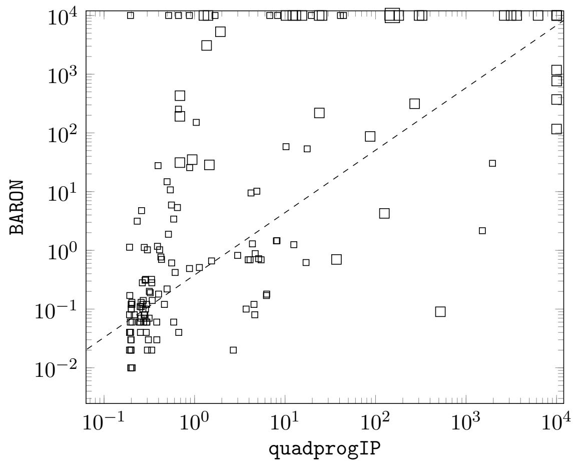
Results on CUTEr, Globallib, and RandQP instances.

In Figures 2.5, we compare the performance of `quadprogIP` on the CUTEr, Globallib, and RandQP instances against the other solvers. As Figure 2.5a illustrates, except for a few instances, `quadprogIP` has shorter solution times than `quadprogBB` on the more general CUTEr, Globallib, and RandQP instances of QP. Moreover, `quadprogIP` typically solves these problems about 10 times faster than `quadprogBB`. For these CUTEr, Globallib, and RandQP we find nine (9) instances that are successfully solved by `quadprogIP` but not by quadprogBB within the maximum allowed solution time of 10^4 seconds.

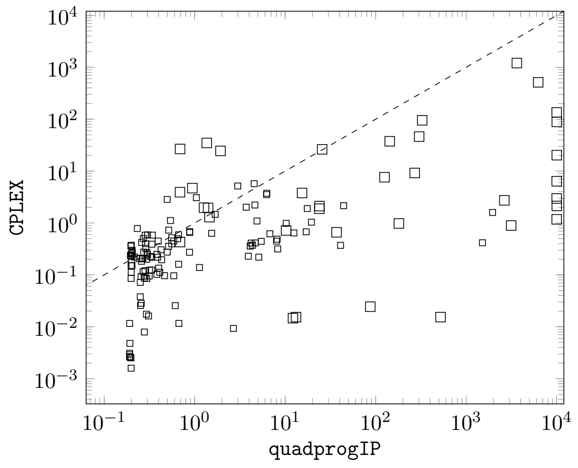
As for BARON, it can be seen from Figures 2.5b that quadprogIP is faster on most of the CUTEr, Globallib, and RandQP instances, with quadprogIP being able to solve a fair number of instances that BARON is not able to solve within the maximum allowed time of 10^4 seconds. On the other hand, CPLEX is able to solve most of the CUTEr, Globallib, and RandQP instances faster than quadprogIP; however, still a number of instances are solved faster than CPLEX, and most instances are solved by quadprogIP in a time no larger than 10 times the solution time of CPLEX. Figure 2.5d summarizes these results.



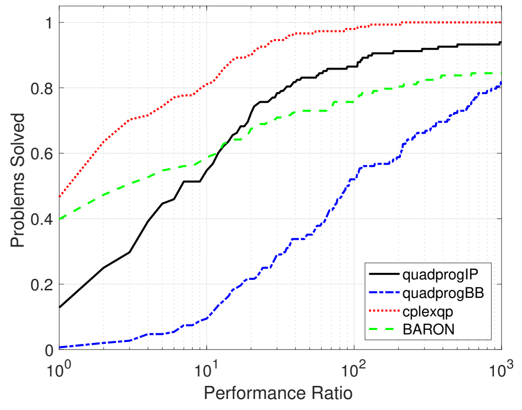
(a) quadprogIP vs quadprogBB.



(b) quadprogIP vs BARON.



(c) quadprogIP vs CPLEX.



(d) Performance profile for CUTEr, Globallib, and RandQP instances.

Figure 2.5: Solution time in seconds of CUTEr, Globallib, and RandQP instances. Size of squares illustrates size of the instance. A square at the maximum value of an axis represents an instances for which the solver in that axis reached maximum running time without solving it.

2.4 Conclusion

In this paper, we present a new simple and effective approach for the global solution of (non-convex) linearly constrained quadratic problems (QP) by combining the use of the problem's necessary KKT conditions together with state-of-the-art integer programming solvers. This is done via a reformulation of the QP as a mixed-integer linear program (MILP). We show that in general, this MILP reformulation can be obtained for QPs with a bounded primal feasible set via fundamental results related to the solution of perturbed linear systems of equations [see, e.g., 55]. In practice, `quadprogIP` is shown to typically outperform by orders of magnitude `quadprogBB`, `BARON`, and `CPLEX` on standard QPs. For general QPs, `quadprogIP` outperforms `quadprogBB`, outperforms `BARON` in most instances, while `CPLEX` performs the best on these instances. For box-constrained QPs, `quadprogIP` has a comparable performance to `quadprogBB` and `BARON` in small- to medium-scale instances, but is outperformed by these solvers on large-scale instances, while `CPLEX` performs the best on box-constrained QP instances. Also, unlike `quadprogBB`, the solution approach proposed here is able to solve QP instances whose dual feasible set is unbounded. The performance of this methodology on standard QP problems allows for the potential use of this solution approach as a basis for the solution of *copositive programs* [cf., 42]. Which is an interesting direction of future research work.

The proposed IP formulation of general QPs requires the computation of certain type of Hoffman bound [see, e.g., 60] on the system of linear equations defining the problem's feasible set. Thus, obtaining general and effectively computable bounds of this type is an interesting open question.

We finish by mentioning that a basic implementation of the proposed solution approach referred as `quadprogIP` is publicly available at `https://github.com/xiawei918/quadprogIP`, together with pointers to the test instances used in the article for the numerical experiments, and the raw data of all the solution times used to construct the figures throughout the article in the PDF file `raw data.pdf`.

Chapter 3

Multi-Commodity Network Flow Problems with Resource Constraints

3.1 Introduction

Air transportation is a steadily growing market that has seen a 60% growth over the past decade [1]. A recent advance in the technology of electric Vertical Take-off and Landing aircrafts (eVTOLs) has made on-demand aviation transportation a practical solution to improve urban mobility, as well as alleviate the pressure on ground transportation. eVTOLs, which are similar to helicopters, have several advantages that make them the ideal vehicle for an on-demand aviation network. Environment-wise, eVTOLs are quieter, that is, the noise generated is no louder than ground traffic noise at peak. eVTOLs are also more environmentally friendly as they are electrically powered. The maximum cruising speed of eVTOLs is about 170 miles per hour, and a fully charged battery allows the eVTOL to travel up to 120 miles [61], making eVTOLs a feasible and efficient transportation option for inner and even inter-city transit.

A basic on-demand aviation transportation network (or eVTOL network) consists of the hubs, the eVTOLs, and the passengers. The hubs provide parking decks for idling eVTOLs to park and recharge, and operating eVTOLs to take-off and land. The hubs are also the only locations where passengers can load or unload from the eVTOLs. The number and locations of the hubs are predetermined optimally based on the traffic pattern of the city. According to the solution of a facility location model, for a typical metropolitan area, 5 hubs placed optimally can serve about 30% to 50% of the traffic demand. eVTOLs are the primary transportation vehicle for efficient transit between hubs. An eVTOL can accommodate up to 3 passengers, not including the pilot. The passengers may utilize the eVTOL network by traveling to a nearby hub by foot or any form of ground transportation, and take an eVTOL to the hub that is closest to their destination, then get to their destination by foot or ground transportation.

The development and operation of such an on-demand aviation transportation network involves strategic decisions which are vital for the success of the operation. The feasibility and efficiency of the network, in other words, the passenger throughput and time savings, are some of the most important aspect to consider when operating the network, and the routing of the eVTOLs in the network is the key to an efficient operation.

In this article, we consider a multi-commodity network flows model (MCNF) model to help determine optimal routes of the eVTOLs on an on-demand aviation network. The eVTOLs and passengers can be viewed as commodities in the network, and routing them is equivalent to finding the optimal flow of each commodity through the network. However, the flow of passengers between hubs is constrained by the availability of the eVTOLs, and the flow of eVTOLs is constrained by the remaining battery level of the eVTOLs. The optimal flow of both passengers and eVTOLs is the flow that transport most passengers to their respective destination on time.

The organization of the article is as follows: Section 3.1 gives a general introduction of the problem. Section 3.2 summarizes the literatures related to eVTOLs. Section 3.3 provides a more detailed description and the assumptions of the problem. Section 3.4 proposes a formal mathematical model for the problem. Section 3.5 discusses two heuristics that help to generate initial feasible solutions and improve on incumbent feasible solutions, allowing to speed up the solution time of the model. Finally, Section 3.6 summarizes the performance of the heuristics over a set of instances of the problem.

3.2 Literature Review

Since eVTOLs are a relatively new technology that has become popular in recent years, the lines of research on eVTOLs are limited. One line of research focuses on the design aspect of the eVTOLs. [11] studied the electric multirotor design for eVTOLs. The work discussed the method of propulsion component selection for eVTOL propulsion system design, and presented a framework for both analysis of an existing propulsion system, as well as optimization of a propulsion system given a set of requirements for the vehicle. Electric propulsion is also discussed in [41], where a study is conducted on how the vertical flight vehicle may be designed to reduce cost by considering electric propulsion. According to [41], electric propulsion can simplify power transmission comparing to mechanical drive trains, and thus potentially reduce the cost of maintenance. The work further examined this potential from a reliability standpoint. On an optimal control aspect of eVTOLs, [82] focuse on the formulation of the fixed final time multiphase optimal control problem, where the energy consumption of the eVTOL is used as performance index. It is mainly concerned with using a multiphase optimal control solution to ensure the eVTOL meets the required time of arrival, and achieve the most energy efficient arrival trajectory at the same time. On a different topic, [77] investigates the management and operation of a fleet of eVTOLs for on demand aviation. It further discussed preliminary requirements for On-Demand Mobility air operations control centers. Key functional requirements are put forward related to vehicle safety, customer experience, and airspace integration. The most relevant work with respect to our problem is [48], where a case study of utilizing eVTOLs for cargo delivery in the San Francisco bay area is studied. In particular, the cargo delivery is carried out in two phases, where the first phase involves using eVTOLs to transport the cargo to warehouses, and second phase refers to the last-mile delivery by car. An optimization model is formulated to determine the optimal location of warehouses such that the maximum amount of package demand is satisfied. The detailed performance statistics of the eVTOL network is studied, which includes sizing, mission performance, recharge time requirements and daily package throughput.

3.3 Problem Description

The goal of using mathematical modeling techniques for the eVTOL on-demand transportation problem is to analyze the impact of different parameters on the dynamics of the eVTOL network. To be more specific, how much passenger demand can be fulfilled by introducing on demand eVTOL transportation, how many eVTOLs are needed, how should the flights of eVTOLs be scheduled, and how long is the average waiting time for passengers are all questions of interest. In this article, we mainly focus on answering the question of network throughput, which is to determine the maximum number of passengers the eVTOLs can transport to their destination in a given city. This statistic is of high interest, because it pertains to the eVTOL network's operation performance, and its ability to alleviate the burden on the ground traffic. More importantly, it helps decide if the operation will be economically feasible.

Many factors could affect the performance of the eVTOL transportation network, so practical assumptions are made in order for a compact but still realistic model to be formulated. The time horizon of the problem is one day. We simplify the time component by discretizing the timeline into equal sized time-bands, with a customized time-bandwidth. We only consider the set of passengers who are traveling long distances, between 20 miles to 120 miles in the city. We assume that all the passengers' information is known at the beginning of the day. The information includes the pick-up and drop-off time, and the pick-up and drop-off location of the passenger. To be more specific, the latitude and longitude of the passenger's origin and destination. The drop off time is assumed to be the latest time at which the passenger needs to get to his or her destination. It is assumed that all passengers have to get to their destinations before their latest arrival time, either by eVTOL or by ground transportation. If taking an eVTOL cannot get the passenger to his or her destination on time, then the passenger is assumed to arrive at the destination at the latest arrival time by ground transportation. The number and locations of hubs are assumed to be given. In practice, the locations of the hubs may be determined by solving a facility location model, in which the maximum number of passengers are covered. A passenger is said to be covered when there is a hub that is within the defined radius of the passenger's origin, and similarly when another hub within a defined radius of the destination. The potential location of the hubs can be obtained by clustering on all passengers' origins and destinations, and the exact locations are selected out of the potential locations based on the number of hubs and the coverage of passenger demand. There is also a capacity for each hub, which is the maximum number of eVTOLs it can hold at any given time. For simplicity, all hubs are assumed to have the same capacity. eVTOLs also have capacity, where a maximum of 3 passengers can be onboard besides the pilot. eVTOLs are electricity powered, and the capacity of the battery is 140 kw. At the cruising speed of 150 mph, 71kwh power is required [61]. Since the eVTOLs consume battery energy when operating, it is necessary to ensure the eVTOLs are not operating with insufficient battery levels. We assume that when an eVTOL is parked on the ground, it is always recharging. The recharging amount is assumed to be a linear function of time between 25% to 90%, and it takes 30 minutes to charge up the battery to 80%. For simplicity, the recharging rate is assumed to be a linear function with respect to time, and the slope is approximated to be 0.062 kw/s. Clearly, to ensure the safety of passengers, the energy level of the eVTOLs has to be considered in determining the routes of eVTOLs.

3.4 Model Formulation

We formulate the eVTOL on-demand transportation problem as a MCNF problem with resource constraints, where the resource is the battery of the eVTOLs. A path of a passenger or an eVTOL is determined by a series of movement through time, and to explicitly represent the path, we construct a time-space network to illustrate the flow of passengers and eVTOLs through time.

A time-space network consists of two axis, the y-axis denotes time, and the x-axis denotes space. Since the time component is discretized, any location (origin, destination or hub) at a given time is represented as a node. In the network, a node has two coordinates, where the first coordinate indicates the location of the node, and the second coordinate indicates the time of the node. An arc connecting two nodes indicates a flow in the network, from the first node's location at the first node's time, to the second node's location at the seconds node's time. In other words, the time spent traveling on the arc is the difference between the second node's time and the first node's time. For instance, a time space network with 1 passenger (denoted passenger i), 2 hubs and 5 minutes discretized intervals is shown in Figure 3.1.

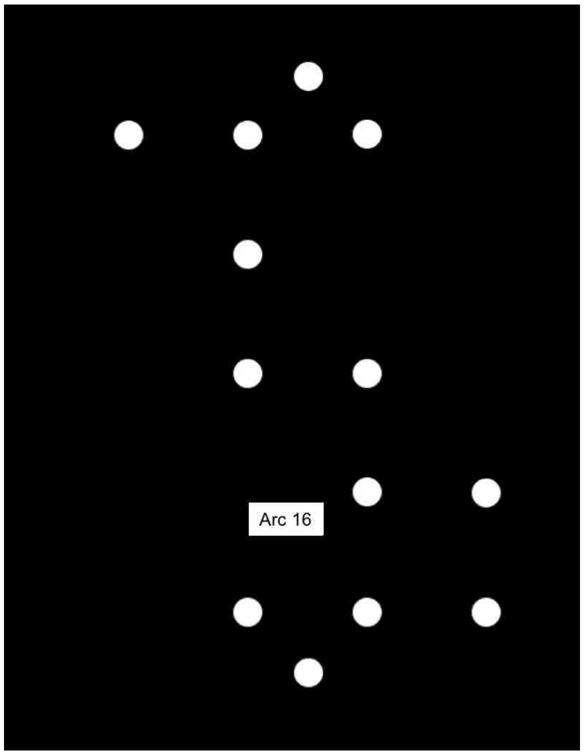


Figure 3.1: Time space network with 5 minutes intervals

Note the time-bandwidth is 5 minutes, starting from 00:00 of the day. From left to right, each vertical line represent the timeline of a location, namely origin, Hub 1, Hub 2, and destination. Each node in the network represents a location at a certain time point. For example, node O1 in Figure 3.1 represents there is a passenger i at his origin at time 00:00, in other words, there is a supply of 1 of passenger i at node O1. And D2 represents the destination of passenger i, and the latest arrival time is 00:20, also there is a demand of passenger i at D2. An arc connecting two nodes represents the flow of either an eVTOL or a passenger from one location to another, from the time at the start node until the time at the end node. There are two classes of arcs, one class is passenger arcs, and the other class is eVTOL arcs. Each class of arcs may have different types of arcs. For a passenger, there are 6 types of arcs. An arc that connects an origin to a hub is an origin-hub arc, for example arc 1 in Figure 3.1, representing a passenger getting to a hub from his or her origin. For each passenger, there may be multiple origin-hub arcs, depending on how many hubs are within the predefined radius of the origin of the passenger. Similarly, an arc that connects a hub to a destination is a hub-destination arc, such as arc 4 and 5. There may be multiple hub-destination arcs coming from different hubs, meaning there are multiple hubs that are within the predefined radius of the destination of the passenger, these arcs are not flight arcs, and the time associated with the arc is calculated as the ground transportation time. There may also be multiple hub-destination arcs from the same hub, this may happen if the passenger has the option to take eVTOL flights at different times that arrive at the hub at different times. For example, in Figure 3.1, if the passenger takes arc 2, then he will arrive at hub 2 at 00:10, and he can take arc 4 to get to his destination. It could also happen that the passenger takes arc 3 due to eVTOL unavailability, then he will arrive at hub 2 at 00:15, and he may only take arc 5 to his destination in order to not violate his latest arrival time. Note that a passenger may arrive at his destination time earlier than the latest arrival time, and the arc connecting the destination node to the latest arrival node is the destination ground arc, which is arc 8 in Figure 3.1. The destination ground arc makes it possible for an early arrived passenger to get to the latest arrival time destination node, and satisfy the passenger demand. In order to travel between hubs, a passenger may utilize the passenger flight arcs. There maybe multiple passenger flight arcs, this is possible if there are multiple hubs that are close to the origin or the destination of the passenger, or there are multiple flight times for the passenger to choose from at the hub. In Figure 3.1, there are two passenger flight arcs, arc 2 at 00:05, and arc 3 at 00:10. This may happen when the passenger has abundant time to take a later flight and still get to his destination before the latest arrival time. A passenger waiting in the hub is represented as a passenger ground arc, such as arc 6 in Figure 3.1. A passenger taking arc 6 means the passenger is unable to take the flight at arc 2, and has to wait for 5 minutes and take the next flight. This may happen if there is no available eVTOL at hub 1 at 00:05, or the eVTOLs available do not have enough room to accommodate the passenger. Finally, there is a taxi arc that connects the origin to the latest destination node, representing that the passenger cannot get to destination on time by taking an eVTOL, and has chosen to take a taxi to get to the destination at the latest arrival time. This is arc 9 in Figure 3.1.

