Polynomial and Indefinite Quadratic Programming Problems: Algorithms and Applications

by

Cihan Tuncbilek

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Industrial and Systems Engineering

APPROVED:

[illegible]

Dr. Hanif D. Sherali, Chairman

Faiz Al-Khayyal

Dr. Faiz A. Al-Khayyal

[illegible]

Dr. C. Patrick Koelling

Subhash C. Sarin

Dr. Subhash C. Sarin

Antonio Trani

Dr. Antonio A. Trani

November 29, 1994

Blacksburg, Virginia

Polynomial and Indefinite Quadratic Programming Problems:

Algorithms and Applications

by

Cihan Tuncbilek

Dr. Hanif D. Sherali, Chairman

Industrial and Systems Engineering

(ABSTRACT)

This dissertation is concerned with the global optimization of polynomial programming problems, and a detailed treatment of its particular special case, the indefinite quadratic programming problem. Polynomial programming problems are generally nonconvex, involving the optimization of a polynomial objective function over a compact feasible region defined in terms of polynomial constraints. These problems arise in a variety of applications in the context of engineering design and chemical process design situations. Despite the wide applicability of these classes of problems, there is a significant gap between the theory and practice in this field, principally due to the challenge faced in solving these rather difficult problems. The purpose of this dissertation is to introduce new solution strategies that assist in closing this gap.

For solving polynomial programming problems, we present a branch and bound algorithm that uses a Reformulation Linearization Technique (RLT) to generate tight linear programming relaxations. This bounding scheme involves an automatic reformulation of the problem via the addition of certain nonlinear implied constraints that are generated by using the products of the simple bounding restrictions and, optionally, products involving the structural constraints. Subsequently, in a linearization phase, each distinct nonlinear term in the resulting problem is replaced by a new variable to obtain a linear program. The underlying purpose of this procedure is to closely approximate the convex envelope of the objective function over the convex hull of the feasible region. Various implementation issues regarding the derivation of such a bounding problem using the RLT, and the dominance of such bounds over existing alternative schemes, are investigated, both theoretically and computationally. The principal thrust of the proposed method is to construct a tight linear programming relaxation of the problem via an appropriate RLT procedure, and to use this in concert with a suitable partitioning strategy, in order to derive a practically effective algorithm that is theoretically guaranteed to be globally convergent. To address various implementation issues, empirical experiments are conducted using several test problems from the literature, many motivated by the aforementioned practical applications. Our results on solving several such practical engineering design problems demonstrate the viability of the proposed approach.

This approach is also specialized and further enhanced for the particular class of indefinite (and concave) quadratic programming problems. These problems are important in their own right, and arise in many applications such as in modelling economies of scale in a cost structure, in location-allocation problems, several production planning and risk management problems, and in various other mathematical models such as the maximum clique problem and the jointly constrained bilinear programming problem. The proposed algorithm is more than just a specialization of the polynomial programming approach; it involves new, nontrivial extensions that exploit the particular special structure of the problem, along with many additional supporting features that improve the computational efficiency of the procedure. Certain types of nonlinearities are also retained and handled implicitly within the bounding problem to obtain sharper bounds. Computational results are presented on a set of test problems from the literature to demonstrate the efficiency of the approach. (One of these test problems had not previously been solved to optimality.) It is shown that for many problems, including randomly generated problems having upto 50 variables, the initial relaxation itself produces an optimal or a near optimal solution. This is significant in that the proposed methodology affords an approach whereby such hard nonconvex problems can be practically solved via a single or a few higher dimensional linear programming problems.

Acknowledgements

I would like to thank my advisor, Dr. Hanif D. Sherali, for his patience and input at each stage of this work. Without his support, this dissertation would not have been completed. I would also like to thank Dr. Faiz A. Al-Khayyal, Dr. C. Patrick Koelling, Dr. Subhash C. Sarin, and Dr. Antonio A. Trani for serving on the dissertation committee and for their helpful suggestions.

I would finally like to acknowledge the National Science Foundation for their support under Grant No. DMII-9121419.

Table of Contents

List of Illustrations

List of Tables

Chapter I

Introduction

This research is concerned with the development of global optimization algorithms for solving polynomial programming problems, and its particular special case, the indefinite quadratic programming problem. This type of problem arises in a variety of applications concerned with engineering design, such as the design of flywheels, transformers, heat exchangers, and membrane separators, chemical process design situations as in distillation column sequencing, the design of alkylation process, pooling/blending, and reactor/separator/recycle system problems, and in VLSI design, to mention a few. Despite the wide applicability of these classes of problems, there is a significant gap between the theory and practice in this field, principally due to the challenge faced in solving these rather difficult problems. The purpose of this dissertation is to introduce new solution strategies that will assist in closing this gap.

1.1 Problem Description

A polynomial programming problem, as its name would reveal, involves the minimization or maximization of a polynomial objective function subject to polynomial equality or inequality constraints, in addition to operational lower and upper bounds on the variables. In order not to cloud the reader's view, we postpone the formal mathematical formulation of this problem, which involves the introduction of a particular notation in describing polynomial functions, to a later chapter. However, at this point, we can envisage polynomial programs as being related to the following more familiar class of signomial geometric programming problems.

Geometric programming originated as a solution method for engineering design problems involving functions of the type $\Sigma_{i\in I} c_i \Pi_{j\in J} t_j^{a_{ij}}$ with design variables $t_j > 0$, $j\in J$, and arbitrary constants c_i, $i\in I$, and a_{ij}, $i\in I$, $j\in J$. Minimization problems where all the coefficients c_i are positive and all constraints are inequalities of less than or equal to type are the well known posynomial geometric programs, which can be transformed to convex programs by variable substitution. In signomial geometric programming, the coefficients c_i are allowed to be negative. In polynomial programming we also allow the coefficients c_i to have mixed signs, but in addition, the design variables are now permitted to *equal* zero as opposed to being strictly positive, while the exponents a_{ij} are restricted to be nonnegative integers.

Polynomial programming subsumes a large class of global nonconvex optimization problems. General global optimization has attracted substantial research interest in the last fifteen years, and recently a separate technical journal dedicated to this area has emerged. Although theoretically convergent algorithms have been proposed for several classes of problems in recent years, the difficulty in recognizing a global optimal solution from local information using well-established differential calculus based optimization theory has made global nonconvex optimization problems computationally notorious. To gain some computational advantage in the attack on these foreboding problems, the special structure inherent in different subclasses of nonconvex problems has had to be exploited, leading to the development of specialized algorithms for various classes of such problems.

In this research, we not only present a general algorithm for nonconvex polynomial programming problems, but we also specialize this methodology to solve indefinite quadratic programming problems. This problem involves the minimization/maximization of an indefinite quadratic function subject to linear constraints. We assume that the feasible region is closed and bounded, i.e., it is compact. A mathematical description of the indefinite quadratic program is as follows:

$$\begin{aligned} \text{QP}: \quad & \text{minimize} \quad cx + x^tQx \\ & \text{subject to} \quad Ax \leq b \\ & \qquad 0 \leq l_k \leq x_k \leq u_k \leq +\infty \qquad k = 1,\dots,n \end{aligned}$$

where $c \in \mathbb{R}^n$, A is an $m \times n$ real matrix, $b \in \mathbb{R}^m$, Q is an $n \times n$ symmetric indefinite matrix, and where the decision variables are $x \in \mathbb{R}^n$. We remark here that our specialization of the polynomial programming approach to solving indefinite quadratic programs is so substantial that similarities exist between the two approaches only in a broad conceptual framework. Hence, we will actually present our approach for the indefinite quadratic programs first, and then *generalize* this, losing a great deal of applicable tricks, to the class of polynomial programming problems.

Another problem that is a very close relative of the indefinite quadratic program is the concave quadratic program, where the objective function is concave implying that Q is negative semidefinite. Strictly speaking, concave quadratic programming is not a special case of indefinite quadratic programming, however, our algorithm will treat it as if it is just another instance of indefinite quadratic programming.

Since, in general for nonconvex problems, there may exist several local minima, and the main difficulty is that the local optimality criteria do not convey much about global optimality, one approach is to partition the feasible region and examine each partition separately. As a widely used tool in combinatorial optimization, the branch and bound method effectively implements such a divide and conquer strategy. The algorithm we propose will also be of this kind. Branch and bound is not a new concept in the nonconvex optimization area. In fact, Horst (1990) has given conditions under which a prototype branch and bound algorithm for nonconvex programming problems is convergent. However, rather than a ready-to-use algorithm, this is a framework for developing convergent algorithms, and constructing a bounding scheme seems to be its most important component.

Our bounding scheme involves including nonlinear implied constraints in the original problem, and subsequently defining a new variable for each distinct nonlinear term in the augmented problem to obtain a lower bounding linear program. (In some cases, as we shall see later, retaining certain types of nonlinearities within the bounding problems is also possible, without hampering the efficiency of solving such problems in comparison with their linear counterparts.) The nonlinear implied constraints used here are automatically generated by taking pairwise products of the bound factors $(u_k - x_k) \geq 0$, $(x_k - l_k) \geq 0$, $k = 1, \ldots, n$, and, optionally, inter-products of bound and constraint factors, the latter being of the form $(b_i - a_i x) \geq 0$, $i = 1, \ldots, m$, for the problem QP. This process is known as a Reformulation Linearization Technique (RLT). This technique has been introduced in the context of integer and mixed programming to hierarchically obtain the convex hull of the feasible region by Sherali and Adams (1990,1994). RLT has also been subsequently extended for solving bilinear programming problems by Sherali and Alameddine (1990), and for location-allocation problems by Sherali and Tuncbilek (1992a) and Sherali, Ramachandran and Kim (1993).

1.2 Applications

Polynomial programming problems arise in various applications, such as engineering design, chemical engineering, and VLSI design. The design problems include the flywheel design (Siddall (1972)), process design (Colville (1968)), P.O. box design (Rosenbrock (1960)) and transformer design (Bartholomew-Biggs (1976)). For example, a flywheel absorbs energy when a vehicle is coasting and feeds it back to the drive when power is required. The design objective is to maximize the stored energy subject to a maximum allowable stress, where the decision variables are the diameter, width, and rotational speed of the flywheel. This problem has a degree seven polynomial objective function. Another design problem involves heating a cold fluid by passing it through three heat exchangers where hot fluids flow at the same rate as the cold fluid. The objective of the heat exchanger design problem is to minimize the total heat transfer area of the three heat exchangers (Avriel and Williams (1971)). This problem involves bilinear constraints. The 3-stage membrane separation problem is a representative of a large number of chemical flow processes with recycle streams, and its formulation involves bilinear constraints (Dembo (1976)). These problems are included in Appendix 1 to be used as test problems for our proposed algorithm.

Floudas and Pardalos (1990) present other applications of polynomial programming in chemical processes; for example the distillation column sequencing problem, pooling/blending problems, and the reactor/separator/recycle systems problems. Distillation column sequencing problems involve separating a single multicomponent feed stream of known conditions into a number of multicomponent products of specified composition. The objective is to minimize the total annual cost by selecting an optimal distillation sequence. Nonconvexities in the formulation result from the bilinear terms describing the flowrates and the composition of the streams in the objective function and in some of the constraints. In refinery and petrochemical processing problems, in addition to the product flows, it is often necessary to model the properties of streams, where nonconvex relationships are introduced in certain pooling and blending procedures. Reactor/separator/recycling systems involve the extraction of desired products by a sequence of separation tasks following a reaction system, and then recycling unconverted reactants back to the reactor system. Given a specified number of reactors and separators, a nonconvex programming model has been proposed by Floudas and Pardalos (1990) to determine the reactor network, the separator network, and the interaction between them in order to optimize a given performance measure. This problem is of polynomial degree two.

Another application area presented in Floudas and Pardalos (1990) is the VLSI design problems. Certain aspects of physical chip design can be formulated as nonconvex programs, often with quadratic objective function and constraints. The design objective is to minimize the chip area where the constraints result from geometric design rules, from distance and connectivity requirements between various components of the circuit, and possibly from user specified constraints.

We can also express some other problem forms involving generalized polynomial terms as polynomial programs. Hansen and Jaumard (1992) showed that it is possible to handle some terms with negative or fractional exponents by defining new variables. For example, the gravel box design problem is a classical introductory example for geometric programming as given by Duffin, Peterson and Zener (1967), with an objective function of the form $40/x_1x_2x_3 + 20x_2x_3 + 40x_1x_2 + 10x_1x_3$. The complicating fractional term, which relates the total amount of gravel to the volume of the box, can be handled by defining a new variable $y = 1/x_1x_2x_3$, and including the constraint $yx_1x_2x_3 = 1$ in the problem. As another example, a formulation of the alkylation process by Bracken and McCormick (1968), which involves some variables having negative exponents, is reduced to a polynomial program in Problem 29 of Appendix 1. In other complicating cases, problems having variables with fractional exponents can be reduced to polynomial programs by noting that $y = x^{p/q}$ is equivalent to $x^p = y^q$, where p and q are pairwise prime. The drawback of this technique is the increased degree of the problem; nevertheless, we still consider such a case in Problem 31 of Appendix 1. Hansen and Jaumard also noted that, provided convergence conditions are satisfied, trigonometric and transcendental functions can be polynomially approximated by their expansions in MacLaurin series.

The maximum clique problem consists of finding the largest complete subgraph of a given undirected graph $G(V, E)$. That is, if C is a clique of G defined via a set of vertices of G, then for all pairs of vertices $v_i, v_j \in C$, the edge $(v_i, v_j) \in E$. Moreover, if C is a maximum clique of G, then it is a clique having a maximum number of vertices. The maximum clique problem is equivalent to the maximum vertex packing and the minimum vertex covering problems. Pardalos and Phillips (1990) stated this problem as an indefinite quadratic problem (see Problem 32 of Appendix 1).

Concave quadratic functions can be used to model economies of scale in the cost structure of, for example, production planning problems. For some other activities, diseconomies of scale can be more relevant, resulting in a cost function having both convex and concave components. If the cost function is quadratic, then this is an indefinite quadratic program. A similar objective function structure can occur in profit maximization for an investment planning problem (see Tuy, 1987).

Several optimization problems, like linear and quadratic 0-1 programming, quadratic assignment, 3-dimensional assignment, and bilinear programming problems, can be reduced to concave quadratic minimization problem (for details and related refences, see Pardalos and Rosen, 1987). Hence, the vast number of applications for these problems are approachable via specializations of solution techniques for the concave programming problem.

1.3 Complexity Results

A special case of the general quadratic programming problem arises when Q is positive semidefinite, which implies that QP is a convex quadratic program. In this case, every local optimum is also a global optimum, and the problem can be solved polynomially. The first polynomial-time algorithm to solve convex quadratic programs appears in Kozlov, Tarasov and Hacijan (1979), and subsequently, other efficient interior point algorithms have been proposed as in Kapoor and Vaidya

(1986), Ye and Tse (1989), and Adler and Monteiro (1991). At the other extreme, if Q is negative semidefinite, then the objective function is concave, and Sahni (1974) has shown that this class of quadratic programs is NP-hard. In fact, Pardalos and Vavasis (1991) showed that if even one of the eigenvalues of Q is negative, then QP is NP-hard. Furthermore, even simpler versions of nonconvex quadratic programming problems where the feasible region is a box/hyperrectangle, or the constraints define a simplex, are shown to be NP-hard by Vavasis (1991). Moreover, Vavasis (1990) proved that the decision version of general quadratic programming is in NP. Therefore, general quadratic programming is NP-complete.

The above NP-hardness results are stated with respect to determining global optimal solutions. However, one might expect that ascertaining local optimality is "easier." Counter to this intuition, Murty and Kabati (1987), and Pardalos and Schnitger (1988) have shown that dropping the compactness property of the feasible region, the complexity of checking local optimality for a given feasible point is NP-hard. In the case of a compact feasible region, the complexity of checking local optimality is an open question. However, as noted by Pardalos and Vavasis (1992), these results do not rule out the possibility of finding some particular points for which local optimality can be easily verified. A comprehensive treatment of the complexity issues related to quadratic programming can be found in the recent book by Vavasis (1991).

Since general quadratic programming is just an instance of polynomial programming, all the above results apply to polynomial programming problems as well.

1.4 Research Organization

In the next chapter, we review the literature on nonconvex quadratic programming and on polynomial programming. Chapters 3 and 4 present a branch and bound algorithm for solving indefinite quadratic programming problems. The development of an RLT based bounding problem is discussed in Chapter 3, while Chapter 4 focuses on various implementation issues related to the proposed algorithm. Following this, Chapter 5 introduces the general polynomial programming formulation, and presents a branch and bound algorithm to solve this problem. A convergence proof and other relevant results are also included in this chapter. In Chapter 6, two different approaches are discussed for the construction of an RLT bounding problem for polynomial programming problems, and these are theoretically and practically compared. Various classes of additional RLT constraints are also developed and compared separately for one-dimensional and multi-dimensional polynomial programming problems. In Chapter 7, we address several issues regarding the implementation of the overall branch-and-bound algorithm, and solve test problems from the literature. Finally, Chapter 8 concludes the discussion, and suggests related avenues for future research.

Chapter II

Literature Review

This chapter consists of three sections; concave programming, indefinite quadratic programming, and polynomial programming. For concave programming, we concentrate more on different approaches, and summarize representative algorithms for major approaches. The literature on indefinite quadratic programming is reviewed more extensively. Although it is not stated explicitly, the presentation of summaries for the different algorithms follows an approach-oriented organization, which also somewhat reflects a chronological order. There are relatively fewer papers on polynomial programming in the general form we consider. As a result, along with brief reviews on special cases of polynomial programming problems and on some interesting approaches to more general problems, we explain recent papers on this subject in greater detail.

A substantial amount of research has been conducted on nondeterministic methods for solving global optimization problems. Although it does not directly affect our deterministic approach, we will present a brief review with some seminal references on nondeterministic approaches as well.

2.1 Concave Programming

Pure concave minimization problems have attracted special attention over the last two decades, and a substantial amount of literature has accumulated on this subject. Here, our objective is to summarize the basic concepts and major approaches to solve this problem. The problem itself may be stated as follows, requiring the minimization of a concave function $f(x)$ over a nonempty, compact poyhedral set D.

$$\text{CP :}\quad \begin{array}{ll} \text{Minimize} & f(x) \\ \text{subject to} & x \in D \subseteq R^n \end{array} \tag{2.1}$$

It is well known that the global optimum is attained at an extreme point of D. Many algorithms exploit this property of concave minimization problems. One approach for these problems is based on enumerating the extreme points of D. The objective of the enumerative methods by ranking the extreme points is to find the global minimum, hopefully before totally enumerated the extreme points of D. In a computational survey of the extreme point ranking algorithms, McKeown (1978) observed that as the nonlinearity of the objective function increases, the efficiency of the algorithms decreases. He also noted that the computational efficiency of algorithms of this type is very adversely affected by degeneracy.