For each eVTOL, there are also different types of arcs. For simplicity, we assume all eVTOLs start and end at the same location. The starting location is represented by a dummy node C1 for the beginning of the day, which we refer to as starting central node. Similarly, node C2 represent the finishing location at the end of the day, which we refer to as the ending central node. The dummy nodes are connected to all the hubs, and the arcs are dummy arcs, they are assumed to cost zero battery power and zero time to travel for the eVTOLs. This allows the model to help decide on the initial hub for the eVTOLs to start at, rather than us predefining the starting hub of each eVTOL and cause unnecessary flights for the eVTOLs. Given the number of eVTOLs in the network, denote it V_n, we can say there is a supply of V_n eVTOLs at node C1, and a demand of V_n eVTOLs at node C2. There are two types of arcs for the eVTOLs to travel. One of them is the eVTOL ground arcs. Taking the eVTOL ground arc meaning the eVTOL remains on the ground in the corresponding hub for the arc's time duration. This can be seen in Figure 3.1 in arcs 13,14, 6, 7,15,17. Note that the eVTOLs taking the ground arcs are being recharged for the duration it remains on the ground, and the number of eVTOLs taking the same ground arc is subject to the constraint of the capacity of the hub. The eVTOLs may travel between hubs by taking the flight arcs, which in this case are arcs 2, 3, 6. Note that the flight arcs may overlap with some passengers' flight arcs, so that the eVTOLs traveling the arcs may take the passengers traveling the same arc. The number of passengers allowed to travel the flight arc is decided by the number of eVTOLs traveling the arc, as well as the capacity of the eVTOLs. An eVTOL flight arc not overlapping with any passenger flight arc is a repositioning arc. This arcs exists to allow the eVTOLs to reposition themselves after a flight to better serve upcoming passengers. All eVTOLs start from C1, and flow through the network by traveling ground arcs and flight arcs, subject to the battery level constraints of the eVTOLs, and eventually returning to C2.

All the information required to define the network is assumed to be known in advance. The duration of any flight arcs is calculated by the distance between the departure hubs and arrival hub divided by the speed of the eVTOL. The duration of any arc connecting an origin or a destination to a hub is calculated by the Haversine [45] distance between the origin or destination, multiplied by a scalar to approximate the actual driving distance, then divided by the trip average speed. The taxi arcs are created using the passenger pickup time and drop off time.

Figure 3.1 only illustrates the structure of the time space network for a simple scenario of 1 passenger, passenger i. To build the complete time space network, we need to generate a similar network for each passenger. Note that, since the time component is discretize into intervals of 5 minutes, the departure and arrival times will be mapped to the closest discretized time point after the actual time. So if an eVTOL arrives at a hub at 12:01, it will be rounded to 12:05, which is the nearest discretized time point in the future. We start by generating the timelines for all hubs. Next, to generate all nodes and arcs for a passenger, we first generate the taxi arc using the data from the dataset, then we generate the origin to hubs arcs, where the end node of the arcs will be rounded to the nearest future discretized node. The eligible departure hubs are the hubs that are within 25 miles of the origin, and arrival hubs are eligible if they are within 25 miles of the destination. Then we generate the possible flight arcs for the passenger, which connect the hub nodes to the arrival hub nodes. Note that if a flight arc does not get the passenger to the destination within the time savings threshold, the flight arcs and the subsequent hub to destination arcs shall not be generated. After that, we generate the arrival hub to destination arcs, by connecting the destination node with the arrival hub node, where the time of the destination node can be computed with the arrival hub node information. We assume that each passenger's network takes the form of a bipartite graph, since the passengers are assumed to always travel from a hub that is close to the origin to a hub close to the destination, without making a connection flight in any of the hubs in between. Finally, we generate the ground arcs by connecting the consecutive nodes on the same location timelines. Last but not least, we need to generate some arcs to allow the eVTOLs to reposition themselves to better serve the next potential flight. These are generated right after every flight arc for a passenger, they start from the end node of the flight arc, and connect to the other hubs nodes, where the duration is the flight time it takes to go from the current hub to the other hub. Repositioning of eVTOLs is necessary, but can be an expensive operation. Flying an eVTOL in the air without transporting a passenger will only induce cost and not generate any revenue. Each eVTOL flight arc contains information of the battery consumption of the flight. The battery consumption is a linear function with respect to the flight time, and we assume the number of passengers on board does not affect the rate of battery consumption. For eVTOL ground arcs, the battery recharge amount information is also available, which similarly is a linear function with respect to time. This concludes the network building phase.

3.4.1 Parameters

V:	the set of all eVTOLs
O:	the set of all origins of passengers
D:	the set of all destinations of passengers
H:	the set of all hubs
P:	the set of all passengers
U:	The set of taxi arcs
T_0:	the beginning of the time horizon
T:	the end of the time horizon
N:	the set of all nodes, where each node is in the form of (h, t), where $h \in H \cup O \cup D$ and $t = T_0, \ldots, T$
A^{Pass}:	the set of passenger arcs
A^E:	the set of eVTOL arcs
F^{Pass}:	the set of passenger flight arcs, where $F^{Pass} \subset A^{Pass}$
F^E:	the set of eVTOL flight arcs, where $F^E \subset A^E$
G_i:	the set of all ground arc for hub $i \in H$
$S^n_{i,t}$:	the supply (+) or demand (-) or transit (0) of passenger n at hub i at time t
$(i, t, j, \bar{t})$:	arc that leaves hub $i \in N$ at time $t = T_0, \ldots, T$ and arrives at hub $j \in H$ at time $\bar{t} = T_0, \ldots, T, \bar{t} > t$
$In(i, t)$:	the set of incoming arcs of hub $i \in N$ at time $t = T_0, \ldots, T$
$Out(i, t)$:	the set of outgoing arcs of hub $i \in N$ at time $t = T_0, \ldots, T$
C_v:	the seat capacity of a eVTOL
C^H_i:	the parking capacity of hub i
B:	the max battery level of an eVTOL
$\hat{h}$:	the central hub at which all the eVTOLs start and end

3.4.2 Variables

$$x^v_i = \begin{cases} 1 & \text{if eVTOL } v \in V \text{ travels Arc } i \in A^E \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

$$y_i^p = \begin{cases} 1 & \text{if passenger } p \in P \text{ travels Arc } i \in A^{Pass} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

3.4.3 Model

$$\begin{aligned}
\min \quad & \sum_{p\in P}\sum_{i\in U} y_i^p && && (3.3)\\
\text{s.t.} \quad & -\sum_{j\in In(i,t)} y_j^p + \sum_{k\in Out(i,t)} y_k^p = S_{i,t}^p && \forall (i,t)\in N, \forall p \in P && (3.4)\\
& \sum_{i\in Out(\hat{h},T_0)} x_i^v = |V| && \forall v \in V && (3.5)\\
& \sum_{i\in Out(h,t)} x_i^v - \sum_{j\in In(h,t)} x_j^v = 0 && \forall h \in H, t\in T, \forall v \in V && (3.6)\\
& \sum_{i\in In(\hat{h},T)} x_i^v = |V| && \forall v \in V && (3.7)\\
& \sum_{v\in V} x_i^v \le C_j^H && \forall i \in G_j, \forall j \in H && (3.8)\\
& \sum_{p\in P} y_j^p \le \sum_{v\in V} C_v x_j^v && \forall j \in F^{Pass} \cap F^E && (3.9)\\
& x_i^v \in \{0,1\} && \forall a \in A^E && (3.10)\\
& y_i^p \in \{0,1\} && \forall a \in A^{pass} && (3.11)
\end{aligned}$$

1. (3.3): Minimize the number of passengers traveling by ground transportation; this is equivalent to maximizing the number of passengers transported by eVTOLs, since each passenger has to take either eVTOL or ground transportation.

2. (3.4): Passenger flow conservation constraints. The outflow and inflow of a passenger p at any given node should be equal to supply or demand of the node.

3. (3.5): All eVTOLs start at the same initial hub at the beginning of the time horizon.

4. (3.6): eVTOLs flow conservation constraints. The outflow and inflow of a eVTOLs at any given node should be equal.

5. (3.7): All eVTOLs finish at the same initial hub at the end of the time horizon.

6. (3.8): The number of eVTOLs parked at a hub cannot exceed the parking capacity of

the hub.

7. (3.9): The passengers cannot fly between 2 hubs unless there is enough eVTOL capacity to transport them.

8. (3.10): Decision variable of whether a eVTOL flies an arc has to be binary.

9. (3.11): Decision variable of wether a passenger travels an arc has to be binary.

3.4.4 Battery Constraints

Note that the mathematical model above does not incorporate the battery feasibility of eVTOLs. This needs to be considered because the remaining battery level for an eVTOL at any given point dictates if it can take the next flight or not. In this problem, we allow partial recharge, which means we do not force the eVTOL to recharge up to full battery every time it recharges. Instead, it may recharge up to any amount. This adds an extra layer of complexity to modeling the battery information, as the battery level at any given time depends on all the previous arcs that lead to the current node. Modeling the battery using a variable will introduce exponential number of variables to represent the current hub and time, as well as all the previous arcs, and also binary variables to indicate which previous arcs were selected. To avoid this complexity, we model the battery information by adding battery infeasibility constraints. To be more specific, for every possible subpath, in other words, every possible series of arcs, if an eVTOL with full battery takes the path and end up with negative battery level at any node on the subpath, we generate a constraint so the subpath is infeasible for any eVTOL. It is easy to observe that the number of battery infeasibility constraints is exponential, as there are exponential possible subpaths. To improve the performance of the model, we generate the constraints as lazy constraints. Lazy constraints are constraints that do not start in the model formulation, but are checked whenever an integer feasible solution is discovered. If any lazy constraints are violated, they are added to the model, and the model is solved again. This process continuous until a optimal solution which satisfies all the lazy constraints is found. The advantage of the lazy constraints is that only a subset of the constraints are added, so we can avoid adding exponential number of constraints into the model.

The battery infeasibility constraints is generated whenever a infeasible subpath is found in an integer feasible solution. For an integer feasible solution, it provides a path for each eVTOL in the network. A path is a set of consecutive arcs that start from the starting central node and ends with the ending central node. A subpath is a subset of the arcs of a path, where the arcs in the subset are connected through time. An infeasible subpath is defined to be a set of arcs on the eVTOL's path, where the arcs are between the last time the battery of the eVTOL is full before the battery level is negative, to the first time the battery level of the eVTOL is nonnegative again after the battery level is negative. This ensures that all the flight arcs that led to the battery level being negative are included in the infeasible subpath. A battery infeasible constraint is generated from the infeasible subpath by considering the minimal set of flight arcs of the infeasible subpath such that, if an eVTOL travels all the flight arcs, the battery level is negative, but if any flight arc is removed from the subpath, the battery level is feasible for the subpath. This is similar to the idea of a minimal cover cut, except instead of a cover we have a minimal battery infeasible subpath. A battery infeasible constraint is generated for every battery infeasible subpath of the solution, which means it is possible to have multiple battery infeasible constraints added to the model every time a integer feasible solution is found, since there might be multiple eVTOLs, and each path of eVTOL may have multiple infeasible subpaths. A typical battery constraint for a minimal infeasible subpath of an eVTOL i is of the following form

$$\sum_{i \in MIS} x_i^v \leq \text{card}(MIS) - 1 \qquad \forall i \in MIS, \tag{3.12}$$

where MIS is the set of flight arcs in the minimal infeasible subpath for eVTOL i.

3.5 Solution Methods

The problem is formulated as a mixed-integer-linear-program (MILP), and the battery constraints are added as lazy constraints. The problem is solved using Gurobi's mix integer program solver [56]. Two heuristics are proposed to improve the solution time of Gurobi.

3.5.1 Perturbed Dijkstra's Algorithm Heuristic

The complexity of the model largely attributes to the battery constraints. The battery constraints are exponential in number, and are added to the model as lazy constraints, which means they are checked every time a feasible integer solution of the model is found. If the solution is feasible to all the lazy constraints, then it is indeed a feasible integer solution of the full problem, otherwise the subset of lazy constraints that are violated are added to the model, and the model is resolved. The lazy constraints makes finding an integer feasible solution difficult, which increases the solution time of the model. We propose a heuristic, based on Dijkstra's Algorithm [92] for finding shortest paths in a network, which finds a feasible integer solution that is also battery feasible, and can be used as a good initial solution to the model.

The objective of the model is to find paths for eVTOLs and passengers such that as many passengers are transported using eVTOLs as possible. It can be observed that, if the eVTOL paths are fixed, the paths of the passengers can be found by solving a MILP. Finding the routing of any eVTOL is to find a path that starts from the starting central node of the network to the end central node of the network. Dijkstra's Algorithm is an efficient algorithm which determines the shortest path of any given pair of nodes in a network, where the distance of the arc is represented by the weight of the arc. In other words, the path should yield minimum total weight among all the paths connecting the pair of nodes. We need to maximize the passenger transported, as well as maintain battery feasiblility, so we propose a modified version of Dijkstra's Algorithm which finds a path that maintains battery feasibility, and aims to allow as many passengers covered as possible. The modified version of Dijkstra's Algorithm is described in Algorithm 1.

The modified version of Dijkstra's Algorithm is similar to Dijkstra's Algorithm. It uses a dictionary to store the weight of the nodes that is visited, in this case, the weight is the number of potential passengers that can travel through the arc, and traverse through the network in a breadth first fashion. The main differenmodified version of Dijkstraeen the modified version of Dijkstra's Algorithm and Dijkstra's Algorithm is that, instead of storing only the weight information, modified version of Dijkstra's Algorithm also stores the battery information of each visited node. And instead of updating the node's weight when a smaller weight path is discovered, we update the node only when the new path to the node is battery feasible, and has a larger weight than the stored weight of the node. This ensures that the path found by the algorithm satisfies the battery constraints, and has the best weight in the current iteration. However, since the modified version of Dijkstra's Algorithm needs to also consider battery information at each arc, it does not guarantee finding the best battery feasible path. The following example demonstrate such a scenario.

Algorithm 1: Perturbed Dijkstra's algorithm(*Graph*, *source*, *sink*, *source_battery*)

```
Create vertex set $Q$ sorted by time of vertex
for vertex $v$ in Graph do
  weight[$v$] $\leftarrow$ INFINITY
  battery[$v$] $\leftarrow$ source_battery
  prev[$v$] $\leftarrow$ UNDEFINED
  add $v$ to $Q$
end for
weight[$source$] $\leftarrow$ 0
while $Q$ is not empty do
  u $\leftarrow$ vertex in $Q$ with min time
  remove $u$ from $Q$
  for each neighbor $v$ of $u$ do
    neighbor_weight $\leftarrow$ weight[u] + weight($u$,$v$)
    neighbor_battery $\leftarrow$ min(battery[u] + battery($u$,$v$), full_battery)
    if neighbor_battery $> 0$ then
      if $v$ not visited then
        weight[v] $\leftarrow$ neighbor_weight
        battery[v] $\leftarrow$ neighbor_battery
        prev[v] $\leftarrow$ u
      else if neighbor_weight = weight[$v$] then
        if neighbor_battery $\geq$ battery[$v$] then
          battery[v] $\leftarrow$ neighbor_battery
          prev[v] $\leftarrow$ u
        end if
      else if neighbor_weight = weight[$v$] then
        weight[v] $\leftarrow$ neighbor_weight
        battery[v] $\leftarrow$ neighbor_battery
        prev[v] $\leftarrow$ u
      end if
    end if
  end for
end while
return weight[], prev[]
```

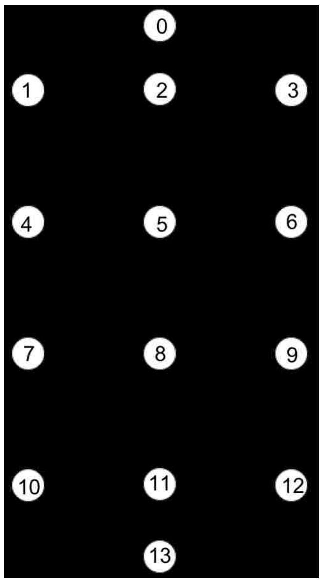


Figure 3.2: Example network where modified version of Dijkstra's algorithm gives suboptimal solution

Consider Figrue 3.2. There are 3 hubs in the graph, and nodes 1,4,7,10 are hub 1 nodes, 2,5,8,11 are hub 2 nodes, and 3,6,9,12 are hub 3 nodes. Suppose an eVTOL starts at the starting central node 0 with 100% battery, and needs to find a path that takes it to the ending central node 13 without violating the battery constraints. On each flight arc, namely (2,4), (4,8), (6,8), (8,10), (8,12), there is a pair of weights. The first weight is the number of passengers the arc can transport, and the second weight is the percentage of battery it will consume for the eVTOL to travel the arc. According to modified version of Dijkstra's algorithm, the algorithm will do a breath first search starting with node 0, and store at each node the current best path leading to the node, while updating the information only when the new path yields a better objective, which in this case is the number of passengers transported. So at node 8, it will record a path which is (0,2,4,8), with objective 2 and remaining battery at 60%. And when we search to 8 again from node 6, the new path (0,3,6,8) has an objective of 1 and remaining battery level at 70%. So the node will not update (0,3,6,8) as the best path to get to node 8. And this will lead to a solution of (0,2,4,8,11,13) with objective 2, since at node 8 it does not have enough battery to travel any other flight arc. However, clearly the optimal path is (0,3,6,8,12,13) with objective 4. This shows that the modified version of Dijkstra's algorithm does not necessarily yield the optimal solution, because the added battery constraints force the algorithm to balance transporting passengers and remaining battery feasibility, which makes it difficult to find the optimal solution.

One contribution of the algorithm is to find a good quality initial solution for the model, since it is difficult for the solver to find a initial solution that satisfies all the battery constraints.

The modified version of Dijkstra's Algorithm can also be used to fix a battery infeasible solution to become battery feasible, thus providing an incumbent solution to the solver, and potentially improve the global upper bound in the branch and bound process.

3.5.2 Feasible solution repair heuristic

It can be observed that the incumbent solutions found during the branch and bound process may be cut off by the battery constraints, which makes it difficult to find incumbent solutions which in turn increases the solution time. Many of the feasible integer solutions only violate few batterybattery constraints, meaning a small segment of the eVTOL route is infeasible. We can modify the integer feasible solution by replacing the battery infeasible subpath with a new subpath found by the modified version of Dijkstra's Algorithm, such that the new route is battery feasible, and thus yield an incumbent solution.

For a given integer feasible solution, we can identify the path for each eVTOL by looking at the values of the variables. For each eVTOL path, there may exists some subpath which is battery infeasible, we name such a subpath the battery infeasible subpath. The goal is to replace the battery infeasible subpath with a new subpath, starting and ending at the same nodes but being battery feasible, such that a integer feasible solution is constructed. To find the replacing battery feasible subpath, the starting node and the ending node of the battery infeasible subpath is determined. The starting node is defined to be the last node where the battery is fully charged before the flight that resulted in battery infeasibility. The ending node is defined to be the first node where the battery is full after the battery infeasibility. The new subpath can be generated by specifying the starting node, the remaining battery at the starting node, and the ending node for the modified version of Dijkstra's Algorithm. By replacing each battery infeasible subpath in the current integer feasible solution, we get an incumbent solution that is battery feasible.

The feasible solution repair heuristic modifies a battery infeasible integer feasible solution into a battery feasible integer solution, but the objective of the modified solution may not improve on the original integer feasible solution. In most cases, it results in a worse objective, because the battery infeasibility is resolved by taking fewer flight arcs. But, the incumbent solution it provides may still improve on the current global upper bound and leads to better performance, which can be seen is often the case in practice.

3.6 Numerical Result

To test the overall performance of the heuristic, experiments are conducted on the model with different sets of parameters. The passenger information used in the experiments are the 2016 Yellow Taxi Trip Data `https://data.cityofnewyork.us/dataset/2016-Yellow-Taxi-Trip-Data/k67s-dv2t`. The dataset includes trip records from all trips completed in yellow taxis from in NYC from January to June in 2016, where the pick-up and drop-off time, pick-up and drop-off location, and trip distance for each trip are provided. Figure 3.3 gives a visualization of a subset of the trips data with 2500 passengers, and figure 3.4 provides a potential locations of the 5 hubs in greater New York area. We consider that each trip is consist of 1 passenger, and we select the trips for which the trip distance is between 20 miles to 120 miles. The reason is the target customers for the eVTOL transportation service is for long distance traveling passengers, and 120 miles is the range of a typical eVTOL. Due to the large number of parameters associated with the model, we primarily investigate the performance of the heuristic under 2 parameters, namely, the number of eVTOLS, and the number of passengers. All the other parameters will be constant. There are also some implicit assumptions that further simplify the experiments:

1. All passenger may take no more than 1 flight, i.e. no passenger should be taking connecting flights in the eVTOL network.

2. The time horizon is divided into equal-sized time bands of 5 minutes.

3. All eVTOLs start from the same hub at the beginning of the time horizon, and end at the same hub at the end of the time horizon.

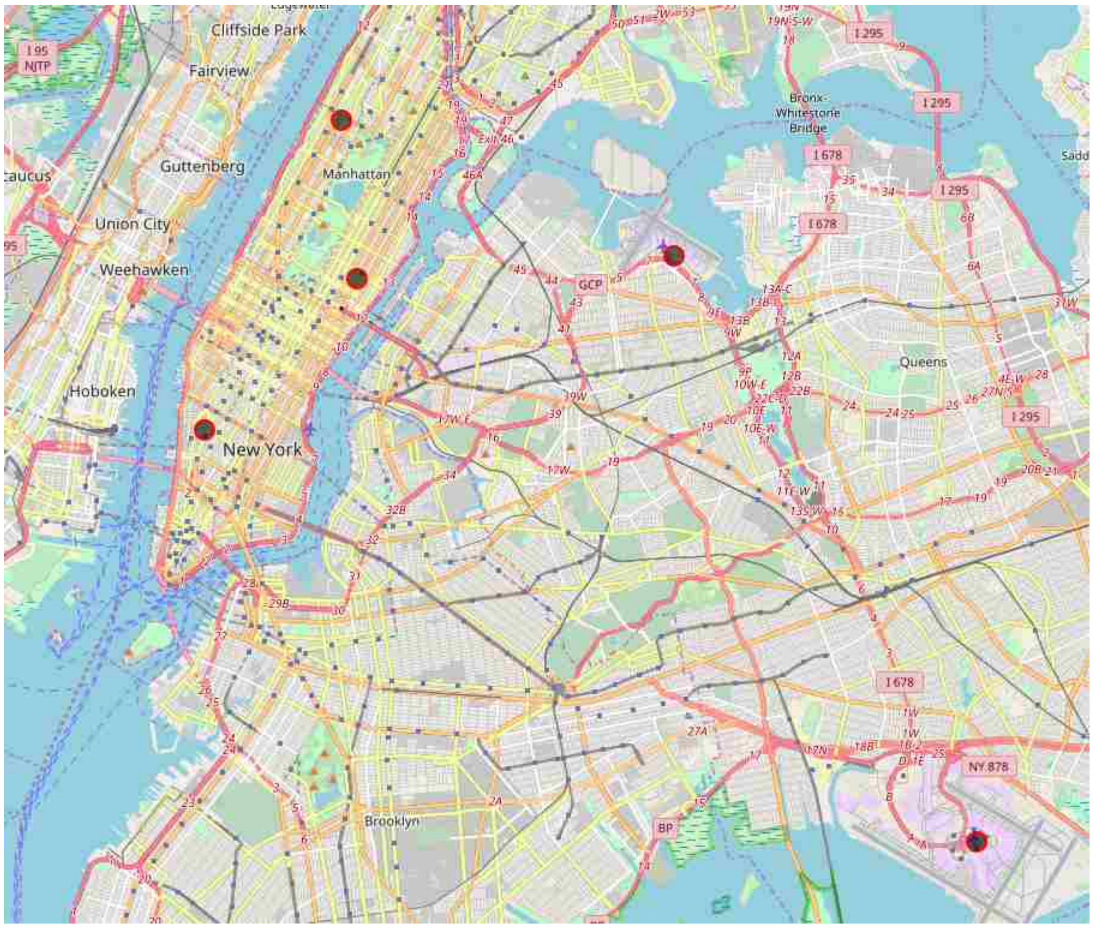


Figure 3.3: Passengers pickup (blue) and dropoff (red) locations in Greater New York area from NYC taxi trip dataset.

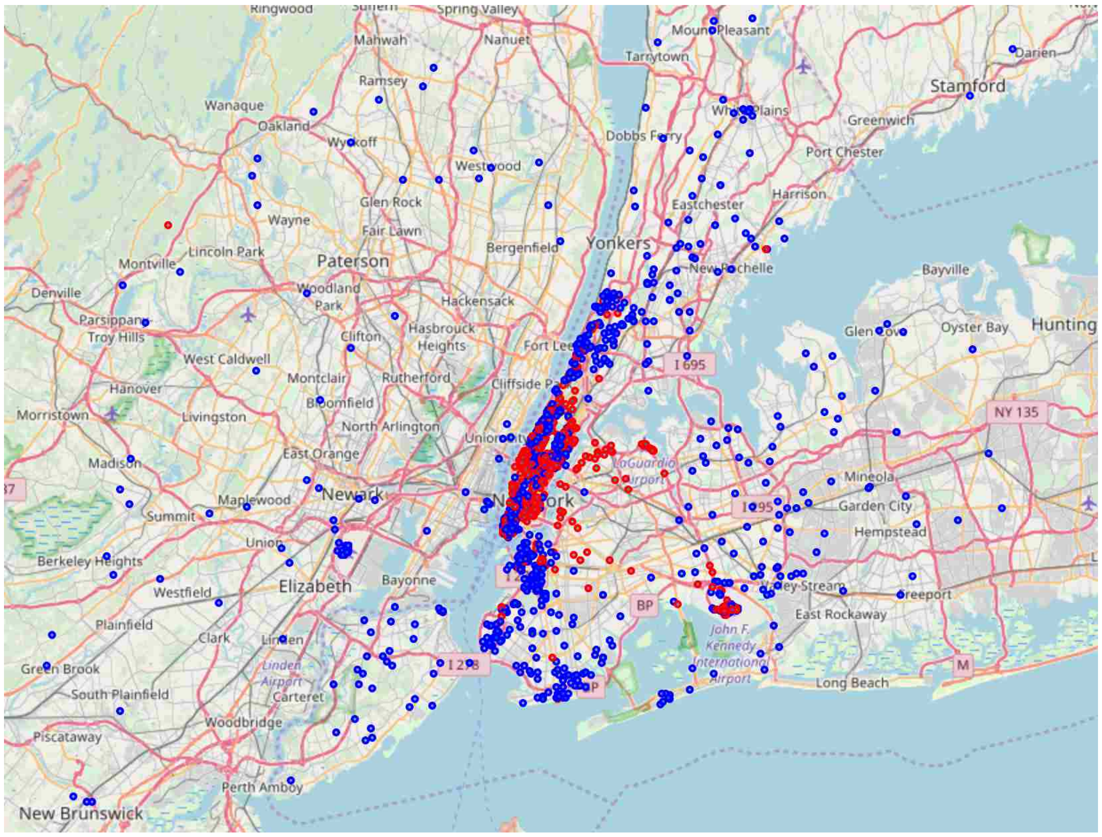


Figure 3.4: Hub locations in Greater New York area generated from NYC taxi trip dataset.

We generated 3 sets of instances, each has 216, 268, and 398 passengers. Each set includes 5 instances, namely with 1,2,3,4,5 eVTOLs available for transportation in the network. We use Gurobi's MILP solver, and the heuristics proposed were implemented in Python and used in conjunction with Gurobi's solver. The battery feasibility constraints are implemented as lazy cuts in Gurobi's MIPSOL callbacks. The feasible solution repair heuristic is implemented in Gurobi's MIPNODE callbacks. The time limit is set to 900 seconds, and the optimality gap tolerance is set to 3%. The tests are conducted on a Mac machine with 2.9GHz Intel Core i5 processor and 8GB memory.

We first look at the solution time it takes for the heuristic to generate an initial solution. We compare the solution time and quality of the initial solution found by the modified version of Dijkstra's Algorithm heuristic versus Gurobi's first incumbent feasible solution.

		Gurobi		Gurobi with modified version of Dijkstra algorithm heuristic	
Passenger #	eVTOL #	Initial Solution Gap	CPU Time	Initial Solution Gap	CPU Time
773	1	1.17	6.92	2.02	0.02
	2	1.32	386	13.7	0.03
	3	-	1000	21.1	0.04
	4	-	1000	24.00	0.05
	5	-	1000	26.1	0.07
994	1	1.94	29.97	3.23	0.01
	2	-	1000	13.2	0.03
	3	-	1000	22.9	0.05
	4	-	1000	20.9	0.07
	5	0.00	61.36	14.7	0.15
1362	1	-	1000	3.91	0.02
	2	-	900	14.70	0.05
	3	-	900	23.70	0.07
	4	-	900	22.00	0.10
	5	0.17	282.07	17.00	0.11

Table 3.1: Comparison of initial solution gap and CPU time between GuRoBi and Dijkstra's Algorithm Heuristic ("-" indicates no initial solution is found)

3.6.1 Initial Solution Generation

Table 3.1 displays the information of initial solutions found by Gurobi and the modified version of Dijkstra algorithm. The initial solution gap is the relative gap between the objective of the initial solution and the current best lower bound, and the CPU time is the time in seconds when the initial solution is found. We can observe that for smaller instances, Gurobi is good at finding a good initial solution. In most cases, it finds the optimal solution within the time limits. However, it does need some time to find the solution. On the other hand, the modified version of Dijkstra's algorithm finds initial solutions that have relatively larger gap, but it takes very lettle time to get to an initial solution with good gap. This is especially true for larger instances, where Gurobi fails to find an initial solution within the time limit, but the heuristic is able to provide good quality initial solutions in less than a second. This is useful when a good quality solution is needed in a short time. The solution obtained by the modified version of Dijkstra's algorithm can be used as a starting solution for Gurobi's Branch and Bound approach, which significantly improves the time to find a initial integer feasible solution that is battery feasible.

Next, we compare the solution time with Gurobi and the solution time with Gurobi and the modified version of Dijkstra's Algorithm heuristic. We set the time limit of the experiments to be 1000 seconds, this is because in practice a solution is required in a short period of time. The optimal relative tolerance is set to 2.5%.