Partitioning of the feasible region, and the use of outer approximations, have become two major approaches for solving concave minimization problems. In both approaches, some bounding method is employed to find a lower bound for $f(x)$ over a given region. One such bounding method, often used implicitly, is a direct result of the extreme point optimality property of the problem CP. Let v_i, $i = 1,\ldots, m$, be the vertices of a bounded polyhedron $S \subseteq R^n$ that contains the feasible region, i.e., $S \supseteq D$. Since any $x \in D$ can be expressed as a convex combination of v_i, $i = 1,\ldots, m$, we obtain

$$\underset{i=1,\ldots,m}{\text{minimum}} \; f(v_i) \leq f(x), \quad \forall x \in D. \tag{2.2}$$

A second bounding method is based on the convex envelope concept. Let $L(x)$ be the convex envelope of $f(x)$ over S. Then, $L(x)$ is the tightest convex underestimator of $f(x)$ over S, and their values match at the vertices of S. Therefore, $\min\{L(x) : x \in S\} = \min\{f(x) : x \in S\}$. $L(x)$ can be expressed as

$$L(x) = \min\left\{ \sum_{i=1}^{m} \alpha_i f(v_i) \; : \; \sum_{i=1}^{m} \alpha_i v_i = x, \; \sum_{i=1}^{m} \alpha_i = 1, \; \alpha_i \geq 0, \; i = 1,\ldots, m \right\}. \tag{2.3}$$

Note that $L(v_i) = f(v_i)$, $i = 1,\ldots, m$. If S is a simplex, then $m = n + 1$, and $L(x) = a^t x + b$, $a \in R^n$, $b \in R^1$, is a linear function, where (a, b) is the unique solution of the $n + 1$ linear equations, $a^t v_i + b = f(v_i)$, $i = 1,\ldots., n + 1$.

In partitioning approaches the feasible region is usually contained in a region with a simple structure, for example, a cone, a simplex or a rectangular region, which can be used conveniently in partitioning and bounding procedures. Such a method was first used by Tuy (1964) in a cone splitting algorithm to solve problem CP. Here, an initial cone $M_0 \supset D$ is constructed by extending the rays from a local minimum corner point $x_0 \in D$ through n neighboring vertices of x_0 (nondegeneracy is assumed). On each ray $i \in \{1,\ldots, n\}$ the most remote point y_i from x_0

is located such that $f(x) \geq f(\hat{x})$ for all x on the line from x_0 to y_i, where $\hat{x}$ is the current best solution. At the initial step, $\hat{x}$ is taken as x_0. If $f(x) \geq f(\hat{x})$ for all points along the ray i, then y_i is selected at a sufficiently large distance from x_0. The points y_i, $i = 1, \ldots, n$ define a cutting plane C_0. Notice that if C_0 does not trim off D then x_0 solves the problem.

To show how this cut is used in a convergent algorithm, we describe Thoai and Tuy's (1980) branch and bound cone splitting algorithm. Let T_0 be the hyperplane passing through the n neighboring vertices of the apex of the initial cone M_0, and let C_k be a Tuy cut generated for the subcone $M_k \subseteq M_0$, where M_k, and M_0 share the common apex. If C_k trims off $D \bigcap M_k$ then C_k is moved farther away from the apex of M_k, until it no longer cuts $D \bigcap M_k$. Now, the first bounding method can be used to give a lower bound $\mu(M_k)$. If $\mu(M_k) \geq f(\hat{x})$, then the subcone M_k is eliminated. Otherwise, M_k is split by bisecting the longest edge of the $(n-1)$-simplex spanned by the points where the edges of M_k meet T_0.

Considering the general nonconvex programming problem, where $f(x)$ is continuous, and D is a compact set, Horst (1976) presents a theoretical branch and bound algorithmic framework, which is based on a partitioning of the feasible region using general compact partitions. When this algorithm is applied to a concave programming problem having a compact, convex feasible region, simplex partitions and the convex envelope based bounding method are used. At any stage in this process, a lower bound is calculated over $S \bigcap D$, for each simplex partition S such that $S \bigcap D \neq \emptyset$. Suppose, $\mu_k(S_k)$ is the smallest lower bound attained at $x_k \in S_k \bigcap D$. If $\mu_k(x_k) = f(x_k)$, the algorithm stops. Otherwise, S_k is partitioned into two subsimplices. Let $[v_p, v_q]$, v_p, $v_q \in V_k$, be the longest edge of S_k, where V_k is the set of vertices of S_k, and let x_l be the midpoint of $[v_p, v_q]$. Sets of the vertices of the two subsimplices are given by $(V_k - \{v_p\}) \bigcup \{x_l\}$, and

$(V_k - \{v_q\}) \bigcup \{x_l\}$. If $\{x = (x_1, \ldots, x_n) \in R^n : x_i \geq 0,\ i = 1, \ldots, n\} \subseteq D$ then the initial simplex that contains D can be determined by the points 0 (the origin), and $z_0 e_i$, $i = 1, \ldots, n$, where $z_0 = \max\{\sum_{i=1}^{n} x_i : x \in D\}$, and where e_i is the i^{th} unit coordinate vector.

Rosen (1983) uses rectangular partition sets in a concave quadratic programming algorithm. A rectangular region $\bar{R} \supset D$ is constructed by moving $2n$ hyperplanes as far as possible on D from the global maximum solution along selected orthogonal directions. The special structure of $\bar{R}$ and the objective function permits the calculation of a lower bound without explicitly finding the vertices of $\bar{R}$. By using surfaces parallel to those of $\bar{R}$, another rectangular region $\hat{R}$ having a maximum volume is constructed such that $\hat{R}$ is inscribed within the ellipsoid contour of $f(x)$, at the level $f(x) = f(\hat{x})$, where $\hat{x}$ is the current best solution. Since $f(x) \geq f(\hat{x})$, $\forall x \in \hat{R}$, $\hat{R} \bigcap D$ can be eliminated. Now, the remaining feasible polytopes, each of which is cut from D by a surface of $\hat{R}$, can be handled separately. Using a similar approach, Kalantari and Rosen (1987) present another algorithm. Here, the construction of $\bar{R}$ is accomplished by solving the $2n$ linear programs, $\max_{x \in D} u_i Q x$, and $\min_{x \in D} u_i Q x$, $i = 1, \ldots, n$, where Q is the Hessian of $f(x)$, and u_i, $i = 1, \ldots, n$, are Q-conjugate directions. The convex envelope of $f(x)$ over $\bar{R}$ is linear, and a lower bound is calculated by minimizing it over D. To partition the feasible region, a hyperplane parallel to a surface of $\bar{R}$ bisects $\hat{R}$, where $\hat{R}$ is defined as above, into two parallelpipeds. For a general problem, where some y variables appear only linearly in the objective function in addition to $f(x)$, and where the feasible region is defined on the (x, y)-space, the feasible region is projected onto the x-space.

In the outer approximation approach, a region containing D is successively refined to have a tighter approximation with the motivation of solving CP before the containing region reduces to D. Falk and Hoffman (1976) take the initial containing region as a simplex. This simplex is constructed by truncating the n rays that extend from a local minimum corner point of D by a hyperplane, if possible, using a nonbinding constraint of D. At some iteration k, let S_k be the containing polytope having vertices $v_1,\ldots, v_m$. Using the convex envelope based bounding method, the lower bound $\sum_{i \in A_k} \alpha_i f(v_i)$ is obtained, where $A_k = \{i : \alpha_i > 0,\ i = 1,\ldots, m\}$. If $v_i \in D\ \forall i \in A_k$, then the algorithm stops. Otherwise, the vertex $v_p \notin D$ that contributes most to the lower bound is determined. Activating a constraint that is violated at v_p tightens the containing region. After finding the newly generated vertices, the algorithm continues with the bounding operation.

Other approaches for various types of concave minimization problems include inner approximation, transformation of CP to a bilinear program followed by the use of specialized cutting planes, and mixed integer programming approximations. For an extensive survey on this subject, see Pardalos and Rosen (1987). Above, we have summarized the approaches to solve CP and have provided greater details for the major ones, by describing how they are actually implemented in representative example algorithms. As we review the algorithms for solving the indefinite quadratic programming problems, we often encounter these approaches either at a conceptual level, or actually applied in a form close to the one described above.

2.2 *Indefinite Quadratic Programming*

Earlier work in pursuit of solving general quadratic programming problems includes that of Ritter (1966), who gave an algorithm based on Tuy-cuts for concave programming, Manas (1968), who gave an algorithm based on the enumeration of vertices of the feasible region, and Mueller (1970), who used gradient projection searches in an adjacent extreme point framework. Zwart (1973) has subsequently shown via a counter example that Ritter's algorithm may converge to a non-global optimum. Also, although Mueller menaged to solve test problems having up to 38 variables and 49 constraints, the convergence of this procedure is not guaranteed.

Hesse (1973) has developed a penalty function based heuristic search procedure for estimating a global optimum for nonconvex programming problems. Once a local minimum is found at a boundary point, unlike classical exterior point methods which stop at this point, the search continues in the infeasible region by manipulating the penalty parameter such that the contribution of the feasibility conditions to the penalty function decreases relative to the original objective function. The search in the infeasible region converges to a finite stationary point due to the multiplicative form of the penalty function. Then, the penalty for infeasibility is automatically increased to urge the search to re-enter the feasible region at a point that improves the objective function with respect to the previous local minimum. Also, two methods are proposed to cut off a local optimum in order to continue the search within the feasible region. Unfortunately, these cuts may also unknowingly delete the global optimal solution. However, this algorithm successfully solved most of the selected problems from the literature, although the convergence to optimality is not guaranteed.

Since the constraints of QP are linear, Karush-Kuhn-Tucker (KKT) conditions hold at any local minimum, and so also at the global minimum. Balas (1975) and Benacer and Pham Dinh Tao (1986) use slightly different approaches to find the best (lowest objective function value) KKT point. The primal feasibility and dual feasibility conditions of QP can be expressed as linear equalities with nonnegative variables. These conditions are augmented by an objective function based constraint, that eliminates the points which give a higher objective function value than the best known KKT point. This augmented set, say *F*, involves only linear functions, since the objective function of QP can be expressed as a linear function by using the dual variables at a KKT point. Balas constructs a convex polyhedral cone, which is called a generalized polar of *F*, whose interior contains no point from *F* that satisfies the complementary slackness condition, and at the same time, gives a lower objective function value than the current best one. In a finitely convergent algorithm, the set *F* is outer approximated by a polyhedral set using tractable number of vertices - initially a simplex. At each iteration, a sequence of linear programs are solved to check if the vertices of the outer approximating region are in the generalized polar set of *F*. If the answer is no for a vertex then it is cut off by activating a violated constraint from *F*, which results in a better outer approximation having an increased number of vertices. As better KKT points are found, the set *F*, and its generalized polar is updated. The algorithm stops when all the vertices of the outer approximating set are shown to be in the generalized polar of *F*. Benacer and Pham Dinh Tao's approach is a variant of Balas' algorithm, in which QP is reduced to a linear programming problem with one reverse convex constraint by using the KKT conditions. (A reverse convex constraint is of the form $g(x) \geq 0$ where $g(\cdot)$ is a convex function.) They relax this problem by considering a polyhedral outer approximation of the set *F*, which is represented by its vertices. An optimal solution of the relaxed problem occurs at one of these vertices, where the reverse convex inequality holds. If the optimal vertex is in *F*, the problem is solved. Otherwise, it is cut off by using a violated constraint from *F*, which therefore guarantees an additional set of new vertices.

Kough (1979) proposes an ε-finite algorithm using a generalized Benders' cut procedure to solve the problem QP in the following transformed (polar) form:

$$\begin{aligned}&\text{Minimize} \quad x^t x - y^t y \\ &\text{subject to} \quad Ax + By + c \leq 0, \quad x \in R^k, \quad y \in R^p\end{aligned} \tag{2.4}$$

(Kough originally considered a maximization problem, but to show its relation to a subsequent algorithm, we have changed it to a minimization problem). Note that QP can be reduced to this form by first eigen-transforming the problem to make the objective function separable, and then by scaling variables. For any fixed $y = y_k$, let x_k solve the convex subproblem

$$\theta(y_k) = \min\{x^t x\colon\ Ax \leq -(By_k + c)\}. \tag{2.5}$$

Then, the master problem $\min\{z\colon\ z \geq \theta(y) - y^t y,\ y \in Y_0\}$ is relaxed by using linear feasibility cuts to approximate $Y_0 \equiv \{y\colon\ y$ is feasible for some x in (2.5)$\}$, and by using quadratic Benders' cuts of the form $z \geq x_k^t x_k - y^t y + \lambda_k(Ax_k + By + c)$, where λ_k denotes a set of optimal dual multipliers for the subproblem (2.5). Let y_{k+1} solve the stage k problem. If $y_{k+1} \notin Y_0$ then a feasiblity cut is generated; otherwise, Kough developes an exact Benders' cut to tighten the master problem approximation. However these Benders' cuts are reverse convex constraints, and hence, the main difficulty is embedded in solving the master problem approximations. To make the problem more tractable, Kough suggests to minimize each Benders' cut in turn subject to the rest of the cuts. In this case, each problem becomes a concave minimization problem with linear constraints due to the cancellation of identical quadratic terms in the Benders' cuts.

Notice that QP in the form (2.4) is a dc-programming problem, where dc stands for the difference of two convex functions. After giving a general dc-programming algorithm, Tuy (1987) specializes it to solve (2.4). He first transforms (2.4) to a pure concave program with convex constraints by introducing a new variable to account for the convex part of the objective function as follows:

$$\begin{aligned}&\text{Minimize } t - y^t y\\ &\text{subject to } \theta(y) \le t,\ y \in Y_0\end{aligned} \tag{2.6}$$

where Y_0 and $\theta(y)$ are the same as defined above for Kough's algorithm. In (2.6), Y_0 is relaxed by constructing an outer approximating polytope $S_0 \supset Y_0$. Also, given a point y_0, $\theta(y) \le t$ is approximated by $\lambda_0^t B(y - y_0) + x_0^t x_0 \le t$ using a subgradient based first-order linearization of $\theta(\cdot)$ at y_0, where x_0 and λ_0 are, respectively, the optimal primal and dual solution to the subproblem (2.5) corresponding to $\theta(y_0)$. If the optimal solution (y_1, t_1) to the relaxed problem is feasible to (2.6), the problem is solved. Otherwise, if $\theta(y_1) \le t_1$ is violated, then a subgradient based cut is generated at the point y_1. If $y_1 \notin Y_0$ then a feasibility cut similar to Kough's is generated. Including the new cut in the relaxed problem, the algorithm continues with a tighter relaxation of (2.6). Tuy notes that, at each iteration of this algorithm, only one concave minimization is solved, as opposed to Kough's algorithm, where a few of them are solved at each iteration. Tuy also raises some questions about the convergence of Kough's algorithm, whereas his algorithm enjoys ε-finite convergence.

In order to be able to use previously described techniques for quadratic concave minimization, Pardalos, Glick and Rosen (1987) eigen-transform the problem to obtain a separable objective function such that all the quadratic terms appear in pure form. The transformed variables corresponding to the concave quadratic terms are called "concave variables". After obtaining simple bounds on variables, they replace each concave quadratic term with its linear convex envelope over the corresponding variable's simple bounds. For example, the convex envelope of the concave term $-x_j^2$ over $x_j \in [l_j, u_j]$ is the linear function $u_j l_j - (l_j + u_j)x_j$. The resulting problem is a convex programming problem which gives a lower bound on the global minimum value of the original problem. Also, an upper bound is obtained by evaluating the original objective function at the optimal solution of this lower bounding problem. To improve the lower bound by constructing tighter convex envelopes, they bisect the feasible interval of some concave variable term. After constructing the convex envelope over each resulting half interval, two separate convex programs are solved over the original feasible region. A half interval is eliminated if the resulting lower bound exceeds the upper bound. This procedure is repeated for each concave variable in turn. Hence, if there are k concave variables, a total of $2k$ convex programs are solved. For each eliminated half interval the remaining half interval can be bisected again. This procedure continues until there is no further elimination of intervals. If the gap between the current best lower and upper bounds on the global optimum value is not small enough , then a piecewise linear approximation of the problem over the remaining intervals is formulated as a mixed integer problem. Piecewise linear functions are generated through an interpolation of the values of each univariate concave function appearing in the objective function evaluated at adjacent grid points. Suppose that $(k+1)$ grid points divide the feasible interval $[0, \beta]$ of a concave variable x into intervals of length β/k. A new variable $w_i \in [0, 1]$, is associated with the i'th interval, denoting how much this interval contributes to the value of x. Now, x can be represented in terms of the new variables as $x = (\beta/k)\sum_{j=1}^{k} w_j$. Note that if an interval partially contributes to the value of x then the intervals before that one should be contributing fully. This condition is satisfied by the constraints $w_{j+1} \leq z_j \leq w_j$, $j = 1,\ldots, k$, which introduces Boolean z-variables into the problem. One can improve the piecewise linear approximation of the original problem by increasing the number of grid points. They report solving problems having up to 50 constraints and 330 variables, 30 of which are concave variables.

Instead of the mixed integer piecewise approximation of the problem above, Phillips and Rosen (1990) use a branch and bound approach by bisecting the remaining interval of some concave variable to partition the problem. Corresponding to each node subproblem, they start the procedure of bisecting intervals and solving $2k$ convex programs until there is no further elimination of subintervals. In their computational experience, they show that a parallel implementation of the algorithm improves the solution time, sometimes greater than the number of processors, over the sequential implementation. Computational results are presented for problems having 25-125 nonlinear variables (with the concave variables ranging from 25 to 50), and up to 400 linear variables.

For a particular definition of ε-approximate solution, Vavasis (1992) proposes an algorithm based on covering the feasible region with small sets, and then enumerating those sets by using a linear approximating function similar to the one given by Pardalos et al. (1987) above. This algorithm is polynomial in $1/\varepsilon$ for a fixed number of concave variables. For the special case of the indefinite quadratic knapsack problem, it is shown that a more efficient approximation algorithm exists by using a dynamic programming approach.

An indefinite quadratic programming problem can be equivalently transformed to a jointly constrained bilinear program by introducing a duplicate set of variables.

$$\min\{c^t x + x^t Q x \ : \ x \in S\} = \min\{c^t x + x^t Q y \ : \ x = y, \ \ x \in S, \ \ y \in S\} \tag{2.7}$$

By using graph theoretic approaches, Hansen and Jaumard (1992) improve the above transformation such that either the number of additional variables is a minimum, or the number of complicating variables is a minimum. In the literature, for the most part, bilinear programming problems are presented as being separably constrained, that is, the feasible region consists of two disjoint polyhedra associated with each of two distinct sets of variables. In this case, an optimum occurs at an extreme point, and various cutting plane type algorithms have been developed exploiting this property. Exceptions to this treatment, are the papers by Al-Khayyal and Falk (1983), and Sherali and Alameddine (1992), which consider jointly constrained bilinear programs. For a recent review on this subject, see Al-Khayyal (1990). Note that the bilinear program is an instance of the indefinite quadratic program.

Muu and Oettli (1991) consider the more general bilinear problem $\min\{p^t x + x^t My + q^t y \ : \ (x, y) \in \bar{S}\}$, where $\bar{S} \subset \mathbb{R}^n \times \mathbb{R}^m$ is a closed convex nonempty set, $p \in \mathbb{R}^n$ and $q \in \mathbb{R}^m$ are given vectors, and M is a given real $n \times m$-matrix. They propose a branch and bound algorithm using simplex bisections and a decomposition based bounding problem. This algorithm solves QP in the bilinear form (2.7). A lower bound for the problem (2.7) over a simplex B with extreme points v^i, $i = 1,..., m + 1$, is found by solving the following $m + 1$ linear programs.

$$\begin{aligned}\min\{c^t x + x^t Qy \ : \ x \in S \cap B, x = y\} &\leq \min\{c^t x + x^t Qy \ : \ x \in S \cap B, y \in B\} \\ &= \min\{ \min_i c^t x + x^t Qv^i \ : \ x \in S \cap B\} \\ &= \min_i(\min\{c^t x + x^t Qv^i \ : \ x \in S \cap B\})\end{aligned}$$

The incumbent value is updated as each linear program yields a feasible solution to the original problem, and the simplex B is fathomed if the above lower bound is greater than the incumbent value. Initially, a simplex that contains S is constructed, and if a simplex is not fathomed, two smaller simplices are created by bisecting its longest edge. This algorithm can be modified to handle nonlinear convex constraints in the above, more general form.

For jointly constrained bilinear programs, Al-Khayyal and Falk (1983) develop a branch and bound algorithm using convex envelope based underestimating problems, and a rectangular partitioning scheme. The convex envelope of the objective function can be formed by considering the convex envelope of each bilinear term separately. The convex envelope of the bilinear term $x_i x_j$ over the rectangular region $\{(x_i, x_j)\colon\ l_i \le x_i \le u_i,\ l_j \le x_j \le u_j\}$ is given by $\max\{l_j x_i + l_i x_j - l_i l_j,\ u_j x_i + u_i x_j - u_i u_j\}$. At the optimal solution to the underestimating problem, if the values of the bilinear terms and their convex envelopes match, then the original problem is solved. Otherwise, the rectangular region corresponding to the bilinear term that gives the largest gap is partitioned into four different sets of bounds by using the current optimal solution of the underestimating problem as the partitioning point. (Actually, this is not supported by their convergence theorem, which is based on a bisection approach.) Over each subrectangle, a tighter convex envelope representation of the poorly approximated bilinear term is obtained, and an underestimating subproblem is solved. This algorithm can also be used to solve biconvex problems. Later, Al-Khayyal (1992) extended this approach to solve bilinear programs having bilinear constraints by approximating the convex envelope of the bilinear functions using the convex or concave envelope of each bilinear term. The resulting underestimating problem is formulated as a linear program. Sherali and Alameddine (1992), using the RLT procedure described in the introductory section, obtain stronger bounds by solving linear bounding programs. They also embed this bounding procedure in a branch and bound algorithm using interval partitioning, and establish a more general convergence theorem that permits greater flexibility in the choice of the partitioning scheme.

In particular, it admits using the scheme based on partitioning on a single variable at the level of its value in the current relaxation, which in practice, turns out to be computationally most effective. In an accompanying paper, Sherali and Alameddine (1990) also derive explicit convex envelope characterizations for bivariate bilinear programs over a particular polytope that has no edge having finite positive slope. In this work, we generalize their algorithm.

Considering the minimization of a nonconvex quadratic function over a ball, the only variable that appears in the Hessian of the Lagrange function is the dual multiplier associated with the (quadratic) ball constraint. For some values of this dual multiplier, the Lagrange function becomes convex satisfying the second-order optimality conditions. Ye (1992) shows that any solution to the Karush-Kuhn-Tucker (KKT) first and second-order necessary optimality conditions is a global minimum having a unique optimal dual variable value, and an ε-error solution can be obtained in polynomial computational time. To obtain a solution that satisfies the first and second-order KKT optimality conditions for minimizing a general quadratic function over a polytope, Ye proposes an interior point affine scaling algorithm, that uses ball constrained quadratic minimization subproblems.

To maximize an indefinite quadratic function, where each variable is restricted to be -1 or 1, Kamath and Karmarkar (1992) obtain upper bounds from the continuous relaxation of the problem. Considering the ball $x^t x \leq n$, which encloses the box constraints $x \in [-1, 1]^n$, an upper bound on the objective function $x^t Q x$ can be obtained by using the relationship $x^t Q x \leq (x^t x)\lambda_{\max} \leq n\lambda_{\max}$, where $\lambda_{\max}$ is the maximum eigenvalue of Q. They extend this result to ellipsoids, and propose an iterative procedure that encloses the box constraints in an ellipsoid and changes the shape of the ellipsoid at each iteration to obtain better bounds.

For mixed integer quadratic programs, Al-Khayyal and Larson (1990) developed two forms of piecewise affine convex underestimating functions for linearizing the objective function in a branch-and-bound context.

Based on the convexification of the Hessian of the objective function of QP, Neumaier (1992) show that the global optimality of a strong local optimizer can be checked by solving a linear program. However, this method seems more likely to work for nearly convex problems. Sufficient global optimality conditions for concave quadratic programming have been developed by Hiriart-Urruty and Lemarechal (1990) and Hiriart-Urruty (1989) based on ε-subdifferential calculus. These conditions are reduced to algorithmically more manageable forms by Danninger and Bomze (1993), and further extended for indefinite quadratic programs by Bomze (1992). These conditions involve checking copositivity of symmetric matrices over polyhedral cones at a local optimal solution. (A symmetric matrix is Γ-copositive, if it is positive semi-definite over the cone Γ.) For detecting copositivity, Bomze and Danninger (1993) develop a recursive procedure to which the complexity of the problem is now shifted. If a given local optimal point fails to satisfy the copositivity condition, then this procedure returns a direction that is used to escape from the current point to a better solution. Computational results on this recently proposed algorithm are to be reported in the future, but checking for copositivity is expected to be a principal bottleneck.

For further information, recent reviews on concave and indefinite quadratic programming can be found in Pardalos and Rosen (1987), Pardalos (1988, 1991).

2.3 Polynomial Programming

There are not many deterministic algorithms in the literature to solve polynomial programming problems in its general form. As being notable exceptions, we will explain recent works of Floudas and Visweswaran (1990, 1993), and Shor (1990) in detail, after summarizing works addressing some particular instances of polynomial programming. We also outline some other interesting approaches dealing with more general problems.

Wingo (1985) proposes a derivative free method to minimize a univariate polynomial function over a compact interval of the real line. The idea of this algorithm is to locally approximate the n^{th} degree polynomial objective function by its $(n-1)^{th}$ degree Taylor series expansion, where the error can be easily computed since the n^{th} derivative of the objective function is constant. Starting from one end of the feasible interval, by taking small enough steps such that the magnitude of error does not exceed the ε-optimality criterion, the algorithm scans the entire feasible interval.

Another univariate polynomial minimization algorithm is developed by Bromberg and Chang (1992). This algorithm uses the level sets of the linear underestimating functions to reduce the feasible interval, in order to find an ε-global minimum solution. Having a local minimizer point m, the original interval $[a, b]$ is partitioned into two intervals $[a, m]$ and $[m, b]$, over each of which a separate linear underestimator is constructed such that it coincides with the original objective function $f(\cdot)$ at point m. Now, considering the interval $[m, b]$ for example, the intersection point x^1 of the linear underestimator and the constant function $f(m) - \varepsilon$ is found. If $f(x^1) > f(m)$, we know that the ε-global minimum cannot occur at the interval $[m, x^1]$. As the region of interest reduces to $[x^1, b]$, a new linear underestimator is constructed at point x^1. On the other hand, if $f(x^1) \leq f(m)$, then starting from x^1, a new local minimum point $\bar{x}$ is found, and the interval is reduced to $[\bar{x}, b]$. The algorithm continues until all the subsets of the original interval are eliminated. Bromberg and Chang also suggest some methods to construct finite slope linear underestimators for dc-functions. This is useful in that all functions defined over compact sets and having continuous second derivatives can be expressed as dc-functions. Wang and Chang (1994) improve the above linear underestimators for extended factorable functions that can be recursively expressed in terms of sums, products, or composites of simpler (variable reduced) functions, including min/max operations. They also present comparative computational results using problems and available algorithms from the literature.

In signomial geometric programming, early efforts concentrated on finding a local minimum point. Computational comparisons of various methods conducted by Rijckaert and Martens (1980) show that the most successful ones are the primal condensation methods. To apply condensation methods, a signomial geometric program is equivalently represented as a complementary geometric program, which involves all posynomial functions but with fractional constraints of the form $f(x)/g(x) \leq 1$. If $g(x)$ consists of a single term, i.e., it is a monomial, then $f(x)/g(x)$ reduces to a posynomial function. If this is true for all the fractional constraints then the problem reduces to a posynomial geometric program, which is a convex programming problem. Now, suppose that $g(x)$ consists of the sum of the monomials $p_j(x)$, $j \in J$. Having a feasible point $\bar{x}$, $g(x)$ can be condensed into a monomial using the arithmetic mean versus geometric mean inequality as follows:

$$g(x) = \sum_{j \in J} p_j(x) \geq \prod_{j \in J} \left(\frac{p_j(x)}{p_j(\bar{x})/g(\bar{x})} \right)^{p_j(\bar{x})/g(\bar{x})}$$

Performing a similar condensation for each constraint, a posynomial programming restriction of the original problem is obtained. At the optimal solution of this problem, a new condensation is performed, and this procedure continues iteratively until it converges to a stable point (see Avriel and Williams, 1970). Rijkaerd and Martens (1980) use an improved version of this method in their computational tests. They compute a solution to the KKT conditions through an iterative condensation scheme, where a logarithmic variable transformation reduces each subproblem to finding a solution to simultaneous linear equations. To develop an optimization method for the design of structures, Barr, Sarin and Bishara (1989) solve the dual of the posynomial subproblems that are obtained using the condensation technique, where they incorporate an advanced start strategy for the modified Hooke and Jeeves' search method, and they also address various numerical stability issues. Avriel, Dembo and Passy (1975) propose using a second condensation to reduce each constraint to a monomial. Then, a logarithmic variable transformation yields a linear program. However, its solution may not be feasible to the original problem, and so, the algorithm solves a sequence of linear programs in a cutting plane framework to obtain a feasible point to the posynomial approximation of the original problem. Then, a new condensation is performed at that point. A unified presentation of this topic can be found in Beightler and Phillips (1976). Burns (1987) extends the double-condensation method to handle equality constraints explicitly. Using the result that the logarithm of the condensation of a posynomial about a given point $\bar{x}$ is equivalent to the first-order Taylor series approximation of the logarithm of a posynomial at a point $\bar{x}$, he shows that this method performs Newton-Raphson iterations in the log-space of equality constraints, while a cutting plane procedure approximates the inequality constraints. Burns and Locascio (1991) state various properties of the condensation method in the context of solving systems of nonlinear algebraic equations.

Motivated by the convex envelope and branch and bound concepts in the context of concave programming, Gochet and Smeers (1979) propose a branch and bound algorithm to find the global optimum of a signomial programming problem. This time, the form of the problem is changed to a reverse geometric program, where again all the functions in the problem are posynomials but some of the constraints are of the form $g(x) \geq 1$. A posynomial relaxation of a reverse constraint is constructed by using a cutting plane scheme based on a cone associated with a constant vector, called the generator, whose entries (initially) are some lower bounds on the terms of the constraint. This cone has the generator vector as its apex, and the coordinate directions corresponding to each term $p_j(x)$, $j \in J$, of the constraint $g(x) = \sum_{j \in J} p_j(x) \geq 1$, as its extreme directions. Then, in the $|J|$-dimensional space of the terms, the apex is cut off by a less than or equal to type of monomial constraint, which outer approximates the cutting plane $\sum_{j \in J} p_j \geq 1$. The partitioning scheme is based on the optimal solution to the resulting posynomial programming relaxation. Replacing the entry for each term in the generator by its optimal value in turn, the problem is partitioned into (not necessarily disjoint) $|J|$ subproblems corresponding to each new generator. The algorithm continues until all the partitions are eliminated.

For global optimization of generalized geometric programming problems, Maranas and Floudas (1994) propose a branch-and-bound algorithm using a convex programming relaxation problem based on using an exponential transformation of the variables (i.e., each variable x_k in the problem is replaced by e^{t_k}), and subsequently, constructing affine convex envelopes of the resulting terms of the form $-e^{Y(t)}$ over the feasible range of the linear function $Y(t)$. Branching is performed by partitioning the feasible range of variables in the t-variable space. We refer to this method later in Chapter 6 to compare the performance of this lower bounding problem versus RLT-based bounding problems.

Another related problem is the 0-1 polynomial program, where the variables are restricted to be binary valued. Various linearization, algebraic, enumerative, and cutting plane methods have been developed for nonlinear 0-1 programs, as recently surveyed by Hansen, Jaumard, and Mathon (1989). In particular, for constrained polynomial 0-1 programming problems, the linearization-cutting plane method proposed by Balas and Mazzola (1984a, b) appears to perform quite favorably (see Hansen et al., 1989). However, this approach exploits the 0-1 structure of the problem, and is therefore not directly applicable for continuous polynomial programming problems. Sherali and Adams (1994) also discuss the construction of a hierarchy of representations leading to the exact convex hull representation for 0-1 polynomial programming problems.

Shor (1990a,b) introduces a method to reduce a polynomial programming problem into a quadratic minimization problem subject to quadratic constraints by defining new variables of the form $y_i = x_i^2$, $y_{ij} = x_i x_j$, etc. in the constraints. In the sequel, we refer to this method as "Shor's quadrification" or simply "quadrification" scheme. For the unconstrained minimization of polynomials having a finite minimum, Shor gives sufficient conditions for having no duality gap between the actual minimum and the optimal value of the Lagrangian dual problem of the quadrified problem, where all the constraints are dualized. Noting that a polynomial having a finite minimum can be shown to be equivalent to a nonnegative polynomial, the duality gap disappears for problems involving the unconstrained minimization of nonnegative polynomials that can be represented as a sum of the squares of other polynomials. This is particularly true for univariate polynomial functions. However, it has long been known that there exist nonnegative polynomials of as few as four variables and of the fourth degree that cannot be expressed as the sum of the squares of other polynomials. Considering linearly constrained nonconvex quadratic programming problems, the Hessian of the Lagrange function formed by dualizing all the constraints is equivalent to the Hessian of the original objective function itself. Hence, the infimum of the Lagrange function for any fixed value of the dual multipliers is negative infinity. In order to be able to change the characteristic of the Hessian of the Lagrange function, Shor suggests using the constraint factor products as additional implied constraints. Notice that this suggestion has a flavor of the RLT, where we also include constraint factor type products in the original problem before any linearization takes place. Later, we will compare applying RLT directly to the original problem versus applying RLT to the above quadrified problem, and exhibit the superiority of the former over the latter technique.

Recently, Floudas and Visweswaran (1990, 1993) proposed another algorithm to solve polynomial programming problems. Since they deal with problems similar to ours, we explain this algorithm in detail.