3.6.2 Solution Time

	Gurobi		Gurobi with heuristic	
eVTOL	Gap (%)	CPU Time (s)	Gap (%)	CPU Time (s)
1	2.45	5.00	2.11	0.62
2	6.86	1000.00	5.56	1000.00
3	8.14	1000.00	4.99	1000.00
4	4.47	1000.00	4.58	1000.00
5	2.85	436.26	2.84	301.02

Table 3.2: Instance with 773 passengers

	Gurobi		Gurobi with heuristic	
eVTOL	Gap (%)	CPU Time (s)	Gap (%)	CPU Time (s)
1	2.30	5.45	2.98	3.19
2	6.97	1000.00	6.15	1000.00
3	26.40	1000.00	7.32	1000.00
4	42.88	1000.00	35.79	1000.00
5	1.94	29.97	0.66	38.62

Table 3.3: Instance with 994 passengers

	Gurobi		Gurobi with heuristic	
eVTOL	Gap (%)	CPU Time (s)	Gap (%)	CPU Time (s)
1	2.43	491.61	2.54	32.01
2	41.50	1000.00	8.27	1000.00
3	52.97	1000.00	11.61	1000.00
4	56.62	1000.00	6.46	1000.00
5	2.99	196.26	1.02	93.94

Table 3.4: Instance with 1362 passengers

Table 3.2 summarizes the solution information for the instances with 216 passengers. It can be observed that all instances are solved to optimality within the time limit, with instances of number of eVTOLs between 2 to 5 solved within 5 seconds for both approaches. For all instances, the Gurobi with heuristic approach has a better solution time, suggesting using the heuristic improves the efficiency of the solver.

Table 3.3 summarizes the solution information for the instances with 268 passengers. Gurobi was able to solve 2 out of the 5 instances within the time limit, while Gurobi with heuristic was able to solve 3 out of 5 instances. For the instances solved, Gurobi with heuristic has better solution time. And the for instances that are not solved within the time limit, Gurobi with heuristic has a gap that is at least as good as Gurobi's.

Table 3.3 summarizes the solution information for the instances with 398 passengers. Gurobi with heuristic was able to solve the instance with 5 eVTOLs, but Gurobi was not able to solve any of the instances. For the instances that are not solved within time limit, Gurobi with heuristic has a smaller gap when the time limit is reached, suggesting the heuristic improves the global upper bound by finding better incumbent solutions.

3.7 Conclusion

We proposed a multi-commodity network flow model for maximizing the throughput for an on-demand aviation transportation network, where the primary means of transportation are eVTOLs. The model helps make strategic decisions for the network by routing passengers and eVTOLs, while maximizing the number of passengers transported. The battery limitations of the eVTOLs are considered in the model as resource constraints, and are implemented as lazy constraints in order to avoid adding exponentially many constraints to the model. A heuristic is proposed based on Dijkstra's algorithm which generates good quality initial solutions in short time, and the heuristic is also used to reconstruct battery feasible incumbent solutions from battery infeasible solutions that violated the battery constraints. According to the experiments, the heuristic improves the efficiency of the solver by finding better incumbent solutions, thus improving the global upper bound. It may also be used to find good quality solutions when an optimal solution is not required.

Chapter 4

Solving Copositive Programs via Semi-infinite Programming approach

4.1 Introduction

A copositive program is a conic optimization problem over the cone of the copositive matrices. Specifically, given $b \in \mathbb{R}^m$, $A_i \in \mathcal{S}^n$ for $i = 1, \dots, m$, $C \in \mathcal{S}^n$, where $\mathcal{S}^n$ is the set of symmetric matrices, a copositive program is the problem

$$\begin{array}{rll} d^{COP} = & \max & b^\intercal y \\ & \text{s.t.} & S = C - \sum\limits_{i=1}^{m} y_i A_i \\ & & S \in \mathcal{COP}^n, y \in \mathbb{R}^m, \end{array} \tag{COP}$$

where $\mathcal{COP}^n$ is the copositive cone [10], which is defined as

$$\mathcal{COP}^n = \{X \in \mathcal{S}^n : u^T X u \geq 0 \text{ for all } u \in \mathbb{R}^n_+\}, \tag{4.1}$$

where $\mathcal{S}^n$ denotes the set of symmetric matrices in $\mathbb{R}^{n \times n}$. The dual cone of a given cone $\mathcal{K} \subseteq \mathcal{S}$ is denoted as $\mathcal{K}^*$, which is defined as

$$\mathcal{K}^* = \{A \in \mathcal{S} : \langle A, B \rangle \geq 0 \text{ for all } B \in \mathcal{K}\}.$$

Each copositive program has an associated dual problem, namely the completely positive program, which is defined as

$$\begin{aligned} p^{CP} = \quad & \min \quad \langle C, X\rangle \\ & \text{s.t.} \quad \langle A_i, X\rangle = b_i, \quad i = 1, \dots, m \\ & \qquad\; X \in \mathcal{CP}^n, \end{aligned} \tag{CP}$$

where $\mathcal{CP}^n$ is the dual cone of the cone of copositve matrices, namely the completely positive cone [10], and it is defined as

$$\mathcal{CP}^n = \left\{ X \in \mathcal{S}^n : X = \sum_{i=1}^{m} u_i u_i^\intercal, u_i \in \mathbb{R}^n_+, i = 1, \dots, m \right\}. \tag{4.2}$$

From the definition we can see that any completely positive matrix can be written as a finite sum of rank 1 completely positive matrices, and this representation is called the rank 1 representation of a completely positive matrix [10]. Both $\mathcal{COP}^n$ and $\mathcal{CP}^n$ are closed, convex, pointed, full dimensional, nonpolyhedral cones. A large body of work has been done on the properties and characteristics of copositive matrices as well as completely positive matrices, for a comprehensive survey, we refer our readers to [59], [37],[10].

The theory of copositive programming and completely positive programming are closely related to the field of combinatorial and quadratic optimization problems, as they provide convex reformulations for problems that arise from these fields. It has been shown in [18] that the problem of maximizing a quadratic form over the simplex can be reformulated as an equivalent copositive program. Burer showed in [29] that any quadratic program with a mix of binary and continuous variables has an equivalent completely positive program reformulation. More recent advancements on the topic of copositive programs and completely positive programs can be found in [43],[30] and [17].

In this paper, we will focus on a numerical solution method for general copositive program. It is well known that copositive programs are NP-hard [75], despite the fact that they are convex optimization problems. Many approximation hierarchy has been proposed for the cone of copositive matrices and successfully used in the literature for solving copositive programs. One line of approximation scheme is based on the definition of copositivity in terms of quadratic forms. A matrix $A \in \mathcal{S}^n$ is copositive if and only if $P_A(x) \geq 0$ for all

$x \in \mathbb{R}^n$ where

$$P_A(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i^2 x_j^2.$$

Using the sufficient condition that $P_A(x)$ is nonnegative if it admits a sum of squares (SOS) representation, where a form $F(x)$ of degree $2m$ is a sum of squares form if and only if there exists forms $f_1(x), \ldots, f_k(x)$ of degree m such that $F(x) = \sum_{i=1}^{k} f_i(x)^2$. Parrilo [79] constructed the following hierarchy of cones for $r \in \mathbb{N}$:

$$\mathcal{K}_n^r = \left\{ A \in \mathcal{S}^n : P_A(x) \left(\sum_{i=1}^{n} x_i^2 \right)^r \text{ is an sos polynomial} \right\}.$$

Parrilo showed that the hierarchy of cones satisfies $\mathcal{S}_+^n + \mathcal{N} = \mathcal{K}_n^0 \subset \mathcal{K}_n^1 \subset \cdots \subset \mathcal{COP}^n$, and $\text{int}(\mathcal{COP}^n) \subseteq \cup_{r \in \mathbb{N}} \mathcal{K}_n^r$, where $\mathcal{S}_+^n$ is the set of positive semidefinite matrices in $\mathbb{R}^{n \times n}$ and $\mathcal{N}$ is the set of element-wise nonegative matrices, so the hierarchy of cones approximate the copositive cone from within. Since each $\mathcal{K}_n^r$ can be represented as a system of linear matrix inequalities (LMIs), optimizing over $\mathcal{K}_n^r$ amounts to solving a semidefinite program.

A weaker sufficient condition for nonnegativity of the polynomial is used in [35] to construct a hierarchy of approximation of the copositive cone. The hierarchy of approximations is defined as

$$\mathcal{C}_n^r = \left\{ A \in \mathcal{S}^n : P_A(x) \left(\sum_{i=1}^{n} x_i^2 \right)^r \text{ has nonnegative coefficients} \right\}.$$

It is shown in [35] that $\mathcal{N} = \mathcal{C}_n^0 \subset \mathcal{C}_n^1 \subset \cdots \subset \mathcal{COP}^n$, and $\text{int}(\mathcal{COP}^n) \subseteq \cup_{r \in \mathbb{N}} \mathcal{C}_n^r$, and optimizing over each of the polyhedral cone amounts to solving a Linear Program (LP).

Taking a different approach, Yıldırım [2] proposed another hierarchy of outer polyhedral approximations to the copositive cone. The hierarchy of approximations are generated by systematically sampling points from the standard simplex. For $r \in \mathbb{N}$, a regular grid of rational points on the standard simplex is defined as

$$\Delta(n, r) = \{x \in \Delta^n : (r+2)x \in \mathbb{N}^n\},$$

and an outer approximation associated with r is defined as

$$\mathcal{O}_n^r = \{X \in \mathcal{S}^n : d^\mathsf{T} X d \geq 0 \text{ for all } d \in \delta(n,r)\}, \quad r = 0, 1, 2, \ldots,$$

where $\delta(n,r) = \cup_{k=0}^r \Delta(n,k)$. Yıldırım showed that $\mathcal{O}_n^0 \supseteq \mathcal{O}_n^1 \supseteq \cdots \supseteq \mathcal{COP}^n$, and in the limit the polyhedral cones $\mathcal{O}_n^r$ provides a hierarchy of outer approximations that converges to the cone of copositive matrices.

Another line of research focuses on approximating the copositive cone using both inner and outer approximations. Bundfuss and Dür proposed two hierarchies of cones that provides an inner and outer approximations for the copositive cone respectively; which relies on the concept of simplicial partitions. According to [22], a simplicial partition of a simplex Δ is a family $\mathcal{P} = \{\Delta^1, \ldots, \Delta^m\}$ of simplices satisfying

$$\Delta = \bigcup_{i=1}^m \Delta^i \text{ and } \operatorname{int} \Delta^i \cap \operatorname{int} \Delta^j = \emptyset \text{ for } i \neq j.$$

Given a simplicial partition $\mathcal{P}$ of the standard simplex Δ^S, the inner approximation is defined as

$$\mathcal{I}_\mathcal{P} = \{A \in \mathcal{S}^n : v^T A v \geq 0 \text{ for all } v \in V_\mathcal{P}, u^T A v \geq 0 \text{ for all } \{u,v\} \in E_\mathcal{P}\},$$

and the outer approximation is defined as

$$\mathcal{O}_\mathcal{P} = \{A \in \mathcal{S}^n : v^T A v \geq 0 \text{ for all } v \in V_\mathcal{P}\},$$

where $V_\mathcal{P}$ is the set of all vertices of the simplicial partition $\mathcal{P}$, and $E_\mathcal{P}$ is the set of all edges of $\mathcal{P}$. Both approximations are polyhedral, so optimizing over a cone of either the inner or outer approximation is equivalent to solving a LP. Bundfuss and Dür's approximation has the advantage that it does not approximate the copositive cone uniformly. Instead, the approximations are guided by the objective function of the optimization problem such that only the relevant part of the copositive cone is being approximated with high accuracy, thus saving much computational effort.

More recently, an inner approximation scheme for completely positive cone is proposed in [21] using the cone of nonnegative scaled diagonally dominant matrices (SDD). Using the projections of this cone, both uniform and problem-dependent approximation hierarchies can be constructed. Optimizing over this approximation amounts to solving a second-order-cone program, which offers a compromise between semidefinite programing and linear programing based approaches.

In this work, we are concerned with the connection between linear semi-infinite programing (LSIP) theories and copositive programming. A linear semi-infinite program is an optimization problem with linear objective function and linear constraints in which either the number of unknowns or the number of constraints is infinite, but not both. LSIP can be seen as an extension of linear programming, or as a particular branch of both semi-infinite programming and infinite dimensional programming. LSIP has many direct applications, including but not limited to pattern recognition, environmental policies and industrial process. For a comprehensive survey on the applications of LISP, we refer our readers to [51]. One of the main difference between LP and LSIP is that the strong duality result of LP is no longer valid. A lot of work on the theory of linear semi-infinite systems (LSIS) have been developed to provide fundamentals for the LSIP theory. The main topics of the LSIP theory are optimality conditions, duality, the characterization of the extreme points and extreme directions of the primal and dual feasible sets. Several classes of LSISs are essential to the geometry, optimality and duality theory of LSIP. A linear semi-infinite system $\sigma = \{a_t^\intercal x \geq b_t, t \in T\}$, where T is an infinite set, is Farkas-Minkowski (FM) [50] when every linear consequent relation of σ is also the consequence of a finite subsystem of σ. A LSIS σ is locally polyhedral (LOP) [3] if $D(F, x) = A(x)^0$ for all $x \in F$, where $F = \{x : a_t^\intercal x \geq b_t, t \in T\}$, $A(x)^0$ is the positive polar of the active cone at x, and $D(F, x)$ is the cone of feasible direction at x. Another class of LSIS, namely the locally Farkas Minkowski (LFFM) LSIS, is introduced in [83], where σ is LFM if every linear consequent relation of σ determining a supporting hyperplane to F is also the consequence of a finite subsystem of σ. All these special classes of LSIS have nice properties and play an important role in the theory off LSIP. A detailed study of these classes of LSIS and their properties can be found in [52], [51] and [53]. In [85], a detailed survey on different numerical method for solving LSIP is provided.

In this chapter, we propose a cutting-plane algorithm for solving copositive programs. More specifically, we reformulate the copositive program as an equivalent linear semi-infinite program, which is then solved using a cutting-plane algorithm. The cutting-plane algorithm involves a pair of master problem and subproblem, where the master problem, which is an

LP, generates solutions feasible to the current set of cuts, and the subproblem, which is a standard quadratic program, generates the most violated cut with respect to the current solution. Our approach exploit the efficiency of the solver in [95] on SQPs to generate strong inequalities which improves the tightness of the bounds obtained from the master problem. The preliminary experiments are conducted on a set of copositive programs obtained from reformulating the QPs.

4.2 Semi-infinite Solution Approach

We assume that (COP) has a strictly feasible solution, so that strong duality holds for (CP) and (COP).

It can be observed that completely positive matrices may also be represented over the standard simplex $\Delta_n^S = \{x \in \mathbb{R}_+^n : \|x\|_1 = 1\}$, we provide the following lemma.

Lemma 4.2.1. *A matrix X is completely positive if and only if $X = \sum_{i=1}^{m} \lambda_i u_i u_i^\intercal$ where $u_i \in \Delta^S$, $\lambda_i \geq 0$ for $i = 1, \ldots, m$, and $m \in \mathbb{N}$ is finite.*

Proof. Take $X \in \mathcal{CP}^n$, then we have

$$X = \sum_{i=1}^{m} u_i u_i^\intercal = \sum_{i=1}^{m} \|u_i\|_1^2 \left(\frac{u_1}{\|u_i\|_1}\right)\left(\frac{u_1}{\|u_i\|_1}\right)^T = \sum_{i=1}^{m} \lambda_i v_i v_i^\intercal,$$

where $\lambda_i = \|u_i\|_1^2 \geq 0$ and $v_i = \frac{u_1}{\|u_i\|_1} \in \Delta^S$ since $u_i \geq 0$ for $i = 1, \ldots, m$.

Next we prove the other direction. Let $X = \sum_{i=1}^{m} \lambda_i u_i u_i^\intercal$ where $u_i \in \Delta_n^S$ and $\lambda_i \geq 0$ for $i = 1, \ldots, m$. Let $v_i = \sqrt{\lambda_i} u_i \geq 0$, then

$$X = \sum_{i=1}^{m} \lambda_i u_i u_i^\intercal = \sum_{i=1}^{m} (\sqrt{\lambda_i} u_i)(\sqrt{\lambda_i} u_i)^\intercal = \sum_{i=1}^{m} v_i v_i^\intercal, v_i \geq 0.$$

Therefore X is completely positive. □

Similarly, a condition for copositivity is proposed in [22], where a matrix $A \in \mathcal{S}^n$ is copositive if and only if $x^T A x \geq 0$ for all $x \in \Delta^S$, where $\Delta_n^S = \{x \in \mathbb{R}_+^n : \|x\|_1 = 1\}$ is the standard simplex of dimension n.

Using this observation, one can rewrite (CP) as the following semi-infinite program

$$\begin{aligned} p^{\Delta} = \quad \min \quad & \sum_{u\in\Delta^n} \lambda_u \langle C, uu^{\intercal}\rangle \\ \text{s.t.} \quad & \sum_{u\in\Delta^n} \lambda_u \langle A_i, uu^{\intercal}\rangle = b_i, \quad i = 1, \dots, m \\ & \lambda_u \geq 0 \quad u \in \Delta^n, \end{aligned} \tag{pLP$_{\Delta^n}$}$$

and (COP) as the following semi-infinite program

$$\begin{aligned} d^{\Delta} = \quad \max \quad & b^{\intercal} y \\ \text{s.t.} \quad & \left\langle C - \sum_{i=1}^m y_i A_i, uu^T \right\rangle \geq 0 \qquad \forall u \in \Delta^n \\ & y \in \mathbb{R}^m. \end{aligned} \tag{dLP$_{\Delta^n}$}$$

The problem (dLP$_{\Delta^n}$) has finitely many variables and infinitely many constraints, whereas the (Haar's) dual [53] problem (pLP$_{\Delta^n}$) has finitely many constraints and infinitely many variables. By definition, only a finite number of variables of (pLP$_{\Delta^n}$) can take on a non-zero value. This follows from the fact that a completely positive matrix can be written as a finite sum of rank 1 matrices according to Lemma 4.2.1.

Instead of all $u \in \Delta^n$, we consider a finite subset of $u \in U \subset \Delta^n$, which results in a LP relaxation of the primal problem

$$\begin{aligned} p^{U} = \quad \min \quad & \sum_{u\in\Delta^n} \lambda_u \langle C, uu^{\intercal}\rangle \\ \text{s.t.} \quad & \sum_{u\in U} \lambda_u \langle A_i, uu^{\intercal}\rangle = b_i, \quad i = 1, \dots, m \\ & \lambda_u \geq 0 \quad u \in U, \end{aligned} \tag{pLP$_U$}$$

and its corresponding dual LP problem is

$$\begin{aligned} d^{U} = \quad \max \quad & b^{\intercal} y \\ \text{s.t.} \quad & \left\langle C - \sum_{i=1}^m y_i A_i, uu^T \right\rangle \geq 0 \qquad \forall u \in U \\ & y \in \mathbb{R}^m. \end{aligned} \tag{dLP$_U$}$$

It can be observed that (dLP$_U$) is a relaxation of (dLP$_{\Delta^n}$), as it only contains a finite subset of the constraints of (dLP$_{\Delta^n}$). Since (dLP$_U$) and (pLP$_U$) are both LPs, and they are dual of each other, by strong duality we have that $p^U = d^U$. So we have the following lemma

Lemma 4.2.2. *For any finite $U \subset \Delta^n$, it follows that*

$$p^U = d^U \geq d^\Delta = d^{COP}.$$

The strength of the relaxation (dLP$_U$) depends largely on the subset U. A well chosen U may provide a a good approximation for the copositive cone $\mathcal{COP}^n$, yielding a tight upper bound for the optimal objective value of (COP). In the next section, we propose an algorithmic way of selecting $u \in \Delta^n$ iteratively such that a good upper bound for (COP) is obtained. Moreover, the subproblem solved during the algorithm can be used to produce a lower bound and a performance guarantee for upper bound.

The theory of duality, and optimality conditions like Karush-Kuhn-Tucker (KKT) for ordinary nonlinear programming (NLP) can be naturally extended to LSIP problems, under some conditions. We provide such a condition below from [53].

Theorem 4.2.3 (Thm. 2 [53]). *For a pair of problems*

$$\begin{aligned} \inf \quad & c^\intercal x \\ \text{s.t.} \quad & a_t^\intercal x \geq b_t, \quad \forall t \in T, \end{aligned} \tag{PLSIP}$$

and

$$\begin{aligned} \sup \quad & \sum_{t \in T} \lambda_t b_t \\ \text{s.t.} \quad & \sum_{t \in T} \lambda_t a_t = c, \quad \lambda \in \mathbb{R}_+^{(T)}, \end{aligned} \tag{DLSIP}$$

where $c \in \mathbb{R}^n$, $a \in \mathbb{R}^n$,$b \in \mathbb{R}$, T is an arbitrary set that allows only a finite number of associated variables $\lambda_t, t \in T$, to be non-zero. The notation $\mathbb{R}_+^{(T)}$ denotes the positive cone of the linear space of λ. Let $\sigma = \{a_t^\intercal x \geq b_t, t \in T\}$, and denote F and F^* (Λ and Λ^*) be the feasible set and optimal set of (PLSIP) *(of* (DLSIP)*, respectively).*

If σ is locally Farkas-Minkowski (LFM) and $\bar{x} \in F$, then the following statements are equivalent to each other:

1. *$\bar{x} \in F^*$;*

2. *there exists $\bar{\lambda} \in \Lambda$ such that $\bar{\lambda}_t(a_t^\intercal \bar{x} - b_t) = 0$ for all $t \in T$; and*

3. *there exists* $\bar{\lambda} \in \mathbb{R}_+^{(T)}$ *such that* $L(\bar{x}, \lambda) \leq L(\bar{x}, \bar{\lambda}) \leq L(x, \bar{\lambda})$ *for all* $x \in \mathbb{R}^n$ *and for all* $\lambda \in \mathbb{R}_+^{(T)}$*, where* $L(x, \lambda)$ *is the Lagrangian function associated with* (PLSIP)*; i.e.,* $L(x, \lambda) = c^\intercal x + \sum_{t \in T} \lambda_t (b_t - a_t^\intercal x)$.

The following two results characterize when a set $\sigma = \{a_t^\intercal x \geq b_t, t \in T\}$ is LFM.

Proposition 5 (Cor. 3.1.1 in [50] and Thm. 3.2 in [83])**.** *Let* $\{a_t^\intercal x \geq \beta_t, t \in T\}$ *be a consistent system. If* $\operatorname{cone}([a_t^\intercal \quad \beta_t^\intercal]^\intercal, t \in T\})$ *is closed, then it is a LFM system.*

By Theorem 4.2.3 above, for the KKT conditions to hold, one needs to show the system of infinitely linear constraints is FM. In the following lemma, we show that the problem (dLP$_{\Delta^n}$) is a FM system, thus KKT conditions are valid for the pair of problems (pLP$_{\Delta^n}$) and (dLP$_{\Delta^n}$).

Lemma 4.2.4. *Let* σ *be a linear semi-infinite system where*

$$\sigma = \left\{ \left\langle C - \sum_{i=1}^{m} y_i A_i, uu^\intercal \right\rangle \geq 0, u \in \Delta^n \right\},$$

and

$$A_\sigma = \{[-u^\intercal A_1 u, \ldots, -u^\intercal A_m u, -u^\intercal C u]^\intercal : u \in \Delta^n\}.$$

Then σ *is a Farkas-Minkowski system if* $0 \notin \operatorname{conv}(A_\sigma)$.

Proof. We can rewrite the system as

$$\sigma = \{a_u^\intercal y \geq \beta_u, u \in \Delta^n\}.$$

where

$$a_u = [-u^\intercal A_1 u, \ldots, -u^\intercal A_m u]^\intercal \text{ and } \beta_u = -u^\intercal C u.$$

According to Theorem 5, if $K_\sigma = \operatorname{cone}(A_\sigma)$ is closed, then σ is a Farkas-Minkowski system. We show that if $0 \notin \operatorname{conv}(A_\sigma)$, then K_σ is closed. Since Δ^n is compact, and the quadratic form $Q_A(u) = u^\intercal A u$ is continuous on Δ^n, then A_σ is compact, as the continuous image of a compact set is compact (see, sec. 10.2, Thm 5 in [66]). By Proposition 1.3.2 in [13], we have that $\operatorname{conv}(A_\sigma)$ is convex and compact. Then by Corollary 9.6.1 in [87], and the fact that $0 \notin \mathrm{conv}(A_\sigma)$, we have that $K_\sigma = \mathrm{cone}(A_\sigma)$ is closed, as the cone generated by a compact set which does not contain the origin is closed. Therefore σ is a Farkas-Minkowski system. $\square$

4.3 Valid Inequalities for COP

From Lemma 1 we know one can obtain an upper bound of (COP) by solving a relaxation problem (dLP$_U$) where only a finite subset of the constraints induced by Δ is included. The feasible set of the set of constraints induced by $U \subset \Delta$, denoted by $\mathcal{O}_U$, forms an outer approximation of the cone $\mathcal{COP}^n$

$$\mathcal{O}_U = \{A \in \mathcal{S} : u^T Au \geq 0 \text{ for all } u \in U \subseteq \Delta\} \supseteq \{A \in \mathcal{S} : u^T Au \geq 0 \text{ for all } u \in \Delta\} = \mathcal{COP} . \tag{4.3}$$

There are many well-studied outer approximations of the cone of copositive matrices, we list two of them here. One class of outer approximations is the polyhedral outer approximation proposed by [22]

$$\mathcal{O}_\mathcal{P} = \{A \in \mathcal{S} : u^T Au \geq 0 \text{ for all } u \in V_\mathcal{P}\}, \tag{4.4}$$

where $\mathcal{P}$ is a simplicial partition of the standard simplex, and $V_\mathcal{P}$ is the set of all vertices in $\mathcal{P}$. A simple case is to choose $\mathcal{P} = \Delta^S$, leading to $\mathcal{O}_{\Delta^S} = \{A \in \mathcal{S} : A_{ii} \geq 0 \text{ for all } i\}$. Another class of outer approximations of $\mathcal{COP}^n$ is introduced in [2], which is defined as

$$\mathcal{O}_{\delta(n,r)} = \{A \in \mathcal{S} : u^T Au \geq 0 \text{ for all } v \in \delta(n,r)\}, \quad r = 0, 1, 2, \ldots, \tag{4.5}$$

where $\delta(n,r) = \bigcup_{k=0}^{r} \Delta(n,k)$, and $\Delta(n,r) = \{x \in \Delta_n : (r+2)x \in \mathbb{N}^n\}$. The initial, $r = 0$ outer approximation is then

$$\mathcal{O}_{\delta(n,0)} = \{A \in \mathcal{S}^n : A_{ii} \geq 0, i = 1, \ldots, n; A_{ii} + A_{jj} + 2A_{ij} \geq 0, 1 \leq i < j \leq n\}. \tag{4.6}$$

By replacing the copositivity constraint $S \in \mathcal{COP}^n$ with a more relaxed constraint $S \in \mathcal{O}_U$, we obtain a linear program (dLP$_U$) which can be solved in polynomial time, and provides an upper bound for (COP). Note that the outer approximation may be improved by iteratively updating $U \to U \cup \{u\}$ for some $u \in \Delta \backslash U$, the more points we include from the standard simplex, the better the outer approximation. We propose an efficient approach to generate $u \in \Delta \backslash U$ such that the outer approximation is iteratively refined.

Let $y^U \in \mathbb{R}^m$ be the optimal solution of (dLP$_U$), the $S(y) = C - \sum_{i=1}^m y_i A_i$. Then we solve the following standard quadratic program (SQP)

$$\begin{aligned} z(S(y^U)) = \quad & \min \quad u^\intercal S(y^U) u \\ & \text{s.t.} \quad e^\intercal u = 1 \\ & \qquad\;\; u \in \mathbb{R}^n_+. \end{aligned} \tag{stQP(U)}$$

Let u^* denotes the optimal solution to (stQP(U)). Then one has two scenarios:

1. If $z(S(y^U)) = (u^*)^T S(y^U) u^* \geq 0$, then $S(y^U) \in \mathcal{COP}$, and y^U is the optimal solution to (COP),
2. If $z(S(y^U)) = (u^*)^\intercal S(y^U) u^* < 0$, then u^* is a certificate that $S(y^U)$ is not copositive.

If we are in scenario 2, one can generate a tightening inequality for (dLP$_U$) using the optimal solution u^*; namely

$$\left\langle C - \sum_{i=1}^m y_i A_i, u^*(u^*)^T \right\rangle \geq 0. \tag{4.7}$$

The valid inequality (4.8) is guaranteed to cut off the optimal solution y^U of (dLP$_U$) as

$$\left\langle C - \sum_{i=1}^m y_i A_i, u^*(u^*)^T \right\rangle = (u^*)^\intercal S^U u^* < 0. \tag{4.8}$$

We repeat the process until condition 1 is satisfied. The full algorithm is described below

Algorithm 2: Approximation algorithm for (COP)

```
1:  Set $\epsilon > 0$ and the finite set $U \subset \Delta_n^S$,
2:  while True do
3:      Solve (dLP$_U$)
4:      Let $y^U$ denote the optimal solution of (dLP$_U$)
5:      Set $S(y^U) = C - \sum_{i=1}^m y_i^U A_i$
6:      Solve the (stQP(U))
7:      if $z(S(y^U)) \leq -\epsilon$ then
8:          Set $U \leftarrow U \cup \{\arg\min\{S(y^U)\}\}$
9:      else
10:         Break
11:     end if
12: end while
13: return $y^U$ the $\epsilon$-optimal solution of (COP).
```

There is some freedom in the selection of the initial set U, one may choose $U = \delta(n, r)$ for some small r [2] or $U = V_{\Delta_n^S}$ [22], or any other outer approximation. Note that instead of terminating the algorithm when $z \geq 0$, we use the stopping criteria $z \geq -\epsilon$, where ϵ is a user chosen tolerence. This is to avoid numerical difficulty as the matrix $S(y^U)$ gets close to copositivity. And as a result, the optimal solution $S(y^U)$ we get is ϵ-copositive, where a matrix S is ϵ-copositive matrix if $u^\intercal S u \geq -\epsilon$ for all $u \in \Delta$.

4.4 Lower Bound for COP

Algorithm 2 provides a solution to (COP) that is ϵ-copositive [23], but does not provide a measure on how far away our current objective value of (dLP$_U$) is from the optimal solution of (COP) at any given iteration. In this section, we show that a lower bound for the (COP) problem can be obtained using the optimal objective of the SQP subproblem (stQP(U)).

The following proposition provides a way to compute a lower bound of (COP).

Proposition 6 (Lower and upper bounds)**.** *Assume X is an optimal solution to* (CP)*. For any finite $U \subset \Delta^n$ let y^U be the optimal solution to* (dLP$_U$)*, $S(y^U) = C - \sum_{i=1}^m y_i^U A_i$, and*

$z(S(y^U)) < 0$ *be the optimal objective value of* (stQP(U)). *Then we have*

$$d^U + z(S(y^U)) \langle ee^T, X \rangle \leq d^{COP} \leq d^U.$$

Proof. The second half of the inequality follows from Lemma 4.2.2.

Next we prove the first half of the inequality. Let x be the dual multipliers associated with the constraints of (dLP$_U$) and λ be the dual multipliers associated with the constraints of (dLP$_{\Delta^n}$). Let y^* be the optimal solution of (dLP$_U$), x^* be the optimal solution of the dual of (dLP$_U$), and λ^* be the optimal solution of the dual of (dLP$_{\Delta^n}$). Let $L(y, x)$ be the Lagrangian function of (dLP$_U$).

From Lemma 4.2.4 we know that $\mathcal{CP}^n$ is finitely generated, and thus there are only finitely many λ_k^* that are nonzero, thus Theorem 4.2.3 holds for (dLP$_{\Delta^n}$), and we have

$$d^\Delta = \min_x \max_{y,\lambda} \left(b^\intercal y + \sum_{u \in U} x_j \left(u^\intercal C u - \sum_{i=1}^m y_i u^\intercal A_i u \right) + \sum_{u \in \Delta_n^S \setminus U} \lambda_k \left(u^\intercal C u - \sum_{i=1}^m y_i u^\intercal A_i u \right) \right) \tag{4.9}$$

$$= \min_x \max_y \left(L(y, x) + \sum_{u \in \Delta_n^S \setminus U} \lambda_u^* \left(u^\intercal C u - \sum_{i=1}^m y_i u^\intercal A_i u \right) \right) \tag{4.10}$$

$$\geq \max_y \min_x \left(L(y, x) + \sum_{u \in \Delta_n^S \setminus U} \lambda_u^* \left(u^\intercal C u - \sum_{i=1}^m y_i u^\intercal A_i u \right) \right) \tag{4.11}$$

$$= \max_y \left(L(y, x^*) + \sum_{u \in \Delta_n^S \setminus U} \lambda_u^* \left(u^\intercal C u - \sum_{i=1}^m y_i u^\intercal A_i u \right) \right) \tag{4.12}$$

$$\geq L(y^*, x^*) + \sum_{u \in \Delta_n^S \setminus U} \lambda_u^* \left(u^\intercal C u - \sum_{i=1}^m y_i^* u^\intercal A_i u \right) \tag{4.13}$$

$$= d^U + \sum_{u \in \Delta_n^S \setminus U} \lambda_u^* u^T S(y^*) u, \tag{4.14}$$

where $d^U = L(y^*, x^*)$ by strong duality of LP, (4.10) follows from (4.9) by substituting $\lambda = \lambda^*$, (4.11) follows from (4.10) by Lemma 36.1 from [87], and by LP duality, (4.12) follows from (4.11) from the fact that there are only a finite number of λ_u^* that is non-zero. Now note that

$$z(S(y^*)) \leq u^T S(y^*) u \text{ for any } u \in \Delta_n^S,$$

and since $z(S(y^*)) < 0$

$$d^{\Delta} \geq d^U + z(S(y^*)) \sum_{u \in \Delta_n^S \setminus U} \lambda_u^*.$$

Note that the optimal solution X^* to (CP) can be written as

$$X^* = \sum_{u \in \Delta^n} \lambda_u^* u u^T,$$

so we have

$$\langle ee^T, X^* \rangle = \sum_{u \in \Delta^n} \lambda_u^* (eu)^2 = \sum_{u \in \Delta^n} \lambda_u^* \geq \sum_{u \in \Delta_n^S \setminus U} \lambda_u^*,$$

since $\lambda_u^* \geq 0$ for all $u \in \Delta_n^S$. Therefore

$$d^U + z^U \left\langle ee^T, X^* \right\rangle \leq d^{\Delta} = d^{COP}.$$

$\square$

Note that in Proposition 6 we assume $z(S(y^U)) < 0$, since if $z(S(y^U)) \geq 0$ then we have found the optimal solution to (COP).