Consider the following optimization problem:

$$\text{Minimize}\{f(x, y) : g(x,y) \leq 0, \quad x \in X, \quad y \in Y\}, \tag{2.8}$$

where X and Y are compact subsets of some multi-dimensional real space; $f(x, y)$ and the vector function $g(x, y)$ are convex in x for fixed y, and convex in y for fixed x. (Bilinear equality constraints can be handled explicitly in (2.8).) Polynomial programming problems can be transformed into the above form by using Shor's quadrification method. Defining the Lagrange function $L(x, y, \mu) = f(x, y) + \mu^t g(x, y)$, and using the generalized Benders' decomposition concept, (2.8) can be equivalently expressed as follows.

$$\text{Minimize}\{v(y) : \ v(y) \geq \min_{x \in X} L(x, y, \mu), \ \ \forall \mu \geq 0, \ \ y \in Y \bigcap V\} \tag{2.9}$$

where $V = \{y: y \text{ is feasible for some } x\}$. Notice that the constraints in (2.9) involve a minimization problem, and V is not defined explicitly. For any fixed $y \in Y$, the original problem (2.8) is a convex program - called as "primal problem" -, and gives an upper bound on the optimum value. Using some optimal completion (x^k, μ^k) in the original problem (2.8) for some fixed $y = \bar{y} \in Y$, one can obtain the first-order Taylor series approximation of the Lagrange function as $L_k(x, \bar{y}, \mu^k) = L(x^k, \bar{y}, \mu^k) + (x - x^k)^t \nabla_x L(x^k, \bar{y}, \mu^k)$. Since the Lagrange function is convex in x for any fixed y and μ, then $\min_{x \in X} L(x, \bar{y}, \mu^k) \geq \min_{x \in X} L_k(x, \bar{y}, \mu^k)$. Using the simple bounds $l \leq x \leq u$ implied by the set X, $L_k(x, \bar{y}, \mu^k)$ can be easily minimized by checking the sign of the coefficient of each x-variable. Now, using the linearization of the Lagrange function, and ignoring the set V, a relaxation of (2.9) is obtained as

$$\text{Minimize}\{z : \ z \geq \min_{l \leq x \leq u} \ L_k(x, y, \mu^k), \ \ k = 1, \dots, K, \ \ y \in Y\} \tag{2.10}$$

Let y^K solve the above problem. Given y^k, a primal problem is solved by fixing $y = y^K$, and its solution (x^{K+1}, μ^{K+1}) is used to generate an additional constraint in (2.10) to tighten the relaxation. However, (2.10) is still not easy to solve due to the minimization problems within the constraints. Considering the first iteration $k = 1$, to obtain a more favorable structure, Y is partitioned into subsets over each of which some bound combination x^{B_l} solves $\min\limits_{l \le x \le u} L_1(x, y, \mu^1)$. That is, each partitioned subset is such that the sign of the coefficients of the x-variables in L_1 remain the same over this partition, and so, the solution to all such subsets can be readily enumerated. Notice that the coefficient vector of the x-variables in L_1 is $\nabla_x L(x^1, y, \mu^1)$ for some fixed y, and so, the partitioning of Y is done by considering all possible combinations of the bounds on x-variables, denoted by the set CB. This can be potentially explosive. However, Floudas et al. (1994) have shown how several of such problems can be omitted, thereby greatly improving the computational efficiency.

$$\underset{B_l \in CB}{\text{minimum}} \left[\begin{array}{ll} \text{minimize} & z \\ \text{subject to} & z \ge L_k(x^{B_l}, y, \mu^1) \\ & \nabla_{x_i} L(x^1, y, \mu^1) \le 0, \quad \text{if } x_i^{B_l} = u_i \quad \forall\, i \\ & \nabla_{x_i} L(x^1, y, \mu^1) \ge 0, \quad \text{if } x_i^{B_l} = l_i \quad \forall\, i \\ & y \in Y \end{array} \right] \tag{2.11}$$

In a branch and bound context, at the first iteration, each bound combination creates a separate node at the same level of the branch and bound tree. With each node, the same Lagrange function linearization, but different partitioning constraints, are associated. At the next iteration, the node corresponding to the least lower bound is selected, and it is further partitioned by using the current Lagrange function's gradient. A node subproblem includes the current Lagrange function linearization and partitioning constraints, in addition to all the restrictions associated with the nodes on the path from the current node to the root. A node subproblem solution can be infeasible for the original problem since V is ignored in (2.11). In this case, instead of the primal problem, another convex program is solved to generate a feasibility cut. Using this algorithm, Viswesvaran and Floudas (1990) have solved some interesting problems from the literature. Viswesvaran and Floudas (1993) specialize this algorithm for solving indefinite and concave quadratic programming, as well as quadratically constrained quadratic programming problems. They have presented computational results on solving randomly generated indefinite QP's having upto 25 nonlinear variables and 100 linear variables, and a separable objective function. In another specialization of this algorithm, Visweswaran and Floudas (1992) consider one-dimensional polynomial programming problems. Hansen et al. (1993) show that this last specialization is equivalent to applying particular interval arithmetic methods, and extend it for rational functions. They experiment on incorporating various other interval arithmetic techniques in this algorithm, using problems from the literature.

For quadratically constrained quadratic programming problems, Al-Khayyal et al. (1994) extend Al-Khayyal's (1992) algorithm, which we referenced earlier. To obtain a bilinearly constrained bilinear programming problem, they first transform all the terms of the form x^tQx to x^ty by defining new variables $y = Qx$. Lower and upper bounds on y-variables are obtained using the bounds on original variables, which are required to construct convex and concave envelopes of each bilinear term. Then, the problem is transformed back to the x-space. The resulting linear programming relaxation is then embedded in a convergent branch-and-bound algorithm. This algorithm can be seen as a specialized application of the RLT based algorithm that we have presented in Chapter 5. Using randomly generated test problems having upto 16 variables, they have conducted extensive computational experiments.

Quesada and Grossmann (1992) consider a nonconvex problem in which, besides bilinear terms, rational terms of the form x_i/x_j are present. To obtain a tight relaxation representation, in addition to using convex or concave envelope of each nonlinear term, they derive other classes of nonlinear constraints. When they generate a relaxation problem for the example problem BLP1, as we referred to in Chapters 3 and 4, the resulting constraints turn out to be a subset of the ones generated by the RLT scheme using constraint-factors in addition to bound-factors.

Hansen, Jaumard and Lu (1991) present analytical methods in the form of a series of tests within a branch and bound framework, to solve general nonconvex problems involving functions having explicit analytical forms, and where the constraints are assumed to include simple bounds on the variables. They show that many types of tests used in combinatorial optimization have their counterparts in global optimization, and they also develop new tests using analytical and numerical methods. Some of the ideas they use include assigning Boolean variables to constraints that indicate whether or not a constraint is holding as an equality. Using this technique, one can also represent logical relations between constraints as Boolean equations. For example, setting $\alpha_j = \Pi_{i \in I} \alpha_i$, where α_i is the Boolean variable associated with the i'th constraint, expresses that the constraint $g_j(x) \leq 0$ is tight if and only if all the constraints with index in I are tight. If a tight constraint is linear in at least one variable, then an attempt is made to eliminate this variable from the problem. Determining monotonicity properties of the functions with respect to each variable over the feasible region, can help to obtain bounds or bounding functions for the objective function, or for some of the constraints, by iteratively fixing the variables for which the function is monotonous to their lower or upper bounds. Another method to obtain bounds on any of the functions is by using interval arithmetic. Bounding functions on the constraints can be used in tightening the bounds on variables. Bounds on the optimal value can be obtained by using a lower bounding function for the objective function, or by omitting some constraints, or by appropriate Lagrangian relaxations. Hansen et al. develop many different tests based on these ideas. Branching decisions are motivated by an attempt to fix the values of a maximum number of Boolean variables, or to induce monotonicity in the objective function with respect to some variable, or to avoid a division by zero in variable elimination due to a tight constraint. Each subproblem is checked if it can be solved globally, or if it is infeasible, or if it has no solution better than the incumbent. Then, conditional tests try to fix some Boolean variables and generate additional logical relations that are valid for the current subproblem. If at some subproblem, no further information can be obtained by using logical tests, and no further variable elimination can take place, then it is called a blocking subproblem. At this stage, one has to switch to some other exact solution method to solve the reduced size subproblem. Hansen et al. implement this algorithm on the computer algebra system MACSYMA, and solve selected problems from the literature, some of which are solved exactly for the first time.

For polynomial programming problems, Ryoo and Sahinidis (1993) develop and extend various range reduction strategies based on optimality and feasibility conditions. They incorporate these strategies in a branch-and-bound, or as they refer to as branch-and-reduce, algorithm using an available lower bounding problem. Using some test problems from the literature, they show that these strategies perform favorably. However, without the support of a tight bounding problem, these strategies alone may not perform as satisfactorily in a general purpose algorithm for solving polynomial programming problems.

Another interesting approach to global optimization is the interval analysis. The idea of interval computation has originated as a tool for bounding the rounding errors in computer arithmetic. Moore's (1966) book lays the foundations of interval analysis, and extends its use to bound the effects of errors from many different sources, such as approximation errors, or errors in data. Following the publication of this book, interval analysis has found varied applications in global optimization over the years. In interval analysis, one needs to replace the concept of real numbers by intervals (or interval numbers). Each real arithmetic operation has a counterpart in interval arithmetic. For example, interval addition $[a, b] + [c, d] = [a + c, b + d]$ seems like a natural extension of scalar addition. However, differences emerge for even as simple operations as subtraction. Here, $[a, b] - [c, d] = [a - d, b - c]$. Notice that, if we subtract the interval $[a, b]$ from itself, the result is *not* $[0, 0]$ (unless $a = b$), but it is $[a - b, b - a]$. When a function $f(\cdot)$ is evaluated at an interval X using interval arithmetic, the outcome is yet another interval, which puts bounds on the range of the values of $f(\cdot)$ over X. That is, $f(X) = [f^L, f^U]$ such that $f^L \leq f(x) \leq f^U, \ \forall \ x \in X$.

The main idea in applying interval analysis to global optimization is to eliminate intervals that are guaranteed not to contain the global minimum. For this purpose, some available tools are presented by Hansen (1988). An important tool is the interval Newton method, which can determine all the zeros of a system in a given interval (or in a box for multi-dimensional interval-spaces) by eliminating the intervals that cannot contain a solution to the system. Other tools use the natural bounding outcome of function evaluations at an interval. Having lower and upper bounds on some values of interest, such as the actual function value, or the gradients of the functions, or the Hessians of the functions involved in the problem at a given interval, there are various procedures to check if this interval can be deleted. For unconstrained minimization, a sub-box X is deleted if on X, Newton's method proves that there is no stationary point, or if the gradient of the objective function is not zero, or if the objective function is not convex, or if the function value is greater than the incumbent value. For example, if the upper bound on any diagonal element of the Hessian (which are already in interval form) is negative, then the objective function cannot be convex, and so, X can be deleted. For constrained minimization, a sub-box X is deleted if at X, the interval Newton method proves that the Fritz-John necessary optimality conditions are not satisfied, or if every point is infeasible, or if there is no point that gives a lower objective function value than the incumbent value. Recent books by Hansen (1992), and by Ratschek and Rokne (1988) give unified presentations of parts of the extensive literature on this topic.

A very different approach to solving polynomial programming problems is based on adopting homotopy methods, which have been originally designed to solve simultaneous nonlinear equations. Using an easy problem $S(x) = 0$ having a unique solution x^0, classical continuation methods augment a given hard to solve system $F(x) = 0$ to the form $\rho(\lambda, x) = \lambda F(x) + (1 - \lambda)S(x) = 0$. The system $\rho(\lambda, x) = 0$ is solved for increasing values of λ in small steps from $\lambda = 0$ to $\lambda = 1$, at which point, the original system is solved. In this fashion, the trivial problem $S(x) = 0$ is continuously deformed to the original problem $F(x) = 0$. However, this method is prone to fail because the Jacobian matrix of ρ could become singular at some point. In modern homotopy approaches, this problem is resolved by lifting the system into a higher dimensional space to control singularities, and globally convergent probability-one homotopy algorithms have been developed. Watson (1986) explains the probability-one term as for almost all (in technical sense of Lebesgue measure) choices of some parameter vector involved in the homotopy map, there are no singular points and the method is globally convergent. A C^2 map $\rho(a, \lambda, x)$, where $a \in \mathbb{R}^m$, $\lambda \in [0, 1)$, $x \in \mathbb{R}^n$, is constructed such that for any fixed a, the map $\rho_a(\lambda, x) = \rho(a, \lambda, x)$ satisfies the following conditions: (*i*) $\rho_a(\lambda, x) = 0$ has a unique solution x^0; (*ii*) $\rho_a(1, x) = F(x)$; (*iii*) $\rho_a^{-1}(0) \equiv \{(\lambda, x) : \rho_a(\lambda, x) = 0\}$ is bounded; (*iv*) and finally, ρ is transversal to zero, that is, for almost all a, the Jacobian matrix $D\rho_a(\lambda, x)$ has full rank on $\rho_a^{-1}(0)$.

Some typical forms of the homotopy map $\rho_a(\lambda, x)$ are $\lambda F(x) + (1 - \lambda)(x - a)$, $\lambda F(x) + (1 - \lambda)S(a, x)$, or $\lambda F(x) + (1 - \lambda)S(a, \lambda, x)$. Starting at $(\lambda, x) = (0, x^0)$ the algorithm tracks the path $\rho_a^{-1}(0)$ until λ reaches the value 1. By construction, $\rho_a^{-1}(0)$ consists of smooth , nonintersecting curves in $[0, 1) \times \mathbb{R}^n$. An important property of these curves is that they have finite arc lengths in any compact subset of $[0, 1) \times \mathbb{R}^n$. Watson, Billups and Morgan (1987) implement the homotopy algorithms as a collection of Fortran codes in the package HOMPACK. For tracking the arc length parameterized zero curve, they use three different approaches based on ordinary differential equations, or on normal flow, or on augmented Jacobian matrix techniques. For the case of polynomial systems, HOMPACK has a specialized drive, which finds all the real and complex solutions to $F(x) = 0$, including the solutions at infinity.

To solve polynomial programming problems by using homotopy methods, Kostreva and Kinard (1991) propose finding all the solutions of the KKT equations. For problems with inequality constraints, they suggest using squared slack variables to avoid nonnegativity conditions on the dual multipliers. In a second approach, they drop the original inequality constraints and nonnegativity restrictions on the corresponding dual multipliers to find all the solution to a relaxed system that consists of dual optimality and complementary slackness conditions. To handle the inequality constraints, Vasudevan, Watson and Lutze (1988) use

Mangarasian's complementarity theorem, which states that if a monotonically increasing function $\theta\colon \mathbb{R} \rightarrow \mathbb{R}$ exists such that $\theta(0) = \theta'(0) = 0$, then $\{(x, \lambda) : \theta(|g_i(x) - \lambda_i|) - \theta(g_i(x)) - \theta(\lambda_i) = 0,\ i = 1, \ldots m\} = \{(x, \lambda) : g_i(x) \geq 0,\ \lambda_i \geq 0,\ \lambda_i g_i(x) = 0,\ i = 1, \ldots m\}$, by choosing $\theta(t) = t^3$.

A comprehensive treatment of deterministic methods in global optimization, emphasizing branch and bound and outer approximation methods, is given in a recent book by Horst and Tuy (1993).

2.4 Nondeterministic Approaches

Nondeterministic algorithms for solving global optimization problems have attracted considerable attention in the literature. These algorithms adopt a probabilistic convergence criterion that can be stated as follows: as the sample size of the random observations increases, the probability of having the global optimum in the sampled observations approaches unity. In this section, we give a classification and a brief description of nondeterministic algorithms, along with seminal as well as recent references where up-to-date developments can be found. Nondeterministic approaches for global optimization problems can be categorized in the following two groups: (i) random search methods, and (ii) methods based on a stochastic modelling of the objective function.

An excellent introductory review on random search methods, along with a critical evaluation of various approaches, is presented by Schoen (1991). He emphasizes that one of the most difficult and critical issues in most stochastic algorithms is the choice of appropriate stopping criteria, an issue that has currently not been resolved satisfactorily. In a generic form, random search methods involve the generation of random points over the domain of the objective function, an optional application of a local search, and a check if a prescribed stopping criterion is satisfied. The choice of omitting the local search procedure, or applying it only for the final point, or applying it for all the randomly generated points, creates the distinction between the methods known as pure random search, single start, and multistart, respectively. Sample observations may be generated using various methods, such as, from simplest to more complicated ones, generating independent identically distributed (iid) random points over the feasible region, generating random points over some neighborhood of the current point, or adaptively generating random points based on the previously generated ones. In one particular method, given the current point, the next point is determined by moving along a random direction, that is typically generated over a sphere centered at the current point. Patel et al. (1988) and Zabinsky and Smith (1992) consider generating a random point over a convex feasible region such that its corresponding objective function value is strictly less than the current best one. The number of iterations in this algorithm has a linear complexity in the dimension of the problem. However, the difficulty is to generate observations conforming to desired specifications. For this purpose, Zabinsky et al. (1993) use a random directions method coupled with a randomly generated step-size from a population that would yield a feasible point in the given direction. The resulting point is accepted if it improves the current best solution.

In the multistart method, applying a local optimization procedure for each randomly generated point might be computationally quite burdensome as well as wasteful, since different starting points may lead to the same local optimal solution. If the region of attraction of available local minima can be determined, then the local optimization procedure is applied only to those points that are not in the region of attraction of any known local minimum. For this purpose, Becker and Lago (1970) have suggested using statistical clustering techniques that are typically based on some distance measure criteria. To aid the clustering analysis, a preprosessing of the given sample of observations can be conducted by eliminating those that have relatively high objective function values, or by running a few iterations of a local optimization procedure for each sample point (see Torn (1978)). Boender et al. (1982) modify the earlier spherical shaped clusters that are constructed based on critical densities, to ellipsoidal ones, in order to obtain a better approximation of the level sets of the objective function near a local minimum. In this so-called single linkage clustering, clusters are iteratively formed by comparing the distance between a given point and the nearest point in a cluster versus a critical value. In theoretically more sound multi-linkage clustering methods, a local optimization procedure is applied to a point if there is no better observation within a critical distance. Theoretical developments and computational results on clustering types of algorithms have been presented by Rinnooy Kan and Timmer (1987a, b).

Betro (1984) and Boender and Rinnooy Kan (1983) have introduced various stopping rules for multistart algorithms based on Bayesian methods. Given the current number of local minima found, an earlier stopping criterion of a posteriori estimate of the total number of local minima have been later modified by incorporating more complex loss measures. By assigning costs to a local search procedure and to the situation where an estimate on the total number of local minima is greater than the number of the currently available ones, a stopping criterion is computed by minimizing the total estimated cost. Using another cost structure that takes into account the actual value of the current best local minimum, the resulting stopping criterion compares the expected improvement in the objective function from the current best value versus a cost assigned to applying a local search in the same units of the objective function. A review of Bayesian stopping rules, and up-to-date references on this topic can be found in Betro (1991).

To solve linearly constrained concave programming problems, Phillips et al. (1991) adopt a particular parallel implementation of the multistart method. For each randomly generated point, they determine a vertex of the feasible region where they start a local search. The stopping criteria they used is based on the expected total number of local minima. One of the classes of problems considered in their computational experiments is concave quadratic programs, which we have treated as an instance of indefinite quadratic programming in Chapters 3 and 4. As one of the challenging problems, they solved the problem CQP2 described in Chapter 4, consuming 28.4 cpu seconds (or 7.1 wall clock seconds using 4 parallel processors) on a Cray X-MP EA/464 supercomputer. Using the branch-and-bound algorithm we have proposed, the same problem is solved under 2 cpu seconds for 1% optimality criterion, as reported in Section 4.9.

Starting with Kirkpatrick et al.'s (1983) pioneering paper, the use of simulated annealing to solve complex optimization problems has become a popular subject of investigation. For global optimization, this method could be considered as a random search method with an adaptive, stochastic acceptance/rejection rule that permits ascent steps in order to escape from a local optimum. A candidate randomly generated point over the neighborhood of the current point is accepted as the next iterate if the objective function value improves, or if a stochastic criterion based on the objective function value and the current state of the algorithm is satisfied. The state of the algorithm is traditionally called the temperature of the system, and at the initial stages of the algorithm, higher temperatures make it more likely to accept a nonimproving point. As the algorithm progresses, the system is slowly cooled. Various algorithms that combine this approach with other methods have been developed. For example, Piccioni (1987) applies a deterministic descent method while allowing the possibility of ascent moves based on the temperature of the system at random time instances. Lucidi and Piccioni (1989) compute a conditional probability that is parametrized by the temperature of the system, given the previous local optimizers and their starting points, to determine when to start a local optimization procedure. The idea behind simulated annealing also appears in another approach where the next iterate is determined by solving a particular stochastic differential equation, known as the Langevin equation. In Aluffi-Pentini et al. (1985), upon a numerical integration of the Langevin equation, the direction of motion from the current point is governed by, in addition to the familiar descent direction of the negative gradient of the objective function, a random variable that may allow ascent directions. This process is parametrized by a similar notion as the temperature of the system. A recent adaptation to global optimization of the simulated annealing methods that were designed for solving discrete optimization problems, has been presented by Dekkers and Aarts (1991).