To compute the lower bound for (COP) proposed in Proposition 6, we need to find an upper bound for the optimal solution X of (CP). We propose an upper bound of X for the class of completely positive programs obtained by reformulating indefinite QPs.

Consider the (indefinite) QP:

$$\begin{aligned} z = \quad & \min \quad x^{\mathsf{T}} Q x \\ & \text{s.t.} \quad a_i^{\mathsf{T}} x = b_i, \quad i = 1, \ldots, m \\ & \qquad\ \ x \geq 0, \end{aligned} \tag{QP}$$

where $F = \{x \in \mathbb{R}_+^n : a_i^{\mathsf{T}} x = b_i, i = 1, \ldots, m\} := \{x \in \mathbb{R}_+^n : Ax = b\} \neq \emptyset$, and F is bounded.

To show that the upper bound on the variables of (QP) gives an upper bound on the variables of the corresponding completely positive programs, we first state the following proposition.

Proposition 7 (Proposition 1.9 (Farkas lemma III) [98])**.** *For $n, m \in \mathbb{N}$, let $A \in \mathbb{R}^{m \times n}$,*

$b \in \mathbb{R}^m, c \in \mathbb{R}^n$ and $c_0 \in \mathbb{R}$. Let $F = \{x : Ax \leq b\}$ be non-empty. Then $c_0 + c^\intercal x \geq 0$ for all $x \in F$ if and only if there exists $u \geq 0$ such that

$$A^\intercal u = -c, \;\; b^\intercal u \leq c_0.$$

We first show that a valid upper bound can be obtained for the sum of the variables of (QP).

Proposition 8. *Let $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$. If $F = \{x \in \mathbb{R}^n_+ : Ax = b\} \neq \emptyset$, and F is bounded. Then, there exists $u \in \mathbb{R}^m$ such that $e^\intercal x \leq u^\intercal b$ is a valid inequality for F.*

Proof. The feasible set of (QP) $F = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$ can be written as

$$F = \left\{ x \in \mathbb{R}^n : \begin{bmatrix} A \\ -A \\ -I \end{bmatrix} x \leq \begin{bmatrix} b \\ -b \\ 0 \end{bmatrix} \right\}.$$

Since F is nonempty, from Proposition (7), it follows that $c_0 + c^\intercal x \geq 0$ for all $x \in S$ if and only if there exists $u^+ \geq 0, u^- \geq 0, \lambda \geq 0$ such that

$$\begin{bmatrix} A^\intercal & -A^\intercal & -I \end{bmatrix} \begin{bmatrix} u^+ \\ u^- \\ \lambda \end{bmatrix} = -c, \quad \begin{bmatrix} b^\intercal & -b^\intercal & 0 \end{bmatrix} \begin{bmatrix} u^+ \\ u^- \\ \lambda \end{bmatrix} \leq c_0,$$

which means there exists $u \in \mathbb{R}^m, \lambda \in \mathbb{R}^n_+$ such that

$$A^\intercal u - \lambda = -c, \quad b^\intercal u \leq c_0.$$

Since F is bounded, there exist $c_0 > 0$ such $e^\intercal x \leq c_0$ for all $x \in F$, or equivalently $-e^\intercal x + c_0 \geq 0$ for all $x \in F$. Then we have from Proposition 7 that there exists $u' \in \mathbb{R}^m, \lambda' \in \mathbb{R}^n_+$ such that $e = A^\intercal u' - \lambda'$, $b^\intercal u' \leq c_0$.

Then, for any $x \in F$ we have that

$$\begin{aligned} e^T x &= (u')^T A x - (\lambda')^T x \\ &= (u')^T b - (\lambda')^T x \\ &\leq (u')^T b, \end{aligned}$$

where the last inequality follows from $x \geq 0$, $\lambda' \geq 0$. Thus is a valid inequality for F. $\square$

According to [33], under the boundedness and non-empty condition on F, then z is equivalent to:

$$\begin{aligned} z = \quad & \min \quad \left\langle \begin{bmatrix} 0 & f^\intercal \\ f & Q \end{bmatrix}, X \right\rangle \\ & \text{s.t.} \quad \left\langle \begin{bmatrix} 0 & \frac{1}{2}a_i^\intercal \\ \frac{1}{2}a_i & 0 \end{bmatrix}, X \right\rangle = b_i, && i = 1, \ldots, m \\ & \qquad \left\langle \begin{bmatrix} 0 & 0 \\ 0 & a_i a_i^\intercal \end{bmatrix}, X \right\rangle = b_i^2, && i = 1, \ldots, m \\ & \qquad \left\langle \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, X \right\rangle = 1 \\ & \qquad X \in \text{ Completely Positive} \subseteq (X \geq 0), \end{aligned}$$

Also from [33], it follows that the optimal solution of the above problem X^* satisfies

$$X^* \in \text{conv} \left(\begin{bmatrix} 1 \\ x^* \end{bmatrix} \begin{bmatrix} 1 \\ x^* \end{bmatrix}^\intercal : x^* \text{ is an optimal solution of (QP)} \right),$$

thus

$$\left\langle \begin{bmatrix} 1 & e^\intercal \\ e & ee^\intercal \end{bmatrix}, X^* \right\rangle = \sum_{i=1}^{m} \lambda_i \left\langle J, \begin{bmatrix} 1 \\ x^* \end{bmatrix} \begin{bmatrix} 1 \\ x^* \end{bmatrix}^\intercal \right\rangle \leq \sum_{i=1}^{m} \lambda_i (1 + u^\intercal b)^2 \leq (1 + u^\intercal b)^2,$$

where J is the matrix of all ones. Therefore an upper bound on the variables of the completely positive program (CP) can be obtained from the upper bound on the variables of QP.

4.5 Convergence

Algorithm 2 is a cutting-plane type algorithm for solving linear semi-infinite programs. A conceptual frame work for a general cutting-plane algorithm is introduced in [51].

Algorithm 3: Conceptual cutting-plane algorithm for linear semi-infinite programs

```
Step 1  Solve the LP subproblem
            $(P_r)$  inf  $c^\intercal x$
                      s.t. $a_t^\intercal x \geq b_t, \quad t \in T_r$
        If $(P_r)$ is inconsistent then stop.
        Otherwise, calculate an optimal solution of $(P_r)$, $x^r$; then go to Step 2.
Step 2  Calculate $s_r = \inf_{t \in T} g(t, x^r)$.
        If $s_r \geq -\epsilon$ then stop.
        If $s_r < -\epsilon$ then find a non-empty finite set $S_r$ of $\epsilon_r$-minimizers of the
        slack function at $x^r$, i.e. each $t \in S_r$ must satisfy $g(t, x^r) \leq s_r + \epsilon_r$.
        Then replace $r$ by $r + 1$, take $T_{r+1} = T_r \cup S_r$ and loop to Step 1.
```

It is easy to see that algorithm 2 follows the above framework, where the LP subproblem in step 1 is (dLP$_U$), and the optimization problem in step 2, $\inf_{t \in T} g(t, x^R)$; is the standard quadratic program (stQP(U)) where $U = T_r$. And $S_r = \arg\min(z(y^U))$.

According to Theorem 11.2 in [51], algorithm (2) converges, and terminates in finite step if $\epsilon > 0$.

Theorem 4.5.1 (Theorem 11.2 in [51])**.** *Assume that* (PLSIP) *is consistent, that* $\{a_t, t \in T\}$ *is a bounded set and that* $g(\cdot, x^r)$ *is bounded from below (or has minimizers, if* $\epsilon_r = 0$*) at each iteration of Algorithm 3 (this is true when* (PLSIP) *is continuous). Then, it generates either a finite sequence or an infinite sequence having cluster points which are optimal solutions of* (PLSIP)*. Thus finite termination will occur if* $\epsilon > 0$*.*

In the case of Algorithm 2, $\{a_t, t \in T\} = \{[-u^T A_1 u, \ldots, -u^T A_m u^T]^\intercal, u \in \Delta^n\}$ is compact thus bounded, and $g(\cdot, x^r) = u^T S(x^r) u$ which is bounded below at each iteration r since feasible set $u \in \Delta^{n+1}$ is compact. By Theorem 4.5.1, Algorithm 2 converges and achieve finite termination. Note that this is a particular version of Algorithm 3 named Alternating Algorithm [57] for LSIP continuous problems, obtained by selecting $\epsilon = \epsilon_1 = \epsilon_2 = \cdots = 0$, and taking $|S_r| = 1$ in all iterations.

4.6 Other classes of valid inequalities

In step 6 of Algorithm 2, for each iteration r, one has the freedom of adding $u^r \in \Delta^S_{n+1}$ to U such that u^r satisfies $(u^r)^\intercal S(y^U)u^r < -\epsilon$. We introduce two approaches to generate valid u^r's at each iteration r such that $(u^r)^\intercal S(y^U)u^r \leq -\epsilon$, where $\epsilon > 0$ is satisfied.

The first approach is inspired by Lemma 3 of [23]

Lemma 4.6.1 (Lemma 3 in [23]). *Let $A \in \mathcal{S}^n$ and $q(x) = x^T Ax$. Assume we are given $u, v \in, \mathbb{R}^n$ with*

$$\alpha = u^T Au \geq 0, \quad \beta = v^T Av \geq 0, \quad \gamma = u^T Av < 0.$$

Then the function $f(\lambda) = q(\lambda u + (1-\lambda)v)$ attains its minimum at $\bar{\lambda} = \frac{\beta-\gamma}{\alpha-2\gamma+\beta} \in (0,1)$.

Let U_r be the finite set U at iteration r. Using Lemma 4.6.1, at each iteration r of Algorithm 3 one can check if there exists $u, v \in U_r$ such that $u^T Av < 0$, and if $f(\bar{\lambda}u + (1-\bar{\lambda})v) < 0$, then we have found $w = \bar{\lambda}u + (1-\bar{\lambda})v \in \Delta^n$ such that $w^T S(y^U)w < 0$. According to Lemma 4 in [23], one may also check if $\frac{\gamma}{\gamma-\alpha} > \frac{\beta}{\beta-\gamma}$, if the inequality holds then we know $q(w) < 0$ where $w = \bar{\lambda}u + (1-\bar{\lambda})v \in \Delta^n$.

Another approach to generate valid inequalities at each iteration of Algorithm 2 is to utilize Theorem 1 in [36]

Theorem 4.6.2 (Theorem 1 in [36]). *If $M \in \mathcal{S}^n$, the following two statements are equivalent:*

1. *M is copositive;*
2. *For all $J \subseteq \{1, \dots, n\}$, $J \neq \emptyset$, the following system has a solution:*

$$M_{JJ}x_J \geq 0, \quad x_J \geq 0, \quad e^\intercal_{|J|}x_J = 1. \tag{4.15}$$

By Farkas Lemma, for system (4.15) to have a solution, it is equivalent for the following system to be infeasible

$$My < 0, \quad y \geq 0. \tag{4.16}$$

One may find a y by solving the following optimization problem

$$\begin{aligned} l^* = \min \quad & \alpha \\ \text{s.t.} \quad & M_{JJ}y \le \alpha e \\ & e^T y = 1 \\ & y \ge 0, \end{aligned} \tag{LP}$$

if $l^* < 0$, then the optimal solution y^* of (LP) yields $y^T M_{JJ} y < 0$ as $M_{JJ}y \le 0$, $y \ge 0$ and $e^T y = 1$. So we obtain a valid inequality $M_{JJ}y^* \ge 0$.

4.7 Experiment Results

In this section, we present and discuss our computational results. We implemented our algorithm in Matlab 2017a with quadprogIP from [95], where the underlying optimization solver is CPLEX 12.8.0. The algorithm is tested on 2.9 GHz Intel Core i5 with 8 GB memory. The optimal tolerance is $\epsilon = 10^{-6}$.

The test instances were obtained by reformulating the general QPs

$$\begin{aligned} \min \quad & x^T Q x + f^T x \\ \text{s.t.} \quad & Ax = b \\ & x \ge 0 \end{aligned}$$

as the following copositive program

$$\begin{aligned} \max \quad & \sum_{i=1}^m b_i \mu_i + \sum_{i=1}^m b_i^2 \bar{\mu}_i + \mu_0 \\ \text{s.t.} \quad & \begin{bmatrix} 0 & \frac{f^\intercal}{2} \\ \frac{f}{2} & Q \end{bmatrix} - \sum_{i=1}^m \mu_i \begin{bmatrix} 0 & \frac{a_i^\intercal}{2} \\ \frac{a_i}{2} & 0 \end{bmatrix} - \sum_{i=1}^m \bar{\mu}_i \begin{bmatrix} 0 & 0 \\ 0 & a_i a_i^\intercal \end{bmatrix} - \mu_0 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \in \mathcal{COP}^{n+1}. \end{aligned}$$

We selected a subset of non-convex QPs from the test sets CUTEr and globallib, and compare our algorithm, named CopLSIP, with Yıldırım's algorithm proposed in [2]. The results are summarized in Table 4.1. It can be observed that CopLSIP outperforms Yıldırım's algorithm in every instances. CopLSIP is able to obtain the optimal solution within the time limit of 1800 seconds, while Yıldırım's algorithm fails to obtain the optimal solution for most of the instances. For the instances Yıldırım's algorithm did not achieve optimality, the algorithm yields weak bounds for most of the instances. It can be observed that CopLSIP uses significantly more iterations comparing to Yıldırım's algorithm, since each iteration of CopLSIP consists of solving an LP and a SQP, which turns out to be computationally cheaper comparing to the exponentially growing number of LPs Yıldırım's algorithm solves at each iteration. This explains why Yıldırım's algorithm, even on small instances, is unable to compute outer approximations where the r is large efficiently.