A second group of nondeterministic algorithms includes procedures that use a stochastic modelling of the objective function based on Bayesian methods. For one-dimensional problems, a computationally manageable stochastic model is available, which is known as the Wiener process. However, the extension of this approach to multi-dimensional problems requires an update procedure that involves the inversion of a covariance matrix whose order is determined by the sample size at each iteration. Therefore, for multi-dimensional functions, the major difficulty is to find computationally tractable stochastic models. Mockus et al.

(1978) determine the next iterate by minimizing a risk function that represents the expected deviation from the global minimum for a given point. This risk function happens to be multimodal, although the accuracy of its global minimum is not required to be high. This method seems to be more applicable to functions that are computationally very expensive to evaluate. In order to simplify some of the expressions that are computationally burdensome for the stochastic modelling approach, Zilinskas (1982, 1985) approximately express them using an extrapolation theory. For recent references on this topic, see Mockus (1994). A description of the theory and related algorithmic developments are presented in a unified form in a book by Mockus (1989).

For further in depth information on nondeterministic algorithms for solving global optimization problems, we refer the interested reader to the books by Torn and Zilinskas (1987), and in particular, by Zhigljavsky (1991). We now proceed to describe the analytical development of the present research effort.

Chapter III

RLT Development for Quadratic Programming

As introduced earlier, linearly constrained, quadratic programming problems (QP) are a special case of the general class of polynomial programming problems. Let us restate this problem below for convenience:

$$\begin{aligned} \text{QP} : \quad & \text{Minimize} \;\; cx + x^tQx \\ & \text{subject to} \;\; Ax \le b \\ & \qquad 0 \le l_k \le x_k \le u_k < +\infty \qquad k = 1, \dots, n \end{aligned}$$

Notice that we have assumed the lower and upper bounds on all the variables to be finite. These might be prespecified bounds, or they might be derived from the other constraints of QP and imposed herein for certain algorithmic purposes. As we proceed, we will point out modifications to be made in our algorithmic constructs whenever such bounds are known to be implied by the other constraints defining the problem. Our objective is to solve QP for nonconvex objective functions, that is, when the Hessian matrix $2Q$ is indefinite, or negative semidefinite. However, for the latter pure concave case, we will not take advantage of the extremality property of the global optimum, and treat it just as another instance of the indefinite case.

In this chapter, we investigate various specialized RLT applications to generate lower bounding problems for QP, one of which will be later embedded within a branch-and-bound procedure. The first scheme we propose involves the generation of second-order constraints, by considering pairwise products of *both* the constraint-factors and the bound-factors. These constraints are subsequently linearized to yield a lower bounding linear program. We show that the resulting linear program that involves all the constraint-factor and the bound-factor products is invariant under affine transformations. However, if eigen-transformation is used as a particular linear transformation, this enables us to introduce additional nonlinear convex constraints that can indeed further strengthen the linear programming relaxation produced by RLT. Such nonlinear constraints can be suitably handled within a Lagrangian dual procedure, without hampering the efficiency of the solution procedure as compared with that for solving the linear programming bounding problem.

In order to further strengthen the bounding problem, we derive additional constraints based on third-order bound-factor products that are projected onto the quadratic space. Another related, but different, class of constraints generated is based on squaring bound (or constraint) factor *differences*. We also present a rule for reducing the number of new second-order constraints generated for the bounding problem without compromising much on the quality of the resulting lower bound.

3.1 Reformulation Via Quadratic Constraint Generation (RLT-LP)

In this section, we generate quadratic implied constraints, and subsequently linearize them to obtain RLT constraints. To generate the quadratic constraints, we consider pairwise products of bound-factors and constraint-factors. For brevity of presentation, let us combine bound and constraint-factors in a single set as follows:

$$\begin{bmatrix} (b_i - a_i x) \geq 0, & i = 1,\dots, m \\ (u_k - x_k) \geq 0, & k = 1,\dots, n \\ (x_k - l_k) \geq 0, & k = 1,\dots, n \end{bmatrix} \equiv \begin{bmatrix} (g_i - G_i x) \geq 0 \\ i = 1,\dots, m + 2n \end{bmatrix} \tag{3.1}$$

where $a_i x \leq b_i$ is the i^{th} (structural) constraint from $Ax \leq b$, for $i = 1,\dots, m$. At the reformulation step, we take all possible pairwise products of the factors in (3.1), including self-products, to generate the following nonlinear implied constraints that are included in the original problem QP:

$$(g_i - G_i x)(g_j - G_j x) \geq 0 \quad \forall\ 1 \leq i \leq j \leq m + 2n \tag{3.2}$$

We then linearize the resulting augmented problem by substituting

$$w_{kl} = x_k x_l \quad \forall\ 1 \leq k \leq l \leq n \tag{3.3}$$

This substitution associates a separate new variable with each distinct nonlinear term in the problem. We often refer to these variables as *RLT variables.* The resulting problem is called the *first-level* or *first-order RLT*, since it employs first-level (linear) factors to generate new constraints. Denoting $[(g_i - G_i x)(g_j - G_j x)]_\ell \geq 0,\ \forall\ 1 \leq i \leq j \leq m + 2n$ as the resulting linearized constraints, we obtain the following lower bounding first-level RLT linear program, where q_{kl} $(=q_{lk})$ is the (k, l)th element of the symmetric matrix Q.

RLT-LP: Minimize $$\sum_{k=1}^{n} c_k x_k + \sum_{k=1}^{n} q_{kk} w_{kk} + 2\sum_{k=1}^{n-1} \sum_{l=k+1}^{n} q_{kl} w_{kl} \tag{3.4.1}$$

subject to $$[(g_i - G_i x)(g_j - G_j x)]_\ell \geq 0 \qquad \forall\ 1 \leq i \leq j \leq m + 2n \tag{3.4.2}$$

Notice that the original constraints of QP are not included in RLT-LP, since, as we shall show in Proposition 1 later, these constraints are implied by the RLT constraints (3.4.2), even if the feasible region is not assumed to be bounded, provided that at least one variable has a bounded range over the feasible region.

For any feasible solution to problem QP, there exists a feasible solution to RLT-LP having the same objective function value through the definitions (3.3). However, the converse is not necessarily true. Therefore, RLT-LP is a relaxation of QP that yields a lower bound on the global minimum of QP. Moreover, if $(\bar{x}, \bar{w})$ solves RLT-LP, then since $\bar{x}$ is feasible for QP, it provides an upper bound on this problem. In particular, if this solution also satisfies the definitions (3.3) for all the nonlinear terms appearing in QP, then $\bar{x}$ solves QP.

REMARK 1 (Equality Constraints): If there is some equality constraint $G_e x = g_e$ in QP, then we only need to consider the product of the constraint-factor $(g_e - G_e x) = 0$ with each variable x_k, $k = 1, \dots, n$, since all the other RLT constraints generated via this constraint can be obtained by suitably surrogating the constraints $[x_k(g_e - G_e x)]_\ell = 0$, $k = 1, \dots, n$. However, in this case, we need to include the original equality constraint $G_e x = g_e$ in RLT-LP. For example, any RLT constraint of type $[(g_i - G_i x)(g_e - G_e x)]_\ell = 0$ can be obtained using the following surrogating scheme:

$$0 = g_i(g_e - G_e x) + \sum_{k=1}^{n} G_{ik}[x_k(g_e - G_e x)]_\ell$$

$$= \left[g_i(g_e - G_e x) + \sum_{k=1}^{n} G_{ik} x_k (g_e - G_e x) \right]_\ell$$

$$= [(g_i - G_i x)(g_e - G_e x)]_\ell$$

3.2 An Illustrative Example

In this section, we shall use a small illustrative example problem to provide insights on RLT, and on how it works, and why it works. Consider the following concave quadratic program:

$$\begin{array}{ll} \text{Minimize} & z = -(x_1 - 12)^2 - x_2^2 \\ \text{subject to} & -6x_1 + 8x_2 \leq 48 \\ & 3x_1 + 8x_2 \leq 120 \\ & x_1 \leq 24 \\ & x_1, x_2 \geq 0 \end{array}$$

The optimal solution is $(x_1^*, x_2^*) = (24, 6)$, or alternatively, (0, 6), and the optimal objective function value is $z^* = -180$. Although, it is not explicitly stated, the upper bound restriction $x_2 \leq 12$ is implied by the above constraints. The feasible region happens to be bounded, and given by the convex hull of the vertices (0, 0), (0, 6), (8, 12), (24, 6), and (24, 0). Before generating the first-level RLT for this problem, let us identify the bound-factors as $(24 - x_1) \geq 0$, $x_1 \geq 0$, $x_2 \geq 0$, and denote the constraint-factors as $s_1 = (48 + 6x_1 - 8x_2) \geq 0$, and $s_2 = (120 - 3x_1 - 8x_2) \geq 0$. The first-level RLT problem is then generated by constructing the 15 pairwise products (including self-products) of these factors, and then linearizing the resulting quadratically constrained QP via the variable redefinitions

$w_{11} = x_1^2, \quad w_{12} = x_1 x_2, \quad \text{and} \quad w_{22} = x_2^2$

The resulting linear lower bounding problem is given below.

$$\begin{array}{ll}
\text{Minimize} & -w_{11} - w_{22} + 24x_1 - 144 \\
\text{subject to} & [s_1 s_1]_\ell = 2304 + 576x_1 - 768x_2 + 36w_{11} - 96w_{12} + 64w_{22} \geq 0 \\
& [s_1 s_2]_\ell = 5760 + 576x_1 - 1344x_2 - 18w_{11} - 24w_{12} + 64w_{22} \geq 0 \\
& [s_2 s_2]_\ell = 14400 - 720x_1 - 1920x_2 + 9w_{11} + 48w_{12} + 64w_{22} \geq 0 \\
& [(24 - x_1)s_1]_\ell = 1152 + 96x_1 - 192x_2 - 6w_{11} + 8w_{12} \geq 0 \\
& [(24 - x_1)s_2]_\ell = 2880 - 192x_1 - 192x_2 + 3w_{11} + 8w_{12} \geq 0 \\
& [x_1 s_1]_\ell = 48x_1 + 6w_{11} - 8w_{12} \geq 0 \\
& [x_2 s_1]_\ell = 48x_2 + 6w_{12} - 8w_{22} \geq 0 \\
& [x_1 s_2]_\ell = 120x_1 - 3w_{11} - 8w_{12} \geq 0 \\
& [x_2 s_2]_\ell = 120x_2 - 3w_{12} - 8w_{22} \geq 0 \\
& [(24 - x_1)^2]_\ell = 576 - 48x_1 + w_{11} \geq 0 \\
& [(24 - x_1)x_1]_\ell = 24x_1 - w_{11} \geq 0 \\
& [(24 - x_1)x_2]_\ell = 24x_2 - w_{12} \geq 0 \\
& [x_1 x_1]_\ell = w_{11} \geq 0 \\
& [x_1 x_2]_\ell = w_{12} \geq 0 \\
& [x_2 x_2]_\ell = w_{22} \geq 0
\end{array}$$

The solution to this first-level RLT problem is given by $(\bar{x}_1, \bar{x}_2, \bar{w}_{11}, \bar{w}_{12}, \bar{w}_{22}) = (8, 6, 192, 48, 72)$ and has an objective function value of -216. Hence, -216 provides a lower bound on QP. Notice that $\bar{x}_1\bar{x}_2 = \bar{w}_{12}$; however, the same is not true for $\bar{w}_{11} = 192 \neq \bar{x}_1^2 = 64$, and $\bar{w}_{22} = 72 = \bar{x}_2^2 \neq 36$. Notice also that $\bar{x} = (8, 6)$ is feasible for QP and has an objective function value of -52. This provides an upper bound on QP.

Now, let us partition the problem via the dichotomy $x_1 \leq 8$ or $x_1 \geq 8$. When we solve a first-level RLT problem separately for each partitioned subproblem, the optimal value turns out to be -180 in both cases. That is, the original problem is solved after a single branching on x_1.

To show what RLT is trying to achieve, let us introduce cubic RLT constraints. In the following problem, we consider certain selected second-order RLT constraints along with some cubic bound/constraint-factor products to yield additional RLT constraints. (This selection is done for the purpose of illustration, and is motivated by the magnitude of the optimal dual variable values of the first-level RLT problem.) The resulting linear program given below is a (partial) second-level RLT problem.

$$\begin{array}{ll}
\text{Minimize} & -w_{11} - w_{22} + 24x_1 - 144 \\
\text{subject to} & [(24 - x_1)s_1]_\ell = 1152 + 96x_1 - 192x_2 - 6w_{11} + 8w_{12} \geq 0 \\
& [x_1 s_2]_\ell = 48x_1 + 6w_{11} - 8w_{12} \geq 0 \\
& [(24 - x_1)x_1]_\ell = 24x_1 - w_{11} \geq 0 \\
& [(24 - x_1)x_1 s_1]_\ell = 1152x_1 + 96w_{11} - 192w_{12} - 6w_{111} + 8w_{112} \geq 0 \\
& [(24 - x_1)x_2 s_1]_\ell = 1152x_2 + 96w_{12} - 192w_{22} - 6w_{112} + 8w_{122} \geq 0 \\
& [(24 - x_1)x_1 s_2]_\ell = 2880x_1 - 192w_{11} - 192w_{12} + 3w_{111} + 8w_{112} \geq 0 \\
& [x_1 x_2 s_2]_\ell = 120w_{12} - 3w_{112} - 8w_{122} \geq 0
\end{array}$$

Above, additional RLT variables have been defined as before to represent the newly created cubic terms. These are given by,

$$w_{111} = x_1^3, \quad w_{112} = x_1^2 x_2, \quad w_{122} = x_1 x_2^2$$

The solution to this revised problem is given by $(\hat{x}_1, \hat{x}_2) = (0, 6)$, $\hat{w}_{22} = 36$, $\hat{w}_{11} = \hat{w}_{12} = \hat{w}_{111} = \hat{w}_{112} = \hat{w}_{122} = 0$, having the objective function value -180. Hence, the above linear bounding problem composed of selected second and third-order RLT constraints solves the original problem. In fact, we can demonstrate in this instance that these constraints effectively describe the convex envelope of the objective function for the original concave quadratic program.

To see this, let us define a particular surrogate of the above third-order RLT constraints as follows.

$$surr[cubic]_\ell \equiv 1/512[(24 - x_1)x_1s_1]_\ell + 1/192[(24 - x_1)x_2s_1]_\ell + 1/256[(24 - x_1)x_1s_2]_\ell + 1/192[x_1x_2s_2]_\ell \geq 0$$

When we further surrogate this constraint with the objective function, and the second-order RLT constraints using two different sets of weights in turn, we obtain the following two constraints that yield the desired convex envelope representation. The graph of the objective function and its convex envelope over the feasible region is presented in Figure 1.

$$(z + w_{11} + w_{22} - 24x_1 + 144 = 0) + 7/16([(24 - x_1)x_1]_\ell \geq 0) + (surr[cubic]_\ell \geq 0) \Rightarrow z \geq -6x_2 - 144$$

$$(z + w_{11} + w_{22} - 24x_1 + 144 = 0) + 7/144([(24 - x_1)s_1]_\ell \geq 0) + 7/144([x_1s_2]_\ell \geq 0) + (surr[cubic]_\ell \geq 0) \Rightarrow z \geq (10/3)x_2 - 200$$

Hence, it turned out that the given quadratic problem was solved as the above single linear program. In fact, as this example exhibits, the RLT scheme attempts to approximate the convex envelope of the objective function over the feasible region in deriving a lower bounding linear program. If this approximation is composed properly, a tight linear representation of the problem can be derived.

3.3 Insights into the First-Level RLT Application

In this section, we present some results that provide insights into the first-level RLT problem. Some implications of these results will find direct use in designing a suitable branch-and-bound algorithm. In concert with definition (3.1),

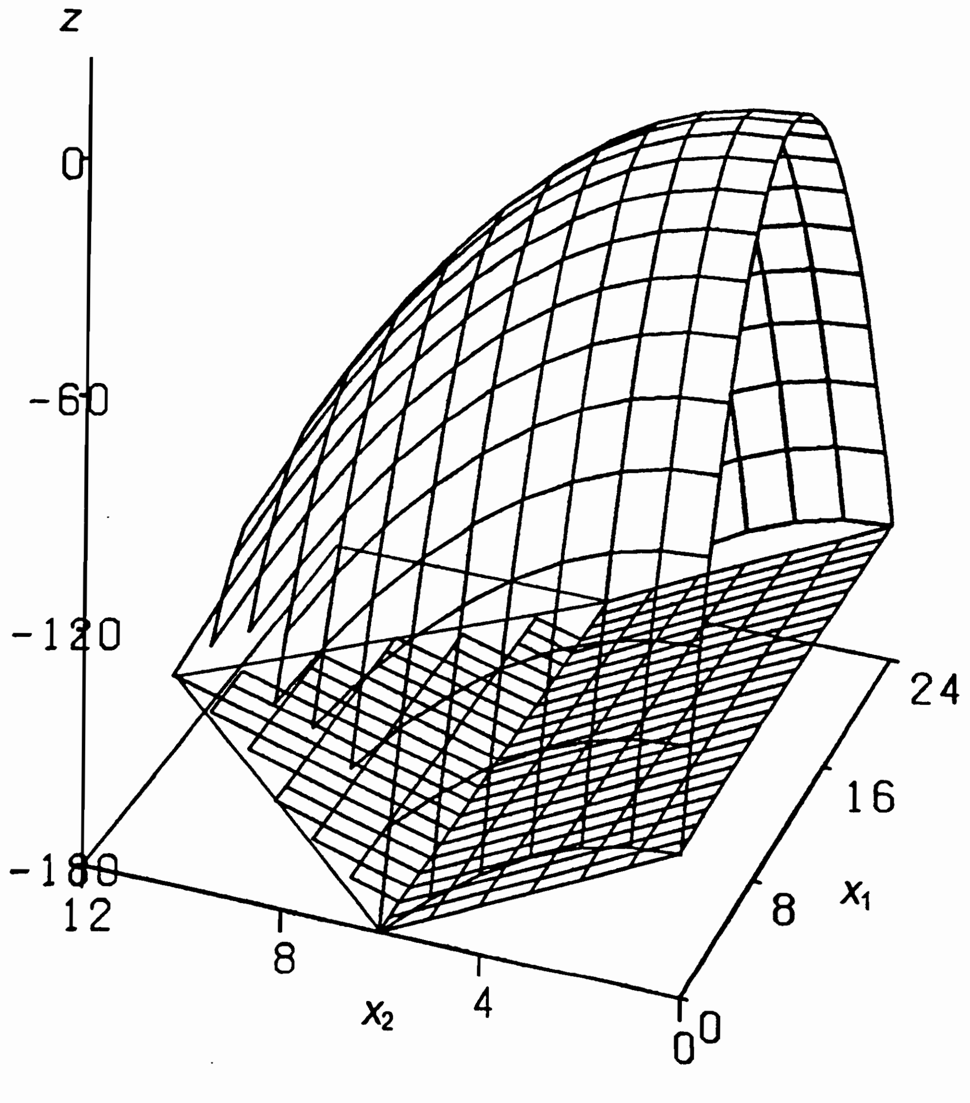

Figure 1. Graph of the objective function and its convex envelope over the feasible region for the example problem of Section 3.2.

we refer to the constraint set of QP, including the simple bound restrictions on the variables, as $Gx \leq g$.

The first result below shows that the original constraints $Gx \leq g$ need not be included in the first-level RLT problem, even under a relaxed boundedness assumption on QP.

PROPOSITION 1: Suppose that the feasible region of QP is not necessarily bounded (possibly including unrestricted variables) but that for some variable x_k, $k \in \{1,..., n\}$, we have,

$$U_k \equiv \max\{x_k : \; G_i x \leq g_i, \, i = 1,..., m + 2n\} < \infty \tag{3.5.1}$$

$$L_k \equiv \min\{x_k : \; -G_i x \geq -g_i, \, i = 1,..., m + 2n\} > -\infty \tag{3.5.2}$$

where $U_k > L_k$. Then, the original constraints $Gx \leq g$ are implied by the RLT constraints $[(g_i - G_i x)(g_j - G_j x)]_\ell \geq 0 \;\; \forall 1 \leq i \leq j \leq m + 2n$.

Proof: Let $\mu^u \geq 0$ and $\mu^l \geq 0$ be the optimal dual multiplier vectors associated with the constraints in (3.5.1) and (3.5.2), respectively. Then, we have $G^t\mu^u = -G^t\mu^l = e_k$, where $e_k \in \mathbb{R}^n$ is the unit vector with entry 1 at the k^{th} position, and also, $g^t\mu^u = U_k$, and $-g^t\mu^l = L_k$.

Now, for any $j \in \{1,..., m + 2n\}$, consider the surrogate of all RLT constraints involving the constraint-factor $(g_j - G_j x \geq 0)$, obtained by using the weights μ^u.

$$0 \le \sum_{i=1}^{m+2n} \mu_i^u [(g_i - G_i x)(g_j - G_j x)]_\ell = \sum_{i=1}^{m+2n} \mu_i^u (g_i g_j - g_i G_j x - g_j G_i x + [(G_i x)(G_j x)]_\ell)$$

$$= -g_j \sum_{i=1}^{m+2n} \mu_i^u G_i x + (g_j - G_j x) \sum_{i=1}^{m+2n} \mu_i^u g_i + \sum_{i=1}^{m+2n} \mu_i^u G_i [xx^t]_\ell G_j^t$$

$$= -g_j x_k + (g_j - G_j x) U_k + e_k^t [xx^t]_\ell G_j^t$$

After rearranging the terms in the final expression, and noting that $e_k^t[xx^t]_\ell G_j^t = [x_k(G_j x)]_\ell$, we obtain,

$$U_k(g_j - G_j x) - [x_k(g_j - G_j x)]_\ell = [(U_k - x_k)(g_j - G_j x)]_\ell \ge 0 \qquad (3.6.1)$$

Replacing μ^u by μ^l, and then following the same steps as above, we obtain,

$$-L_k(g_j - G_j x) + [x_k(g_j - G_j x)]_\ell = [(x_k - L_k)(g_j - G_j x)]_\ell \ge 0 \qquad (3.6.2)$$

The surrogate of (3.6.1) and (3.6.2) gives

$$(U_k - L_k)(g_j - G_j x) \ge 0 \qquad (3.7)$$

which implies $G_j x \le g_j$, since $U_k > L_k$. Since this is true for any $j \in \{1, \dots, m+2n\}$, including the bounding constraints, the proof is complete. ■

Consider the constraints $Gx \le g$, and suppose that some of these constraints are implied by the others. In particular, let us assume that the constraints $G_i x \le g_i$, $i \in \{1, \dots, m'\}$ imply the other constraints within $Gx \le g$, and represent, say a minimal set of non-implied constraints. The second proposition below exhibits that we do not need to use any implied constraint in generating RLT constraints, because such constraints would be implied by the RLT constraints that are generated via the products of the non-implied constraints. Hence, any constraints within $Gx \leq g$ that are known to be implied can be discarded without loss of any tightness in the resulting RLT problem, and moreover, it is futile to use any implied constraint, such as implied bounds on variables, to generate additional RLT constraints. The following result encapsulates the foregoing comment. For convenience, let us refer to the RLT constraints generated via the non-implied set of original constraints as

$$\Gamma \equiv \{[(g_i - G_i x)(g_j - G_j x)]_\ell \geq 0 \quad \forall \ 1 \leq i \leq j \leq m'\} \tag{3.8}$$

PROPOSITION 2: Suppose that the assumption of Proposition 1 holds, and let $\alpha x \leq \beta$ be implied by the constraints $G_i x \leq g_i$, $i = 1, \ldots, m'$. Then, the RLT constraints $[(\beta - \alpha x)(g_i - G_i x)]_\ell \geq 0$, $i = 1, \ldots, m'$, and $[(\beta - \alpha x)^2]_\ell \geq 0$ are also implied by the RLT constraints $[(g_i - G_i x)(g_j - G_j x)]_\ell \geq 0$, $1 \leq i \leq j \leq m'$. In particular, the constraints of the first-level RLT problem are all implied by the RLT constraints contained within Γ.

Proof: Since $\beta \geq \text{maximum}\{\alpha x: G_i x \leq g_i, \ i = 1, \ldots, m'\}$, there exist dual multipliers $\mu_i \geq 0$, $i = 1, \ldots, m'$ such that

$$\sum_{i=1}^{m'} \mu_i G_i = \alpha \quad \text{and} \quad \sum_{i=1}^{m'} \mu_i g_i \leq \beta \tag{3.9}$$

Now, for any $j \in \{1, \ldots, m'\}$, consider the surrogate of the following RLT constraints from Γ involving the constraint-factor $(g_j - G_j x) \geq 0$, obtained by using the weights μ.

$$0 \le \sum_{i=1}^{m'} \mu_i [(g_i - G_i x)(g_j - G_j x)]_\ell = \sum_{i=1}^{m'} \mu_i (g_i g_j - g_i G_j x - g_j G_i x + [(G_i x)(G_j x)]_\ell)$$

$$= -g_j \sum_{i=1}^{m'} \mu_i G_i x + (g_j - G_j x) \sum_{i=1}^{m'} \mu_i g_i + \sum_{i=1}^{m'} \mu_i G_i [xx^t]_\ell G_j^t$$

Since by Proposition 1, $(g_j - G_j x) \ge 0$ is implied by Γ, we get upon using (3.9) that

$$0 \le -g_j \alpha x + \beta(g_j - G_j x) + \alpha[xx^t]_\ell G_j^t = [(\beta - \alpha x)(g_j - G_j x)]_\ell$$

Hence, $[(\beta - \alpha x)(g_j - G_j x)]_\ell \ge 0$ is implied by Γ for all $j = 1, \ldots, m'$. Furthermore, by following the same algebraic steps as above, and using the foregoing assertion, we get

$$0 \le \sum_{j=1}^{m'} \mu_j [(g_j - G_j x)(\beta - \alpha x)]_\ell \le [(\beta - \alpha x)(\beta - \alpha x)]_\ell$$

Hence, the self-product constraint $[(\beta - \alpha x)^2]_\ell \ge 0$ is also implied by Γ. The final assertion of the proposition now follows by inductively taking the implied constraints of $Gx \le g$ one at a time, and establishing as above that the RLT constraints generated via this constraint, including the self-product constraint, are implied by the RLT constraints generated via the other remaining constraints. Discarding such implied constraints sequentially, we deduce that the constraint set Γ implies the other RLT constraints, and this completes the proof. ■

Knowing that the original constraints are implied by the RLT constraints, if an original constraint from $Gx \le g$ is binding at some feasible solution $(\overline{x}, \overline{w})$ to RLT-LP, then one would expect that some of the RLT constraints would also be binding at $(\bar{x}, \bar{w})$. The next result shows that the RLT constraints generated using a constraint-factor that turns out to be binding, are themselves binding. However, for this result, we need the original assumption that the feasible region of QP is bounded.

PROPOSITION 3: Assume that the feasible region of QP is bounded, and suppose that at a given point $(\bar{x}, \bar{w})$, we have $G_i\bar{x} = g_i$, for some $i \in \{1, \ldots, m + 2n\}$. Then, the linearized product of this constraint with any original variable x_k, $k \in \{1, \ldots n\}$ at $(\bar{x}, \bar{w})$ is zero. That is, denoting $[\cdot]_\ell$ evaluated at $(\bar{x}, \bar{w})$ by $[\cdot]_\ell|_{(\bar{x}, \bar{w})}$, we have,

$$[x_k(g_i - G_ix)]_\ell|_{(\bar{x}, \bar{w})} \equiv g_i\bar{x}_k - \sum_{l=1}^{k} G_{il}\bar{w}_{lk} - \sum_{l=k+1}^{n} G_{il}\bar{w}_{kl} = 0, \qquad \forall\ k = 1, \ldots, n. \tag{3.10}$$

In particular, the RLT constraint generated by multiplying this constraint with any other constraint is also binding.

Proof: Under the boundedness assumption of the feasible region, consider the constraints (3.6.1) and (3.6.2), which are implied by or which already exist in Γ by Proposition 2, in a combined form as follows:

$$L_k(g_i - G_ix) \leq [x_k(g_i - G_ix)]_\ell \leq U_k(g_i - G_ix) \qquad \forall\ k = 1, \ldots, n. \tag{3.11}$$

Evaluating (3.11) at $(\bar{x}, \bar{w})$, since $g_i - G_i\bar{x} = 0$, we obtain (3.10) holding true. For any $j \in \{1, \ldots, m + 2n\}$, when we evaluate the following RLT constraint at $(\bar{x}, \bar{w})$,

$$[(g_i - G_ix)(g_j - G_jx)]_\ell = g_j(g_i - G_ix) - \sum_{k=1}^{n} G_{jk}[x_k(g_i - G_ix)]_\ell \geq 0 \tag{3.12}$$

we get by (3.10) and $(g_i - G_i\bar{x}) = 0$, that (3.12) holds as equality. This completes the proof. ■

Next, we address the question whether an application of RLT after using some affine transformation can possibly produce a different relaxation. This might be of interest, in particular, if one wished to investigate the effect of applying RLT to different nonbasic space representations of the linear constraints defining QP, or the effect of employing an eigen-transformation on QP before applying RLT. More specifically, given the quadratic programming problem QP: minimize$\{cx + x^tQx : Gx \leq g\} < \infty$, define a nonsingular affine transformation $s = Bx + p$ to represent QP in s-space as follows, using the substitution $x \equiv B^{-1}s - B^{-1}p$, and where $B^{-t} \equiv (B^{-1})^t$.

$$\text{QP}': \quad -cB^{-1}p + p^tB^{-t}QB^{-1}p + \text{Minimize} \quad (cB^{-1} - 2p^tB^{-t}QB^{-1})s + s^tB^{-t}QB^{-1}s$$
$$\text{subject to } GB^{-1}s \leq g + GB^{-1}p$$

Let RLT-LP and RLT-LP′ be the linear programs obtained by applying the first-level RLT to QP and QP′, respectively, using all possible pairwise constraint-factor products. These problems can be stated as follows, where G_i is the i^{th} row of G, for $i \in \{1, \ldots, m + 2n\}$.

RLT-LP:

Minimize $cx + [x^tQx]_\ell$

subject to $g_ig_j - (g_iG_j + g_jG_i)x + [(G_ix)(G_jx)]_\ell \geq 0 \quad \forall (i,j) \in M_R \equiv \{(i,j): 1 \leq i \leq j \leq m + 2n\}$

RLT-LP′:

$$
\begin{aligned}
&-cB^{-1}p + p^tB^{-t}QB^{-1}p\\
&+\text{Minimize}\quad (cB^{-1} - 2p^tB^{-t}QB^{-1})s + [s^tB^{-t}QB^{-1}s]_\ell\\
&\text{subject to}\ (g_i + G_iB^{-1}p)(g_j + G_jB^{-1}p) - [(g_i + G_iB^{-1}p)(G_jB^{-1}) + (g_j + G_jB^{-1}p)(G_iB^{-1})]s\\
&\qquad\qquad + [(G_iB^{-1}s)(G_jB^{-1}s)]_\ell \geq 0 \qquad \forall (i,j)\in M_R
\end{aligned}
$$

Also, in accordance with our foregoing discussion, let us define the RLT variables for each quadratic term as $w_{kl} \equiv [x_kx_l]_\ell$ and $y_{kl} \equiv [s_ks_l]_\ell$, $1 \leq k \leq l \leq n$. Henceforth, we will let $v[\cdot]$ denote the value at optimality of the corresponding problem $[\cdot]$. The next proposition shows that the first-level RLT is invariant under affine transformations.

PROPOSITION 4: $v[\text{RLT-LP}] = v[\text{RLT-LP}']$. In particular, if (x^*, w^*) solves RLT-LP, and if u^* is the corresponding optimal dual solution, then

$$(s^*, [y^*]) \equiv (s^*, [ss^t]_\ell^*) = (Bx^* + p, B[xx^t]_\ell^*B^t + p(x^*)^tB^t + Bx^*p^t + pp^t) \tag{3.13}$$

solves RLT-LP′, with the corresponding optimal dual solution being u^*, where $[y^*]$ denotes an $n \times n$ matrix representation of y^*, such that for $1 \leq k \leq l \leq n$, $[y^*]_{kl} = [y^*]_{lk} = y^*_{kl}$, and where $[\cdot]_{kl}$ is the $(k, l)^{th}$ entry of the $n \times n$ symmetric matrix $[\cdot]$.

Proof: From the KKT conditions of the linear program RLT-LP, we have

$$-c - \sum_{(i,j)\in M_R} u^*_{ij}(g_jG_i + g_iG_j) = 0 \tag{3.14.1}$$

and

$$-2Q + \sum_{(i,j)\in M_R} u^*_{ij}(G_j^tG_i + G_i^tG_j) = 0 \tag{3.14.2}$$

(Note that in the dual feasibility conditions (3.14.2), there are $n(n-1)/2$ more equations than the number of w-variables, but since the left-hand side is symmetric, (3.14.2) is valid, having $n(n-1)/2$ duplicated equations.) Dual feasibility of u^* for RLT-LP′ then follows directly from (3.14.1) and (3.14.2), since we have,

$$\begin{aligned} 0 &= (3.14.1)B^{-1} + p^t B^{-t}(3.14.2)B^{-1} \\ &\equiv -cB^{-1} + 2p^t B^{-t} Q B^{-1} - \sum_{(i,j)\in M_R} u_{ij}^*[(g_i + G_i B^{-1} p)(G_j B^{-1}) + (g_j + G_j B^{-1} p)(G_i B^{-1})] \end{aligned} \tag{3.15.1}$$

and

$$0 = B^{-t}(3.14.2)B^{-1} \equiv -2B^{-t} Q B^{-1} + \sum_{(i,j)\in M_R} u_{ij}^* B^{-t}(G_j^t G_i + G_i^t G_j)B^{-1} \tag{3.15.2}$$

For verifying the primal feasibility and complementary slackness conditions with respect to (3.13), define

$$s^* = Bx^* + p, \quad \text{so that} \quad x^* = B^{-1}(s^* - p) \tag{3.16.1}$$

and

$$[y^*] = B[w^*]B^t + p(x^*)^t B^t + Bx^* p^t + pp^t = B[w^*]B^t - (pp^t - p(s^*)^t - s^* p^t) \tag{3.16.2}$$

where $[w^*]$ is the matrix representation of w^*, defined similarly as $[y^*]$. Then, consider any constraint of RLT-LP′ evaluated at (s^*, y^*). Recognizing that ss^t in this constraint is replaced by $[y]$, we have, using (3.16.1) and (3.16.2),

$$\begin{aligned} &(g_i + G_i B^{-1} p)(g_j + G_j B^{-1} p) - [(g_i + G_i B^{-1} p)(G_j B^{-1}) + (g_j + G_j B^{-1} p)(G_i B^{-1})]s^* + G_i B^{-1}[y^*]B^{-t} G_j^t \\ &= g_i g_j - (g_j G_i + g_i G_j)B^{-1}(s^* - p) + G_i B^{-1}([y^*] + pp^t - p(s^*)^t - s^* p^t)B^{-t} G_j^t \\ &= g_i g_j - (g_j G_i + g_i G_j)x^* + G_i[w^*]G_j^t \geq 0 \end{aligned}$$

since (x^*, w^*) is primal feasible for RLT-LP. Moreover, this also exhibits that the slack values of constraints in RLT-LP and RLT-LP′ are the same when they are evaluated at (x^*, w^*) and (s^*, y^*), respectively. Hence the complementary slackness conditions for RLT-LP′ follow directly from the KKT conditions of RLT-LP. To complete the proof, we show the equivalence of the optimal objective function values as follows.

$$
\begin{aligned}
v[\text{RLT-LP}'] &= -cB^{-1}p + p^tB^{-t}QB^{-1}p - \sum_{(i,j)\in M_R} u^*_{ij}[(g_i + G_iB^{-1}p)(g_j + G_jB^{-1}p)] \\
&= -\sum_{(i,j)\in M_R} u^*_{ij}g_ig_j - cB^{-1}p - \sum_{(i,j)\in M_R} u^*_{ij}(g_iG_j + g_jG_i)B^{-1}p \\
&\quad + p^tB^{-t}QB^{-1}p - \sum_{(i,j)\in M_R} u^*_{ij}[(G_iB^{-1}p)(G_jB^{-1}p) + (G_jB^{-1}p)(G_iB^{-1}p)]/2 \\
&= -\sum_{(i,j)\in M_R} u^*_{ij}g_ig_j + \left[-c - \sum_{(i,j)\in M_R} u^*_{ij}(g_iG_j + g_jG_i)\right]B^{-1}p \\
&\quad + p^tB^{-t}\left[Q - \frac{1}{2}\sum_{(i,j)\in M_R} u^*_{ij}(G_i^tG_j + G_j^tG_i)\right]B^{-1}p \\
&= -\sum_{(i,j)\in M_R} u^*_{ij}g_ig_j \qquad \text{(by (3.14.1) and (3.14.2))} \\
&= v[\text{RLT-LP}]
\end{aligned}
$$

This completes the proof. ■

3.4 Eigen Transformation (RLT-NLPE)

In this section, we propose using a particular linear transformation based on the eigenstructure of the quadratic objective function. Although we know that the first-level RLT value is invariant under linear transformations, our motivation here is to obtain a favorable RLT structure that permits the use of nonlinear convex constraints within the lower bounding problem. Note that under a regular application of RLT, all the resulting nonlinear terms are linearly approximated. However, if we can isolate certain "noncomplicating" convex terms in the problem, and preserve these as nonlinear, then the resulting convex program can yield a tighter relaxation than its linear counterpart. In addition, if the solution of this lower bounding problem can be shown to be no more complicated than the totally linearized problem, we will have a superior bounding strategy.