Table 4.1: Comparing CopLSIP with Yıldırım's algorithms

		CopLSIP			Yıldırım		
instance	n	Iter	outer approx	Time (s)	r	outer approx	Time (s)
ex2_1_1	7	18	-9050.0000	2.06	9	-9050.0000	23.21
ex2_1_2	9	85	-461.7492	10.73	9	-93.6918	1800.00
ex2_1_4	12	385	-32.4495	52.51	5	55868.2816	1800.00
hs044	11	530	-29.9992	62.90	6	8644.4755	1800 .00
st_bpk1	11	641	-25.9994	78.88	6	6009.2559	1800.00
st_bpk2	11	641	-25.9994	88.86	6	6009.2559	1800 .00
st_bpv2	10	367	-15.9984	44.59	7	25360.3214	1800 .00
st_bsj2	9	683	2.0015	82.88	9	56.5437	1800.00
st_bsj4	11	302	-983175.0751	35.02	6	287994.6429	1800.00
st_e22	8	409	-169.9975	44.22	12	15279.3152	1800.00
st_e23	5	82	-2.1666	8.94	50	-2.1667	1360.90
st_e24	7	226	16.001	24.48	18	581.0979	1800.00
st_fp1	7	18	-9050.0000	2.62	9	-9050.0000	23.1
st_fp2	9	85	-461.7492	9.54	9	-93.6918	1800.00
st_pan1	8	297	-10.5671	33.64	12	88.5276	1800.00
st_ph1	12	455	-460.2063	59.43	5	23355.1765	1800.00
st_phex	8	489	-169.9975	42.92	12	15279.3152	1800.00
st_qpc_m0	5	68	-9.9999	7.30	6	-10	0.4
st_qpc_m1	11	499	-947.54	62.21	6	20192.0452	1800 .00
st_qpk1	7	327	-5.9999	40.74	7	-6	6.00

Next, we compare our basic algorithm with the the version of the algorithm that uses either one of the two different valid inequality generation approaches to generate extra cuts. The algorithm that uses cuts generated by solving an LP is named CopLSIP_LP, and the algorithm that generates cuts by examining the convex combination of previous cuts is named CopLSIP_comb. Note that CopLSIP_LP solves (LP) first to try to obtain a LP cut. If no LP cut is found, then (stQP(U)) is solved. The subprincipal matrix M_{JJ} in (LP) is selected by $J = \{i \in \mathbb{N} : u_i > 0\}$, where u is the optimal solution of (stQP(U)). The results are presented in Table 4.7. It can be seen that both CopLSIP_LP and CopLSIP_comb outperforms CopLSIP, since CopLSIP_comb generates extra inequalities at each iteration to speed up the solution process, while CopLSIP_LP solves a LP in place of a SQP in many iterations. It can be seen that CopLSIP_Hybrid takes the most number of iterations, since CopLSIP_LP only solves a SQP when there is no valid inequality that can be generated by solving (LP), and the valid inequalities generated by solving (LP) are not as strong as the inequalities generated by solving SQPs. However, CopLSIP_LP has better performance, since solving an LP is faster than solving a SQP. One note is that CopLSIP_Hybrid is prone to numerical issues, so a efficient version of CopLSIP_Hybrid may need some fine tuning. It is interesting to note that CopLSIP_comb uses fewer iterations to achieve optimality.

We also compare with another algorithm where both LP cut generation approach are used, and name it CopLSIP_Hybrid. The results are shown in Table 4.7. The main difference of CopLSIP_Hybrid is that we use both cut generation approaches at each iteration. We see good improvement comparing to CopLSIP, since we generate more cuts at each iteration. What's more, we see improvements comparing to CopLSIP_comb, which suggests the cuts obtained by solving (LP) can be beneficial for solving the problem.

4.8 Conclusion

In this chapter, we proposed a linear semi-infinite reformulation of copositive programs, and a cutting-plane algorithm for solving copositive programs. We presented conditions for linear systems of infinitely many equalities to be Farkas-Minkowski, such that strong duality holds between the primal and dual pair of linear semi-infinite reformulations of copositive programs. The cutting-plane algorithm utilizes the efficiency of the non-convex quadratic solver introduced in [95], and it is shown that the algorithm converges and terminates in finite iteration under appropriate selection of parameters. A lower bound for the copositive program at each iteration is provided in a closed form. Our computational results demonstrated that our algorithm is effective in solving the QP reformulated copositive programs.

4.9 Future works

For future work, a question worth exploring is if the bounds of the dual variables of (QP) is valid for the variables of (COP). In practice, when solving the master problem (dLP$_U$), one

		CopLSIP			CopLSIP_comb			CopLSIP_LP		
instance	n	Iter	outer approx	Time (s)	Iter	outer approx	Time (s)	Iter	outer approx	Time (s)
ex2_1_1	7	18	-9050.0000	2.06	13	-9050.0000	2.17	21	-9050.0000	1.01
ex2_1_2	9	85	-461.7492	10.73	50	-461.7497	6.02	105	-461.7489	2.55
ex2_1_4	12	385	-32.4495	52.51	99	-32.4511	13.08	447	-32.4488	16.67
hs044	11	530	-29.9992	62.9	139	-29.9994	18.06	825	-29.9994	23.71
st_bpk1	11	641	-25.9994	78.88	175	-25.9997	27.13	940	-25.9994	22.55
st_bpk2	11	641	-25.9994	88.86	175	-25.9997	27.81	940	-25.9994	22.75
st_bpv2	10	367	-15.9984	44.59	83	-15.9986	10.25	404	-15.9981	11.16
st_bsj2	9	683	2.0015	82.88	177	2.002	25.41	738	2.0015	21.24
st_bsj4	11	302	-983175.0751	35.02	135	-983177.7353	16.1	320	-983178.1178	9.85
st_e22	8	409	-169.9975	44.22	123	-169.9984	15.17	611	-169.9972	15.13
st_e23	5	82	-2.1666	8.94	22	-2.1666	2.74	73	-2.1666	2.5
st_e24	7	226	16.001	24.48	72	16.0004	7.53	326	16.0009	9.14
st_fp1	7	18	-9050.0000	2.62	18	-9050.0000	2.19	21	-9050.0000	1.29
st_fp2	9	85	-461.7492	9.54	50	-461.7491	5.96	105	-461.7489	3.21
st_pan1	8	297	-10.5671	33.64	92	-10.5672	10.28	459	-10.5673	10.53
st_ph1	12	455	-460.2063	59.43	139	-460.2156	18.66	673	-460.2226	12.94
st_phex	8	489	-169.9975	42.92	123	-169.9984	15.81	611	-169.9972	14.64
st_qpc_m0	5	68	-9.9999	7.31	27	-9.9999	2.83	85	-10.0000	3.69
st_qpc_m1	11	499	-947.54	62.21	121	-947.5447	15.13	888	-947.5433	22.81
st_qpk1	7	327	-5.9999	40.74	80	-5.9999	9.85	316	-5.9999	11.15

Table 4.2: Comparing the performance among CopLSIP, CopLSIP_comb, and CopLSIP_LP algorithms

		CopLSIP			Yıldırım			Hybrid		
instance	n	Iter	outer approx	Time (s)	r	outer approx	Time (s)	Iter	outer approx	Time (s)
ex2_1_1	7	18	-9050.0000	2.06	9	-9050.0000	23.21	17	-9049.9999	1.92
ex2_1_2	9	85	-461.7492	10.73	9	-93.6918	1800.00	31	-461.7496	3.45
ex2_1_4	12	385	-32.4495	52.51	5	55868.2816	1800.00	94	-32.45	11.36
hs044	11	530	-29.9992	62.91	6	8644.4755	1800.00	134	-29.9993	16.92
st_bpk1	11	641	-25.9994	78.88	6	6009.2559	1800.00	143	-25.9995	17.87
st_bpk2	11	641	-25.9994	88.86	6	6009.2559	1800.00	143	-25.9995	18.06
st_bpv2	10	367	-15.9984	44.59	7	25360.3214	1800.00	79	-15.9989	9.04
st_bsj2	9	683	2.0015	82.88	9	56.5437	1800.00	248	2.0008	42.38
st_bsj4	11	302	-983175.0751	35.02	6	287994.6429	1800.00	125	-983177.3015	12.96
st_e22	8	409	-169.9975	44.22	12	15279.3152	1800.00	105	-169.9974	11.65
st_e23	5	82	-2.1666	8.94	50	-2.1667	1360.95	35	-2.1666	3.46
st_e24	7	226	16.001	24.48	18	581.0979	1800.00	60	16.0004	6.32
st_fp1	7	18	-9050.0000	2.62	9	-9050.0000	23.12	17	-9049.9999	1.76
st_fp2	9	85	-461.7492	9.54	9	-93.6918	1800.00	31	-461.7496	3.32
st_pan1	8	297	-10.5671	33.64	12	88.5276	1800.00	66	-10.5672	6.94
st_ph1	12	455	-460.2063	59.43	5	23355.1765	1800 .00	128	-460.2147	14.23
st_phex	8	489	-169.9975	42.92	12	15279.3152	1800.00	105	-169.9974	12.04
st_qpc_m0	5	68	-9.9999	7.31	6	-10.0000	0.42	22	-10.0000	2.48
st_qpc_m1	11	499	-947.54	62.21	6	20192.0452	1800.00	87	-947.5478	9.45
st_qpk1	7	327	-5.9999	40.74	7.62	-6.0000	6.27	82	-5.9999	9.15

Table 4.3: Comparing the performance among CopLSIP, Yıldırı's algorithm and CopLSIP_Hybrid

may encounter the issue of unbounded optimal objective. This may happen as the variables of (dLP$_U$) are unrestricted and the particular outer approximation is not bounded. In general, bounds on the variables for (dLP$_U$) are needed. We propose a way to find the bounds on the variables of (dLP$_U$) where the original problem (COP) of (dLP$_U$) is the dual of a (CP) that is obtained by reformulating a QP.

4.9.1 Bounding the dual variable

To compute the lower bound for (COP) proposed in Proposition 6, we need to find an upper bound for the optimal solution X^* of (CP). We propose an upper bound of X for the class of completely positive programs obtained by reformulating indefinite QPs with bounded feasible set.

Lemma 4.9.1. *Assume the pair of problems* (CP)-(COP) *corresponds to the reformulations of a bounded* (QP) *in [33] where the feasible set is* $\{x \in \mathbb{R}^n : Ax = b\}$. *Assume* (x^*, λ^*, μ^*) *is a KKT point for* (QP) *such that* x^* *is optimal. Then*

1. *one can construct* S, y *optimal solution to* (COP) *from* (x^*, λ^*, μ^*),

2. *and bounds on* λ^* *implies the optimal objective of* d^U *is bounded if* $\{e_1, \dots, e_n\} \subseteq U$.

Proof. Consider the following QP:

$$\begin{aligned} p^* = \quad & \min \quad x^\intercal Q x + f^\intercal x \\ & \text{s.t.} \quad a_i^\intercal x = b_i, \qquad i = 1, \dots, m \\ & \qquad\; (a_i^\intercal x)^2 = b_i^2, \quad i = 1, \dots, m \\ & \qquad\; x \geq 0, \end{aligned} \tag{QP+}$$

which is equivalent to (QP). The lagrangian is

$$L(x, \lambda, \mu) = x^T Q x + f^T x + \sum_{i=1}^m (b_i - a_i^T x)\mu_i + \sum_{i=1}^m (b_i^2 - (a_i^T x)^2)\bar{\mu}_i - \sum_{i=1}^n \lambda_i x_i.$$

Let d^* denote the optimal objective value of the dual problem of the (QP+). We have

$$\begin{aligned}
d^* &= \max_{\mu,\lambda\geq 0} \min_{x\geq 0} L(x,\lambda,\mu) \\
&= \max_{\mu,\lambda\geq 0} \min_{x\geq 0} x^T Q x + f^T x + \sum_{i=1}^m (b_i - a_i^T x)\mu_i + \sum_{i=1}^m (b_i^2 - (a_i^T x)^2)\bar{\mu}_i - \sum_{i=1}^n \lambda_i x_i \\
&= \max_{\mu,\lambda\geq 0} \sum_{i=1}^m b_i\mu_i + \sum_{i=1}^m b_i^2 \bar{\mu}_i + \mu_0 \qquad (4.17) \\
&\text{s.t.} \quad \mu_0 \leq x^T Q x + f^T x - \sum_{i=1}^m a_i^T x \mu_i - \sum_{i=1}^m (a_i^T x)^2 \bar{\mu}_i - \sum_{i=1}^n \lambda_i x_i \quad \forall x \geq 0. \qquad (4.18)
\end{aligned}$$

Note that constraint (4.18) can be written as

$$\begin{aligned}
&\begin{bmatrix} 1 \\ x \end{bmatrix}^T \begin{bmatrix} 0 & \frac{f^\intercal}{2} \\ \frac{f}{2} & Q \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} - \sum_{i=1}^m \mu_i \begin{bmatrix} 1 \\ x \end{bmatrix}^T \begin{bmatrix} 0 & \frac{a_i^\intercal}{2} \\ \frac{a_i}{2} & 0 \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} \\
&\qquad - \sum_{i=1}^m \bar{\mu}_i \begin{bmatrix} 1 \\ x \end{bmatrix}^T \begin{bmatrix} 0 & 0 \\ 0 & a_i a_i^\intercal \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} - \mu_0 \begin{bmatrix} 1 \\ x \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} \geq 0,
\end{aligned}$$

where $x \geq 0$, and this is equivalent to

$$\begin{bmatrix} 0 & \frac{f^\intercal}{2} \\ \frac{f}{2} & Q \end{bmatrix} - \sum_{i=1}^m \mu_i \begin{bmatrix} 0 & \frac{a_i^\intercal}{2} \\ \frac{a_i}{2} & 0 \end{bmatrix} - \sum_{i=1}^m \bar{\mu}_i \begin{bmatrix} 0 & 0 \\ 0 & a_i a_i^\intercal \end{bmatrix} - \mu_0 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \in \mathcal{COP}^n . \qquad (4.19)$$

It can be observed that the problem with objective function (4.17) and constraint (4.20) is equivalent to the dual of the (QP+), as well as (COP). Therefore the dual of the above QP is equivalent to (COP), and thus one can construct the optimal solution to (COP) from the optimal solution of the dual of the (QP+).

Consider another representation of (QP)

$$\begin{aligned}
p^* = \quad \min \quad & x^\intercal Q x + f^\intercal x \\
\text{s.t.} \quad & (a_i^\intercal x - b_i)^2 = 0, \quad i = 1, \ldots, m \qquad \text{(QP++)} \\
& x \geq 0,
\end{aligned}$$

which is equivalent to (QP), which is equivalent to (QP) and (QP+). Even more, the corresponding completely positive reformulation are equivalent, thus the dual of the corresponding completely positive reformulations, namely the copositive programs, are also equivalent. We show that a KKT point of (QP) is also a KKT point of (QP++), such that a valid bound on the dual variables of (QP++) is also a valid bound on the dual variables of (QP).

Next, we show that d^U is bounded given $\{e_1, \ldots, e_n\} \subseteq U$. Let

$$S = \begin{bmatrix} 0 & \frac{f^\intercal}{2} \\ \frac{f}{2} & Q \end{bmatrix} - \sum_{i=1}^{m} \mu_i \begin{bmatrix} 0 & \frac{a_i^\intercal}{2} \\ \frac{a_i}{2} & 0 \end{bmatrix} - \sum_{i=1}^{m} \bar{\mu}_i \begin{bmatrix} 0 & 0 \\ 0 & a_i a_i^\intercal \end{bmatrix} - \mu_0 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}. \tag{4.20}$$

Then from the constraint $e_1^T S e_1 \geq 0$ of the outer approximation induced by U we get $\mu_0 \leq 0$. Similarly, for each $i \in \{2, \ldots, n\}$ we get

$$e_i^T S e_i = \sum_{j=1}^{m} \bar{\mu}_j a_{ji}^2 \leq Q_{ii},$$

writing all the constrains from $i = 2, \ldots, n$ in matrix form we get

$$(A^2)^T \bar{\mu} \leq \operatorname{diag}(Q).$$

Since A has full row rank, then $(A^2)^T$ has full column rank, so we can m linearly independent rows of $(A^2)^T$ whose indices are denoted as a set I, and denote the new matrix consist of these linearly independent rows as B, such that

$$B\bar{\mu} = [(A^2)^T]_I \bar{\mu} \leq [\operatorname{diag}(Q)]_I. \tag{4.21}$$

Since B has linearly independent rows and columns, B is invertible, so we can multiply on both side of (4.21) and get

$$\bar{\mu} \leq B^{-1}[\operatorname{diag}(Q)]_I \text{ and } \|\bar{\mu}\|_\infty \leq \|B^{-1}[\operatorname{diag}(Q)]_I\|_\infty.$$

Thus we have

$$(b^2)^T \bar{\mu} \leq (b^2)^T B^{-1}[\operatorname{diag}(Q)]_I.$$

From [95] we know that there exists an optimal KKT point (x^*, μ^*, λ^*) such that $e^T \lambda^* \leq$

M for some $M > 0$. The KKT condition yields the following valid equality for μ, λ:

$$Qx + f - A^{\intercal}\mu - (A^2)^T\bar{\mu} - \lambda = 0. \tag{4.22}$$

Since A has full row rank, we have

$$\begin{aligned}
A^{\intercal}\mu &= Qx + f - (A^2)^T\bar{\mu} - \lambda \\
\mu &= (AA^{\intercal})^{-1}A(Qx + f - (A^2)^T\bar{\mu} - \lambda) \\
\mu &= (AA^{\intercal})^{-1}AQx + (AA^{\intercal})^{-1}Af - (AA^{\intercal})^{-1}A(A^2)^T\bar{\mu} - (AA^{\intercal})^{-1}A\lambda.
\end{aligned}$$

An upper bound of μ is

$$\begin{aligned}
\|\mu\|_\infty &\le \|(AA^{\intercal})^{-1}A\|_\infty\|Qx + f - (A^2)^T\bar{\mu} - \lambda\|_\infty \\
&\le \|(AA^{\intercal})^{-1}A\|_\infty(\|Qx\|_\infty + \|f\|_\infty + \|(A^2)^T\bar{\mu}\|_\infty + \|\lambda\|_\infty) \\
&\le \|(AA^{\intercal})^{-1}A\|_\infty(\|Q\|_\infty\|UB\|_\infty + \|f\|_\infty + \|(A^2)^T\|_\infty\|\bar{\mu}\|_\infty + \|\lambda\|_\infty).
\end{aligned}$$

$\square$

Bibliography

[1] Airbus Global Market Forecast Airbus. Forecast 2017-2036, 2017.