The particular linear transformation we consider is based on the eigen decomposition of the matrix Q. Notice that since Q is a symmetric matrix, its eigenvalues are real, and it possesses a full set of n linearly independent eigenvectors. Therefore, the diagonalization of Q is well defined for our purposes.

Let $Q = PDP^t$, where D is a diagonal matrix whose diagonal elements correspond to the eigenvalues λ_i, $i = 1,\dots, n$, of Q, and the columns of P correspond to orthonormal eigenvectors of Q (see Golub and Van Loan (1989), for example). Define

$$x = Pz, \qquad \text{so that} \qquad z = P^t x \tag{3.17}$$

By substituting (3.17) in QP, we obtain the following eigen-transformed problem.

EQP : Minimize $\quad cPz + z^t Dz \qquad\qquad (3.18.1)$

subject to $APz \leq b$ (3.18.2)

$$l \leq Pz \leq u \tag{3.18.3}$$

$$z \quad \text{unrestricted} \tag{3.18.4}$$

where $l = (l_1, \ldots, l_n)^t$, and $u = (u_1, \ldots, u_n)^t$. As in Section 3.2, let us represent the constraints (3.18.2)-(3.18.3) collectively by $F_i z \leq f_i$, $i = 1, \ldots, m + 2n$. Considering all possible quadratic constraint-factor products, and then defining a new variable for each distinct nonlinear term according to the substitution

$$y_{kl} = z_k z_l \quad \forall\ 1 \leq k \leq l \leq n, \tag{3.19}$$

we generate the first-level RLT problem. But this time, we further tighten the relaxation by including the following nonlinear convex constraints in the problem:

$$z_k^2 \leq y_{kk} \quad \forall\ k = 1, \ldots, n \tag{3.20}$$

This leads to the following lower bounding convex programming problem:

RLT-NLPE: Minimize $\sum_{k=1}^{n} (cP)_k z_k + \sum_{k=1}^{n} \lambda_k y_{kk}$ (3.21.1)

subject to $[(f_i - F_i z)(f_j - F_j z)]_\ell \geq 0 \quad \forall\ 1 \leq i \leq j \leq m + 2n$ (3.21.2)

$$z_k^2 \leq y_{kk} \qquad \forall\ k = 1, \ldots, n \tag{3.21.3}$$

For each $k = 1, \ldots, n$, the nonlinear constraints (3.21.3) prevent the RLT variable y_{kk} from taking values lower than the nonlinear term z_k^2 that it represents. This is especially useful for the variables y_{kk} that are associated with positive eigenvalues ($\lambda_k > 0$) as their cost coefficients, because the minimization process that is attempting to reduce the values of these variables is prohibited from lowering these values below the corresponding z_k^p values. Since just the opposite is true for the variables having negative eigenvalue cost coefficients, it would be more appropriate to include upper bounding types of constraints for such y_{kk}-variables. However, this would lead to reverse-convex constraints. Nonetheless, (3.21.3) might still be useful if some y_{kk}-variable that has a negative cost coefficient ($\lambda_k < 0$) needs to be lowered in value below z_k^p for the sake of some other variables that have more advantageous influence on the objective function. This would again be prevented by the presence of (3.21.3).

3.5 *Inclusion of Suitable Nonlinear Constraints in RLT-LP to Derive RLT-NLP*

Observe that in RLT-LP, the following linear RLT constraints

$$[(x_k - l_k)^2]_\ell \geq 0, \quad [(u_k - x_k)^2]_\ell \geq 0, \quad [(x_k - l_k)(u_k - x_k)]_\ell \geq 0, \quad k = 1,\dots, n \tag{3.22}$$

approximate the relationship $w_{kk} = x_k^2$ over the interval $l_k \leq x_k \leq u_k$, for $k = 1,\dots, n$. Motivated by RLT-NLPE, we propose to replace (3.22) by the nonlinear constraints

$$x_k^2 \leq w_{kk} \leq (u_k + l_k)x_k - u_k l_k, \quad l_k \leq x_k \leq u_k, \quad k = 1,\dots, n \tag{3.23}$$

Notice that the upper bounding linear function in (3.23) is precisely the last constraint in (3.22). The improvement via (3.23), which implies (3.22), appears in the case when the problem RLT-LP tries to reduce the value of w_{kk} for some $k \in \{1,\dots, n\}$ in the relaxed solution. Note that first two constraints in (3.22) merely *approximate* the function $w_{kk} = x_k^2$ from below via tangential supports at the points l_k and u_k. On the other hand, since (3.23) produces the exact lower envelope, it is equivalent to having an additional tangential support at the optimal point included within RLT-LP. Therefore, the enhancement (3.23) corresponds to a tighter bounding nonlinear problem than the linear program RLT-LP. The resulting convex nonlinear problem is explicitly given below.

RLT-NLP :

$$\text{Minimize} \quad \sum_{k=1}^{n} c_k x_k + \sum_{k=1}^{n} q_{kk} w_{kk} + \sum_{k=1}^{n-1} \sum_{l=k+1}^{n} 2q_{kl} w_{kl} \tag{3.24.1}$$

$$\text{subject to} \quad [(b_i - a_i x)(b_j - a_j x)]_\ell \geq 0 \qquad \forall 1 \leq i \leq j \leq m \tag{3.24.2}$$

$$[(x_k - l_k)(b_i - a_i x)]_\ell \geq 0 \qquad k = 1,\ldots, n, \quad i = 1,\ldots, m \tag{3.24.3}$$

$$[(u_k - x_k)(b_i - a_i x)]_\ell \geq 0 \qquad k = 1,\ldots, n, \quad i = 1,\ldots, m \tag{3.24.4}$$

$$[(u_k - x_k)(u_l - x_l)]_\ell \geq 0 \qquad 1 \leq k < l \leq n \tag{3.24.5}$$

$$[(x_k - l_k)(u_l - x_l)]_\ell \geq 0 \qquad 1 \leq k < l \leq n \tag{3.24.6}$$

$$[(x_k - l_k)(x_l - l_l)]_\ell \geq 0 \qquad 1 \leq k < l \leq n \tag{3.24.7}$$

$$[(u_k - x_k)(x_l - l_l)]_\ell \geq 0 \qquad 1 \leq k < l \leq n \tag{3.24.8}$$

$$x_k^2 \leq w_{kk} \leq (u_k + l_k)x_k - u_k l_k \qquad k = 1,\ldots, n \tag{3.24.9}$$

$$l_k \leq x_k \leq u_k \qquad k = 1,\ldots, n \tag{3.24.10}$$

REMARK 2: Note that RLT-NLP and RLT-NLPE are no longer equivalent relaxations. Although RLT-NLPE usually turns out to yield a tighter representation because the additional nonlinear constraints provide a better support for the *"convex" variables* (those associated with $\lambda_k > 0$), the loss in structure due to increased density in the bounding constraints inhibits the development of an efficient solution scheme. Hence, we choose to implement RLT-NLP. (See Section 3.9 for some related computational results.)

3.6 A Lagrangian Dual Approach for Solving RLT Relaxations

Given a problem QP that has n variables and (upto) $m + 2n$ constraints, the corresponding first-level RLT problem has $n(n+1)/2$ additional variables, and a total of $(m+2n)(m+2n+1)/2$ constraints. It is clear that the size of RLT-LP, or even RLT-NLP, gets quite large as the size of QP increases. At each node of the branch-and-bound algorithm, if RLT-NLP is to be used as a device for deriving a lower bound on the node subproblem, then this would require us to solve large-scale nonlinear programs quite often. However, we do not need to solve RLT-NLP exactly. If we can obtain a tight lower bound on RLT-NLP relatively easy, then we can trade-off between the quality of the bound and the effort necessary to obtain it. For this purpose, we propose to use a Lagrangian relaxation of RLT-NLP (see Fisher, 1981, for example), and solve -not necessarily exactly- the Lagrangian Dual Problem **LD-RLT-NLP.**

To define the proposed Lagrangian dual for RLT-NLP, we dualize the constraints (3.24.2)-(3.24.6), and consequently the remaining constraints (3.24.7)-(3.24.10) comprise the Lagrangian subproblem constraints (see Fisher, 1981). Note that the foregoing linearized bound-factor product constraints (3.24.7)-(3.24.8) in the subproblem yield lower and upper bounding linear functions for the linearized cross product terms w_{kl}, for all $1 \le k < l \le n$ over $l_k \le x_k \le u_k$, $k = 1, \dots, n$. These constraints can be expressed in open form as follows:

$$l_l x_k + l_k x_l - l_k l_l \leq w_{kl} \leq l_l x_k + u_k x_l - u_k l_l \quad \forall\ 1 \leq k < l \leq n \tag{3.25}$$

Hence, in order to solve the Lagrangian subproblem, depending on the signs of the coefficients of each w_{kl} variable in the Lagrangian subproblem objective function, we first replace this variable in the objective function appropriately by either its lower or upper bounding function in terms of x as defined in (3.25). The resulting subproblem objective function is then given in terms of the variables (x_k, and w_{kk}, for $k = 1,\ldots, n$) alone, and the subproblem constraints are now defined by (3.24.9) and (3.24.10). This reduced separable Lagrangian dual subproblem can be stated as follows:

$$\begin{aligned}&\text{Minimize } \{\sum_{k=1}^{n} \hat{c}_k x_k + \sum_{k=1}^{n} \hat{q}_{kk} w_{kk}\text{: (3.24.9), (3.24.10)}\}\\ &= \sum_{k=1}^{n} \text{Minimize } \{\hat{c}_k x_k + \hat{q}_{kk} w_{kk}\text{: } x_k^2 \leq w_{kk} \leq (u_k + l_k) x_k - u_k l_k,\ \ l_k \leq x_k \leq u_k\}\end{aligned} \tag{3.26}$$

Suppose that instead of the constraints in (3.26), we use the restrictions

$$w_{kk} = x_k^2, \quad l_k \leq x_k \leq u_k, \quad k = 1,\ldots, n \tag{3.27}$$

to obtain the reduced Lagrangian dual subproblem in the following more favorable form:

$$\text{Minimize } \{\sum_{k=1}^{n} \hat{c}_k x_k + \sum_{k=1}^{n} \hat{q}_{kk} w_{kk}\text{: (3.27)}\} = \sum_{k=1}^{n} \underset{l_k \leq x_k \leq u_k}{\text{Minimize}} \{\hat{c}_k x_k + \hat{q}_{kk} x_k^2\} \tag{3.28}$$

The following proposition asserts that the simpler problem (3.28) equivalently solves (3.26).

PROPOSITION 5: Problem (3.26) is equivalently solved via Problem (3.28).

Proof: For any $k \in \{1, \ldots, n\}$, if $\hat{q}_{kk} > 0$, then in (3.26), w_{kk} can be replaced by x_k^2 in the objective function, which makes it equivalent to (3.28). If $\hat{q}_{kk} \leq 0$, then in (3.26), if w_{kk} is replaced by $(u_k + l_k)x_k - u_k l_k$, the revised objective function becomes a linear function of the variable x_k. Consequently, an optimal value for x_k occurs at either of its bounds, and so, the constraints on w_{kk} in (3.26) are satisfied as equalities at optimality. Examining the same case for (3.28), we similarly obtain a concave univariate quadratic minimization problem over simple bounds, and so, an optimal value for x_k occurs at either of the bounds. Also, since $(u_k + l_k)x_k - u_k l_k = x_k^2$ at either bound, both (3.28) and (3.26) produce the same optimal solutions. This completes the proof. ■

3.7 Constraint Selection Strategy

Given the various types of RLT constraints that can be generated, we now view some of these constraints more critically, and try to reduce their number by eliminating those that might not contribute significantly to determining an optimal solution to RLT-LP. If we can guess which constraints are more relevant in determining an optimum among the constraints in (3.4.2), then we can discard the rest and still derive a good bound on the problem. The starting point for the constraint selection scheme is the eigen transformed problem EQP, which is convenient because of its separable objective function. Consider the concave variables in the objective function in (3.18), that is, the variables whose quadratic terms are associated with the negative eigenvalues of the matrix Q as their cost coefficients. We would like to have upper bounding constraints on the RLT variables representing these concave quadratic terms, since the minimization problem (3.21) is more likely to increase their values. Denoting the index set of *"concave" variables* by $N_v \equiv \{k: \lambda_k < 0, k = 1,\ldots, n\}$, let us re-organize each constraint $[(f_i - F_i z)(f_j - F_j z)]_\ell \geq 0$ from (3.21.2) as follows:

$$\sum_{k \in N_v} - F_{ik} F_{jk} y_{kk} \leq [\text{ the rest of the constraint}] \tag{3.29}$$

Assuming that the original problem has been scaled so that all variables are roughly commensurate with each other, if the condition

$$\left(\sum_{k \in N_v} - F_{ik} F_{jk} \right) \geq 0 \tag{3.30}$$

holds, then on the left-hand side of the inequality (3.29), the positive coefficients have at least as much weight as the negative coefficients. In this case, it is more likely that (3.29) produces relatively strong upper bounds on the y_{kk}-variables that have positive coefficients, i.e., for which $-F_{ik}F_{jk} > 0$, $k \in N_v$. This is useful since the objective function coefficients λ_k of these y_{kk} variables are negative.

Therefore, we suggest generating only those RLT constraints that satisfy the condition (3.30), and suppressing the rest of them. The corresponding RLT constraints in the previous representation RLT-LP are accordingly retained, while the remaining are discarded. Notice that once a selected RLT constraint in RLT-NLPE is tagged by its associated indices (i, j) in (3.21.2), one can keep track of this constraint for the RLT application in any affine transformed representation of QP, as long as the ordering of the original constraints $Gx \le g$ are maintained across all representations.

In the eigen-space representation, the condition (3.30) concentrates on the concave variables, and it is likely that the RLT constraints that relate the convex terms y_{kk} (that are associated with positive eigenvalues as their cost coefficients) with their corresponding variables z_k, might have been discarded. In RLT-NLPE, this problem is taken care of by the nonlinear convex constraints (3.21.3). However, in any other space representation of QP, we may not be able to conveniently generate a similar set of structured nonlinear constraints that provide this function. Nevertheless, the nonlinear constraints (3.27), or their convex equivalent (3.23), in RLT-NLP can be expected to perform satisfactorily for this purpose. So, although the constraint selection process takes place in the eigen-transformed space, we can set up and solve the Lagrangian dual to the reduced RLT-NLP in the original space. Let us call this reduced relaxation having only selected constraints as **RLT-NLP(SC)**.

Observe that we can no longer guarantee that the original constraints are implied by the selected RLT constraints, so, we include the original constraints in the reduced RLT problem. However, for any functional constraint $a_i x \le b_i$, $i \in \{1, \ldots, m\}$, if both $[(x_k - l_k)(b_i - a_i x)]_\ell \ge 0$, and $[(u_k - x_k)(b_i - a_i x)]_\ell \ge 0$ for any $k \in \{1, \ldots, n\}$ are present in RLT-NLP(SC), then by Proposition 1, $a_i x \le b$ is implied; therefore, it is not required to include this constraint in the reduced problem.

3.8 Additional RLT Constraints

In this section, we present two new classes of constraints to strengthen the first-level RLT. The first class of constraints involves the projection of the third-order bound-factor products onto the quadratic space. Here, we take the products of the bound-factors $(x_k - l_k) \geq 0$ and $(u_k - x_k) \geq 0$, $k = 1,\ldots, n$, three at a time, and then linearize them to generate third-order constraints. For example, this strategy would generate constraints of the following type:

$$[(x_k - l_k)^3]_\ell \geq 0, \quad [(x_k - l_k)(u_l - x_l)^2]_\ell \geq 0, \quad [(x_k - l_k)(x_l - l_l)(u_j - x_j)]_\ell \geq 0$$

These third-order constraints are then projected onto the quadratic space by surrogating appropriate pairs in order to eliminate the linearized third degree terms. For example, the following surrogates conduct such an operation:

$$[(x_k - l_k)(x_l - l_l)(u_j - x_j)]_\ell + [(u_k - x_k)(x_l - l_l)(u_j - x_j)]_\ell \geq 0 \tag{3.31.1}$$

$$[(x_k - l_k)^2(u_l - x_l)]_\ell + [(u_k - x_k)^2(x_l - l_l)]_\ell \geq 0 \tag{3.31.2}$$

$$[(x_k - l_k)(x_l - l_l)(x_j - l_j)]_\ell + [(u_k - x_k)(u_l - x_l)(u_j - x_j)]_\ell \geq 0 \tag{3.31.3}$$

Some of the resulting constraints are implied by second-order bound-factor product based RLT constraints, while others yield new restrictions. For example, (3.31.1) is implied by $[(x_l - l_l)(u_j - x_j)]_\ell \geq 0$, whereas (3.31.3) yields a new constraint. We can indeed identify all the new classes of such constraints generated in this process. This operation can also be viewed as an application of Fourier-Motzkin method for solving systems of linear inequalities (see Chvatal, 1983, pp. 241, for an introduction).