[2] E Alper Yıldırım. On the accuracy of uniform polyhedral approximations of the copositive cone. *Optimization methods and software*, 27(1):155–173, 2012.

[3] Edward J Anderson, Miguel A Goberna, and Marco A López. Locally polyhedral linear inequality systems. *Linear algebra and its applications*, 270(1-3):231–253, 1998.

[4] N. Arima, S. Kim, and M. Kojima. A quadratically constrained quadratic optimization model for completely positive cone programming. *SIAM Journal on Optimization*, 23(4):2320–2340, 2013.

[5] N. Arima, S. Kim, and M. Kojima. Extension of completely positive cone relaxation to moment cone relaxation for polynomial optimization. *Journal of Optimization Theory and Applications*, pages 1–17, 2015.

[6] L. Bai, J.E. Mitchell, and J. Pang. On conic qpccs, conic qcqps and completely positive programs. *Mathematical Programming*, pages 1–28, 2015.

[7] P. Belotti. Couenne: a user's manual. Technical report, Clemson University, 2010. available at `http://projects.coin-or.org/Couenne/browser/trunk/Couenne/doc/couenne-user-manual.pdf`.

[8] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wachter. Branching and bounds tightening techiques for non-convex MINLP. *Optimization Methods and Software*, 24(4–5):597–634, 2009.

[9] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications.* MPS-SIAM Series on Optimization. SIAM, Philadelphia, PA, 2001.

[10] Abraham Berman and Naomi Shaked-Monderer. *Completely positive matrices.* World Scientific, 2003.

[11] Dmitry Bershadsky. *Electric multirotor design and optimization.* PhD thesis, Georgia Institute of Technology, 2017.

[12] D. Bertsekas. *Nonlinear Programming.* Athena Scientific, Belmont, MA, 1999.

[13] Dimitri P Bertsekas, Angelia Nedi, Asuman E Ozdaglar, et al. *Convex analysis and optimization*. Athena Scientific, 2003.

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

[15] I. M. Bomze, F. Frommlet, and M. Locatelli. Copositivity cuts for improving SDP bounds on the clique number. *Math. Programming*, 124(1-2):13–32, 2010.

[16] I.M. Bomze. Copositive optimization–recent developments and applications. *European Journal of Operational Research*, 216(3):509–520, 2012.

[17] I.M. Bomze, W. Schachinger, and G. Uchida. Think co (mpletely) positive! matrix properties, examples and a clustered bibliography on copositive optimization. *Journal of Global Optimization*, 52(3):423–445, 2012.

[18] Immanuel M Bomze and Etienne De Klerk. Solving standard quadratic optimization problems via linear, semidefinite and copositive programming. *Journal of Global Optimization*, 24(2):163–185, 2002.

[19] P. Bonami, A. Lodi, J. Schweiger, and A. Tramontani. Solving standard quadratic programming by cutting planes. Technical Report DS4DM-2016-001, hola, 2016. available at `http://cerc-datascience.polymtl.ca/wp-content/uploads/2016/06/Technical-Report_DS4DM-2016-001-1.pdf`.

[20] Pierre Bonami, Oktay Günlük, and Jeff Linderoth. Solving box-constrained nonconvex quadratic programs. *Optimization Online*, 2016.

[21] Mina Saee Bostanabad, João Gouveia, and Ting Kei Pong. Inner approximating the completely positive cone via the cone of scaled diagonally dominant matrices. *arXiv preprint arXiv:1807.00379*, 2018.

[22] S. Bundfuss and Dür. An adaptive linear approximation algorithm for copositive programs. *SIAM J. Optim.*, 20(1):30–53, 2009.

[23] Stefan Bundfuss and Mirjam Dür. Algorithmic copositivity detection by simplicial partition. *Linear Algebra and its Applications*, 428(7):1511–1523, 2008.

[24] S. Burer. On the copositive representation of binary and continuous nonconvex quadratic programs. *Math. Programming*, 120(2):479–495, 2009.

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

[26] S. Burer. Copositive programming. In *Handbook on semidefinite, conic and polynomial optimization*, pages 201–218. Springer, 2012.

[27] S. Burer and H. Dong. Representing quadratically constrained quadratic programs as generalized copositive programs. *Operations Research Letters*, 40(3):203–206, 2012.

[28] S. Burer and D. Vandenbussche. Globally solving box-constrained non-convex quadratic programs with semidefinite-based finite branch-and-bound. *Computational Optimization and Applications*, 43(2):181–195, 2009.

[29] Samuel Burer. On the copositive representation of binary and continuous nonconvex quadratic programs. *Mathematical Programming*, 120(2):479–495, 2009.

[30] Samuel Burer. Copositive programming. In *Handbook on semidefinite, conic and polynomial optimization*, pages 201–218. Springer, 2012.

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

[32] J. Chen and S. Burer. Globally solving nonconvex quadratic programming problems via completely positive programming. *Mathematical Programming Computation*, 4(1):33–52, 2012.

[33] Jieqiu Chen and Samuel Burer. Globally solving nonconvex quadratic programming problems via completely positive programming. *Mathematical Programming Computation*, 4(1):33–52, 2012.

[34] IBM ILOG CPLEX. 12.2 user's manual. *ILOG. See ftp://ftp. software. ibm. com/software/websphere/ilog/docs/optimization/cplex/ps_ usrmancplex. pdf*, 2010.

[35] Etienne De Klerk and Dmitrii V Pasechnik. Approximation of the stability number of a graph via copositive programming. *SIAM Journal on Optimization*, 12(4):875–892, 2002.

[36] Etienne de Klerk and Dmitrii V Pasechnik. A linear programming reformulation of the standard quadratic optimization problem. *Journal of Global Optimization*, 37(1):75–84, 2007.

[37] Peter JC Dickinson. Geometry of the copositive and completely positive cones. *Journal of Mathematical Analysis and Applications*, 380(1):377–395, 2011.

[38] C. Dobre and J. C. Vera. Exploiting symmetry in copositive programs via semidefinite hierarchies. *Math. Programming*, 151(2):659–680, 2015.

[39] H. Dong. Symmetric tensor approximation hierarchies for the completely positive cone. *SIAM Journal on Optimization*, 23(3):1850–1866, 2013.

[40] H. B. Dong and K. Anstreicher. Separating doubly nonnegative and completely positive matrices. *Math. Programming*, 137(1-2):131–153, 2013.

[41] Michael J Duffy, Sean R Wakayama, and Ryan Hupp. A study in reducing the cost of vertical flight with electric propulsion. In *17th AIAA Aviation Technology, Integration, and Operations Conference*, page 3442, 2017.

[42] M. Dür. Copositive programming – a survey. In M. Diehl, F. Glineur, E. Jarlebring, and

W. Michiels, editors, *Recent Advances in Optimization and its Applications in Engineering*, pages 3–20. Springer, 2010.

[43] Mirjam Dür. Copositive programming–a survey. In *Recent advances in optimization and its applications in engineering*, pages 3–20. Springer, 2010.

[44] RODRIGO G Eustaquio, ELIZABETH W Karas, and ADEMIR A Ribeiro. Constraint qualifications for nonlinear programming. *Federal University of Parana*, 2008.

[45] Dominique Feillet, Pierre Dejax, Michel Gendreau, and Cyrille Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

[46] C. A. Floudas and V. Visweswaran. A global optimization algorithm (GOP) for certain classes of nonconvex NLPs–I. Theory. *Computers and chemical engineering*, 14(12):1397–1417, 1990.

[47] D. Y. Gao. Canonical duality theory and solutions to constrained nonconvex quadratic programming. *Journal of Global Optimization*, 29(4):377–399, 2004.

[48] Brian German, Matthew Daskilewicz, Thomas K Hamilton, and Matthew M Warren. Cargo delivery in by passenger evtol aircraft: A case study in the san francisco bay area. In *2018 AIAA Aerospace Sciences Meeting*, page 2006, 2018.

[49] F. Giannessi and E. Tomasin. Nonconvex quadratic programs, linear complementarity problems, and integer linear programs. In *Lecture Notes in Computer Science*, volume 3, pages 437–449. Fifth Conference on Optimization Techniques, Springer, Berlin Heidelberg New York, 1973.

[50] MA Goberna, MA López, and J Pastor. Farkas-minkowski systems in semi-infinite programming. *Applied Mathematics and Optimization*, 7(1):295–308, 1981.

[51] Miguel A Goberna. Linear semi-infinite optimization. *Mathematical Methods in Practice 2*, 1998.

[52] Miguel A Goberna and Marco A López. A comprehensive survey of linear semi-infinite optimization theory. In *Semi-infinite programming*, pages 3–27. Springer, 1998.

[53] Miguel A Goberna and Marco A Lopez. Linear semi-infinite programming theory: An updated survey. *European Journal of Operational Research*, 143(2):390–405, 2002.

[54] N. I. M. Gould, D. Orban, and P.L. Toint. CUTEr and SifDec: A constrained and unconstrained testing environment, revisited. *ACM Trans. Math. Software*, 29(4):373–394, 2003.

[55] O. Güler, A. J. Hoffman, and U. G. Rothblum. Approximations to solutions to systems of linear inequalities. *SIAM J. Matrix Anal. Appl.*, 16(2):688–696, 1995.

[56] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2017.

[57] SÅ Gustafson and KO Kortanek. Numerical treatment of a class of semi-infinite programming problems. *Naval Research Logistics Quarterly*, 20(3):477–504, 1973.

[58] Pierre Hansen, Brigitte Jaumard, Michèle Ruiz, and Junjie Xiong. Global minimization of indefinite quadratic functions subject to box constraints. *Naval Research Logistics (NRL)*, 40(3):373–392, 1993.

[59] J-B Hiriart-Urruty and Alberto Seeger. A variational approach to copositive matrices. *SIAM review*, 52(4):593–629, 2010.

[60] A. J. Hoffman. On approximate solutions of systems of linear inequalities. *Journal of Research of the Natural Bureau of Standards*, 49(4), 1952.

[61] Jeff Holden and Nikhil Goel. Fast-forwarding to a future of on-demand urban air transportation. *San Francisco, CA*, 2016.

[62] R. Horst, P. M. Pardalos, and N.V. Thoai. *Introduction to Global Optimization.* Dortrecht: Kluwer, 2nd edition, 2000.

[63] J. Hu, J. E. Mitchell, J. S. Pang, and B. Yu. On linear programs with linear complementarity constraints. *Journal of Global Optimization*, 53(1):29–51, 2012.

[64] S. Kim and M. Kojima. Second order cone programming relaxation of nonconvex quadratic optimization problems. *Optimization Methods and Software*, 15(3-4):201–224, 2001.

[65] S. Kim and M. Kojima. Exact solutions of some nonconvex quadratic optimization problems via SDP and SOCP relaxations. *Computational Optimization and Applications*, 26(2):143–154, 2003.

[66] Andreĭ Nikolaevich Kolmogorov and Sergeĭ Vasil'evich Fomin. *Introductory real analysis.* Courier Corporation, 2012.

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

[68] J.B. Lasserre. Moments and sums of squares for polynomial optimization and related problems. *Journal of Global Optimization*, 45(1):39–61, 2009.

[69] J.B. Lasserre. *Moments, positive polynomials and their applications*, volume 1. World Scientific, 2009.

[70] M. Laurent. Sums of squares, moment matrices and optimization over polynomials. In *Emerging applications of algebraic geometry*, pages 157–270. Springer, 2009.

[71] Z. Luo, L. Qi, and Y. Ye. Linear operators and positive semidefiniteness of symmetric tensor spaces. *Science China Mathematics*, 58(1):197–212, 2015.

[72] O. L. Mangasarian. A condition number for linear inequalities and linear programs. Technical Report 2231, MRC Technical Summary Report, 1981.

[73] Ruth Misener and Christodoulos A Floudas. Glomiqo: Global mixed-integer quadratic optimizer. *Journal of Global Optimization*, 57(1):3–50, 2013.

[74] T. S. Motzkin and E. G. Straus. Maxima for graphs and a new proof of a theorem of Turán. *Canad. J. Math*, 17(4):533–540, 1965.

[75] Katta G Murty and Santosh N Kabadi. Some np-complete problems in quadratic and nonlinear programming. *Mathematical programming*, 39(2):117–129, 1987.

[76] Y. Nesterov. Semidefinite relaxation and nonconvex quadratic optimization. *Optimization methods and software*, 9(1-3):141–160, 1998.

[77] Victoria C Nneji, Mary L Cummings, Alexander J Stimpson, and Kenneth H Goodrich. Functional requirements for remotely managing fleets of on-demand passenger aircraft. In *2018 AIAA Aerospace Sciences Meeting*, page 2007, 2018.

[78] P. M. Pardalos and S. A. Vavasis. Quadratic programming with one negative eigenvalue is NP-hard. *Journal of Global Optimization*, 1(1):15–22, 1991.

[79] Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization.* PhD thesis, California Institute of Technology, 2000.

[80] J. Peña, J. Vera, and L.F. Zuluaga. Completely positive reformulations for polynomial optimization. *Mathematical Programming*, 151(2):405–431, 2015.

[81] Javier Peña, Juan C. Vera, and Luis F. Zuluaga. Hoffman bounds and norms of set-valued mappings. Technical report, Carnegie Mellon University, 2017.

[82] Priyank Pradeep and Peng Wei. Energy efficient arrival with rta constraint for urban evtol operations. In *2018 AIAA Aerospace Sciences Meeting*, page 2008, 2018.

[83] Rubén Puente and Virginia N Vera de Serio. Locally farkas-minkowski linear inequality systems. *Top*, 7(1):103–121, 1999.

[84] L. Qi, C. Xu, and Y. Xu. Nonnegative tensor factorization, completely positive tensors, and a hierarchical elimination algorithm. *SIAM Journal on Matrix Analysis and Applications*, 35(4):1227–1241, 2014.

[85] Rembert Reemtsen and Stephan Görner. Numerical methods for semi-infinite programming: a survey. In *Semi-infinite programming*, pages 195–275. Springer, 1998.

[86] J. Renegar. *A Mathematical View of Interior-Point Methods in Convex Optimization*, volume 3 of *MPS/SIAM Series on Optimization.* SIAM, 2001.

[87] Ralph Tyrell Rockafellar. *Convex analysis.* Princeton university press, 2015.

[88] R.T. Rockafellar and R.J-B. Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

[89] N. V. Sahinidis. BARON: a general purpose global optimization software package. *Journal of Global Optimization*, 8(2):201–205, 1996.

[90] Andrea Scozzari and Fabio Tardella. A clique algorithm for standard quadratic programming. *Discrete Applied Mathematics*, 156(13):2439–2448, 2008.

[91] H. Sherali and W. Adams. A hierarchy of relaxations for mixed-integer zero-one programming problems. *Discrete Applied Mathematics*, 52:83–106, 1994.

[92] S Skiena. Dijkstra?s algorithm. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley*, pages 225–227, 1990.

[93] M. Tawarmalani and N. V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Math. Programming*, 99:563–591, 2004.

[94] R. J. Vanderbei and D. F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13(1-3):231–252, 1999.

[95] Wei Xia, Juan Vera, and Luis F Zuluaga. Globally solving non-convex quadratic programs via linear integer programming techniques. *to appear in INFORMS Journel of Computing*, 2015.

[96] Wei Xia and Luis F Zuluaga. Completely positive reformulations of polynomial optimization problems with linear constraints. *Optimization Letters*, pages 1–13, 2017.

[97] X. Y. Zheng and K. F. Ng. Hoffman's least error bounds for systems of linear inequalities. *Journal of Global Optimization*, 30(4):391–403, 2004.

[98] Günter M Ziegler. *Lectures on polytopes*, volume 152. Springer Science & Business Media, 2012.

Biography

Wei Xia was born in Beijing, China in 1990. He received his Bachelor of science degree in mathematics and computer science from College of William and Mary in 2013. He joined the Industrial and Systems Engineering program at Lehigh University in the same year. He was the president of the INFORMS student chapter of Lehigh University in 2016. Wei will be joining Zillow as an operations research scientist in 2019.