The second class of constraints is based on the squares of the *scaled differences* of bound-factors. A typical example of this set is the constraint

$$\left[\left(\frac{x_k - l_k}{u_k - l_k} - \frac{x_l - l_l}{u_l - l_l}\right)^2\right]_\ell \geq 0 \tag{3.32}$$

Along with the second-order bound-factor product based RLT constraints, these new classes of constraints imply some more of the constraints described above that are obtained by projecting third-order bound-factor products. For example, (3.32) along with $[(x_l - l_l)(u_l - x_l)]_\ell \geq 0$ implies (3.31.2). It can be shown that each squared bound-factor difference based constraint eliminates two constraints based on projecting third-order bound-factor products, and hence, the inclusion of these constraints reduces the total number of additional nonimplied constraints to be used in the first-level RLT.

We now explicitly generate all the nonimplied constraints via the projection of third-order bound-factor products onto the (x, w)-space. As a short-hand notation, let a and A denote, respectively, $(x_i - l_i)$ and $(u_i - x_i)$, for some $i \in \{1, \ldots, n\}$. Also, let us denote by W the set of variables defined for each distinct third degree polynomial term. By construction, third-order bound-factor products yield greater than or equal to type inequalities, and each inequality has a single W-variable with a coefficient of 1 or -1. Hence, the projection of these inequalities onto (x, w)-space is obtained by considering all possible combinations of surrogating *two* inequalities at a time to eliminate such W-variables. We will see that some of the projected inequalities are implied by the RLT constraints based on second-order bound-factor products, as illustrated above.

We first consider the inequalities involving the W-variable $[x_i^3]_\ell$, where $i \in \{1, \ldots, n\}$. The coefficient of $[x_i^3]_\ell$ is $+1$ in $[aaa]_\ell$, $[aAA]_\ell$, and -1 in $[aaA]_\ell$,

$[AAA]_\ell$. The following projections are implied by the second-order RLT constraints.

$$\begin{aligned}&[aaA]_\ell + [aaa]_\ell = (u_i - l_i)[aa]_\ell \geq 0\,, \qquad [aaA]_\ell + [aAA]_\ell = (u_i - l_i)[aA]_\ell \geq 0\\&[AAA]_\ell + [aAA]_\ell = (u_i - l_i)[AA]_\ell \geq 0\end{aligned} \tag{3.33}$$

Hence, $[AAA]_\ell$ + $[aaa]_\ell \geq 0$ gives the only nonimplied projection for this case.

Similarly, the RLT variable $[x_i^2x_j]_\ell$, where $i,j \in \{1,\ldots, n\}$, $i \neq j$, has the coefficient of $+1$ in $[aab]_\ell$, $[aAB]_\ell$, $[AAb]_\ell$, and -1 in $[aaB]_\ell$, $[aAb]_\ell$, $[AAB]_\ell$, where $b \equiv (x_j - l_j)$, and $B \equiv (u_j - x_j)$. Using the same logic as in (3.33), it can be easily shown that the following implied projected constraints are generated upon eliminating W_{ij}.

$$\begin{aligned}&[aaB]_\ell + [aab]_\ell = (u_j - l_j)[aa]_\ell \geq 0\,, \qquad [aaB]_\ell + [aAB]_\ell = (u_i - l_i)[aB]_\ell \geq 0\\&[aAb]_\ell + [aab]_\ell = (u_i - l_i)[ab]_\ell \geq 0\,, \qquad [aAb]_\ell + [aAB]_\ell = (u_j - l_j)[aA]_\ell \geq 0\\&[aAb]_\ell + [AAb]_\ell = (u_i - l_i)[Ab]_\ell \geq 0\,, \qquad [AAB]_\ell + [aAB]_\ell = (u_i - l_i)[AB]_\ell \geq 0\\&[AAB]_\ell + [AAb]_\ell = (u_j - l_j)[AA]_\ell \geq 0\end{aligned} \tag{3.34}$$

Hence, $[aaB]_\ell$ + $[AAb]_\ell \geq 0$ and $[AAB]_\ell$ + $[aab]_\ell \geq 0$ give the only corresponding nonimplied projections. Finally, the variable $[x_ix_jx_k]_\ell$, where $i, j, k \in \{1,\ldots, n\}$, $i < j < k$, appears with the coefficient $+1$ in $[abc]_\ell$, $[aBC]_\ell$, $[AbC]_\ell$, $[ABc]_\ell$, and with the coefficient -1 in $[abC]_\ell$, $[aBc]_\ell$, $[Abc]_\ell$, $[ABC]_\ell$, where $c \equiv (x_k - l_k)$ and $C \equiv (u_k - x_k)$. For this case, the implied projections are as follows:

$$\begin{array}{ll}
[abc]_\ell + [abC]_\ell = (u_k - l_k)[ab]_\ell \geq 0\,, & [abc]_\ell + [aBc]_\ell = (u_j - l_j)[ac]_\ell \geq 0 \\
[abc]_\ell + [Abc]_\ell = (u_i - l_i)[bc]_\ell \geq 0\,, & [aBC]_\ell + [abC]_\ell = (u_j - l_j)[aC]_\ell \geq 0 \\
[aBC]_\ell + [aBc]_\ell = (u_k - l_k)[aB]_\ell \geq 0\,, & [aBC]_\ell + [ABC]_\ell = (u_i - l_i)[BC]_\ell \geq 0 \\
[AbC]_\ell + [abC]_\ell = (u_i - l_i)[bC]_\ell \geq 0\,, & [AbC]_\ell + [Abc]_\ell = (u_k - l_k)[Ab]_\ell \geq 0 \\
[AbC]_\ell + [ABC]_\ell = (u_j - l_j)[AC]_\ell \geq 0\,, & [ABc]_\ell + [aBc]_\ell = (u_i - l_i)[Bc]_\ell \geq 0 \\
[ABc]_\ell + [Abc]_\ell = (u_j - l_j)[Ac]_\ell \geq 0\,, & [ABc]_\ell + [ABC]_\ell = (u_k - l_k)[AB]_\ell \geq 0
\end{array} \quad (3.35)$$

Hence, the only nonimplied projections for this case are

$$\begin{array}{ll}
[abc]_\ell + [ABC]_\ell \geq 0, & [aBC]_\ell + [Abc]_\ell \geq 0 \\
[AbC]_\ell + [aBc]_\ell \geq 0, & [ABc]_\ell + [abC]_\ell \geq 0
\end{array} \quad (3.36)$$

To summarize, thus far, the set of nonimplied constraints generated by the foregoing projections of the third-order bound-factor products are as follows:

$$\left[(x_i - l_i)^3 + (u_i - x_i)^3\right]_\ell \geq 0 \qquad i = 1,\dots, n \quad (3.37.1)$$

$$\left[(x_i - l_i)^2(u_j - x_j) + (u_i - x_i)^2(x_j - l_j)\right]_\ell \geq 0 \qquad 1 \leq i, j \leq n, i \neq j \quad (3.37.2)$$

$$\left[(x_i - l_i)^2(x_j - l_j) + (u_i - x_i)^2(u_j - x_j)\right]_\ell \geq 0 \qquad 1 \leq i, j \leq n, i \neq j \quad (3.37.3)$$

$$[(x_i - l_i)(x_j - l_j)(x_k - l_k) + (u_i - x_i)(u_j - x_j)(u_k - x_k)]_\ell \geq 0 \qquad 1 \leq i < j < k \leq n \quad (3.37.4)$$

$$[(x_i - l_i)(u_j - x_j)(u_k - x_k) + (u_i - x_i)(x_j - l_j)(x_k - l_k)]_\ell \geq 0 \qquad 1 \leq i < j < k \leq n \quad (3.37.5)$$

$$[(u_i - x_i)(x_j - l_j)(u_k - x_k) + (x_i - l_i)(u_j - x_j)(x_k - l_k)]_\ell \geq 0 \qquad 1 \leq i < j < k \leq n \quad (3.37.6)$$

$$[(u_i - x_i)(u_j - x_j)(x_k - l_k) + (x_i - l_i)(x_j - l_j)(u_k - x_k)]_\ell \geq 0 \qquad 1 \leq i < j < k \leq n \quad (3.37.7)$$

Next, consider (3.37.1) written in its nonlinear form, as follows, for some $i \in \{1,\dots, n\}$.

$$x_i^2 \geq (u_i + l_i)x_i - (u_i^2 + u_i l_i + l_i^2)/3 \tag{3.38}$$

Notice that (3.38) does not hold as an equality for any $x_i \in [l_i, u_i]$. Hence, we would like to translate the linear underestimating function $(u_i + l_i)x_i - (u_i^2 + u_i l_i + l_i^2)/3$ upward until it supports the graph of x_i^2. Consider the tangent line to x_i^2 at $x_i = (u_i + l_i)/2$. By the definition of convexity, we have

$$x_i^2 \geq (u_i + l_i)x_i - (u_i + l_i)^2/4 \tag{3.39}$$

Since, $(u_i^2 + u_i l_i + l_i^2)/3 - (u_i + l_i)^2/4 = (u_i - l_i)^2/12 \geq 0$, (3.39) strictly dominates (3.38), and is therefore used to replace it.

We now derive a second class of new RLT constraints, based on the squared differences of the scaled bound-factor products. This new class of constraints imply (3.37.2) and (3.37.3), while including only half as many members.

$$\left[\left(\frac{x_i - l_i}{u_i - l_i} - \frac{x_j - l_j}{u_j - l_j}\right)^2\right]_\ell \geq 0 \quad 1 \leq i < j \leq n \tag{3.40.1}$$

$$\left[\left(1 - \frac{x_i - l_i}{u_i - l_i} - \frac{x_j - l_j}{u_j - l_j}\right)^2\right]_\ell \geq 0 \quad 1 \leq i < j \leq n \tag{3.40.2}$$

By surrogating appropriate second-order bound-factor products with (3.40.1) and (3.40.2), respectively, we can exhibit that (3.37.2) and (3.37.3) are implied as follows.

$$(u_i - l_i)^2([(x_j - l_j)(u_j - x_j)]_\ell \geq 0) + (3.40.1) = (u_j - l_j) \cdot (3.37.2) \tag{3.41.1}$$

$$(u_i - l_i)^2([(x_j - l_j)(u_j - x_j)]_\ell \geq 0) + (3.40.2) = (u_j - l_j) \cdot (3.37.3) \tag{3.42.2}$$

REMARK 3: (a) The remaining constraints in the set (3.37), namely, the constraints (3.37.4)-(3.37.7), are not implied by the second-order constraints, nor do they imply the second-order bound-factor products. This can be shown by minimizing the left-hand side of a constraint from (3.42.4)-(3.42.7) subject to the constraints from second-order products, and vice versa. For example, by using the scaled bounds $(l_i, u_i) = (0, 1)$ for all variables $i = 1, \dots, n$, consider the following problem that tests the implication of a member from (3.37.4).

$$
\begin{aligned}
\text{Minimize } \ & w_{ij} + w_{ik} + w_{jk} - x_i - x_j - x_k + 1 \\
\text{subject to } \ & w_{ij} \geq 0, \quad w_{ik} \geq 0, \quad w_{jk} \geq 0 \\
& w_{ij} - x_i - x_j \geq -1, \quad w_{ik} - x_i - x_j \geq -1, \quad w_{jk} - x_j - x_k \geq -1 \\
& x_i - w_{ij} \geq 0, \quad x_i - w_{ik} \geq 0, \quad x_j - w_{jk} \geq 0 \\
& x_j - w_{ij} \geq 0, \quad x_k - w_{ik} \geq 0, \quad x_k - w_{jk} \geq 0
\end{aligned}
$$

The optimal objective function value is -0.5, which being negative, asserts that this constraint (3.37.4) is not implied by the second-order bound-factor products.

(b) Suppose that an x-variable is fixed at some value, say $x_i = \bar{x}_i$, and that we replace w_{ij} and w_{ik} by $\bar{x}_i x_j$ and $\bar{x}_i x_k$, respectively, in (3.37). Then, the resulting constraints in (3.37.4)-(3.37.7), are indeed implied by the second-order bound-factor products.

(c) In the spirit of the constraints (3.40.1)-(3.40.2), one can also generate RLT constraints based on the squared (functional) constraint-factor differences.

When expanded and linearized, the complete set of new nonimplied constraints (3.39), (3.40.1), (3.40.2) and (3.37.4)-(3.37.7) can be stated as follows.

$$w_{ii} - (u_i + l_i)x_i + (u_i + l_i)^2/4 \geq 0 \qquad i = 1,\ldots, n \tag{3.43.1}$$

$$\begin{aligned}-2(u_i - l_i)(u_j - l_j)w_{ij} + (u_j - l_j)^2 w_{ii} + (u_i - l_i)^2 w_{jj} + 2(u_j - l_j)(u_i l_j - l_i u_j)x_i \\ + 2(u_i - l_i)(l_i u_j - u_i l_j)x_j + (l_i u_j - u_i l_j)^2 \geq 0 \qquad 1 \leq i < j \leq n\end{aligned} \tag{3.43.2}$$

$$\begin{aligned}2(u_i - l_i)(u_j - l_j)w_{ij} + (u_j - l_j)^2 w_{ii} + (u_i - l_i)^2 w_{jj} + 2(u_j - l_j)(l_i l_j - u_i u_j)x_i \\ + 2(u_i - l_i)(l_i l_j - u_i u_j)x_j + (l_i l_j - u_i u_j)^2 \geq 0 \qquad 1 \leq i < j \leq n\end{aligned} \tag{3.43.3}$$

$$\begin{aligned}(u_k - l_k)w_{ij} + (u_j - l_j)w_{ik} + (u_i - l_i)w_{jk} - (u_j u_k - l_j l_k)x_i - (u_i u_k - l_i l_k)x_j - (u_i u_j - l_i l_j)x_k \\ + (u_i u_j u_k - l_i l_j l_k) \geq 0 \qquad 1 \leq i < j < k \leq n\end{aligned} \tag{3.43.4}$$

$$\begin{aligned}-(u_k - l_k)w_{ij} - (u_j - l_j)w_{ik} + (u_i - l_i)w_{jk} + (u_j u_k - l_j l_k)x_i + (l_i u_k - u_i l_k)x_j + (l_i u_j - u_i l_j)x_k \\ + (u_i l_j l_k - l_i u_j u_k) \geq 0 \qquad 1 \leq i < j < k \leq n\end{aligned} \tag{3.43.5}$$

$$\begin{aligned}-(u_k - l_k)w_{ij} + (u_j - l_j)w_{ik} - (u_i - l_i)w_{jk} + (l_j u_k - u_j l_k)x_i + (u_i u_k - l_i l_k)x_j + (u_i l_j - l_i u_j)x_k \\ + (l_i u_j l_k - u_i l_j u_k) \geq 0 \qquad 1 \leq i < j < k \leq n\end{aligned} \tag{3.43.6}$$

$$\begin{aligned}(u_k - l_k)w_{ij} - (u_j - l_j)w_{ik} - (u_i - l_i)w_{jk} + (u_j l_k - l_j u_k)x_i + (u_i l_k - l_i u_k)x_j + (u_i u_j - l_i l_j)x_k \\ + (l_i l_j u_k - u_i u_j l_k) \geq 0 \qquad 1 \leq i < j < k \leq n\end{aligned} \tag{3.43.7}$$

3.9 A Preliminary Computational Comparison of the Bounding Problems

To evaluate and compare the different approaches for generating and solving a first-level RLT relaxation, we performed some preliminary empirical experiments using test problems from the literature (see Appendix A). Using three bilinear programming problems (Problems 1-3: BLP1, BLP2, BLP3) from Al-Khayyal and Falk (1983), and two concave programming problems (Problems 4,5: CQP1, CQP2) from Floudas and Pardalos (1990), we solved the first-level linear RLT problem

RLT-LP, the nonlinear RLT problem RLT-NLP, the eigen space first-level nonlinear RLT problem RLT-NLPE, and the reduced RLT-NLP problem RLT-NLP(SC) that uses only selected constraints, as well as the Lagrangian dual problems corresponding to the latter three (denoted by the prefix **LD-**). For these computations, the linear and nonlinear programs were solved using GAMS along with the solver MINOS 5.2, and a Fortran code was written for solving the Lagrangian dual problems as per Sections 4.1 and 4.2 of the next chapter. All the runs were conducted on an IBM 3090 mainframe computer. Table 1 reports the resulting lower bounds obtained along with the solution times required to generate these bounds.

RLT-LP yields lower bounds very close to the actual global minimum for all the problems except for BLP1 (BLP2 and BLP3 are solved exactly). RLT-NLPE, where we have included the nonlinear constraints $z_k^2 \leq y_{kk}$ only for $k \in \{1, \ldots, n\}$ such that $\lambda_k \geq 0$, solves all problems to (near) optimality. However, the computational effort is considerably increased since this lower bounding problem is a nonlinear programming problem. When we include the nonlinear constraints $z_k^2 \leq y_{kk}$ for all $k \in \{1, \ldots, n\}$ in RLT-NLPE, the imposed resource limit of 1000 cpu seconds for GAMS is exceeded for BLP2. On the other hand, we were able to solve RLT-NLP, even while including all n nonlinear constraints, and it produced solutions comparable to those obtained by RLT-NLPE. Reconsidering RLT-LP, but this time, including the additional two classes of constraints derived in Section 3.8, improves the lower bound for BLP1 to -1.125, and that for CQP2 to -39.47, the latter of which is greater than any of the bounds reported in Table 1. However, the computational effort for CQP2 increased to 13.36 cpu seconds.

Although the Lagrangian dual problem LD-RLT-NLP should give the same lower bound as does RLT-NLP, the bounds for BLP3 and CQP2 are somewhat worsened due to the inherent difficulty in solving nondifferentiable optimization

Table 1. Comparison of RLT schemes

Problem	m	n	Known $v[QP]$	v[RLT-LP]	cpu secs.
BLP1	2	2	-1.083	-1.50	0.042
BLP2	10	10	-45.38	-45.38	3.75
BLP3	13	10	-794.86	-794.86	22.74
CQP1	11	10	-267.95	-268.02	9.72
CQP2	5	10	-39.00	-39.83	3.23

Problem	v[RLT-NLPE]	cpu secs.	v[LD-RLT-NLPE]	cpu secs.
BLP1	-1.083	0.086	-1.097	0.09
BLP2	-45.38	31.23	-69.17	0.81
BLP3	-794.76	48.55	-11586.88	0.97
CQP1	-268.02	12.16	-269.87	0.85
CQP2	-39.83	4.69	-43.38	0.65

Problem	v[RLT-NLP]	cpu secs.	v[LD-RLT-NLP]	cpu secs.
BLP1	-1.089	0.105	-1.089	0.02
BLP2	-45.38	9.79	-46.10	0.43
BLP3	-794.86	78.41	-829.52	0.57
CQP1	-268.02	13.30	-269.83	0.48
CQP2	-39.83	4.69	-43.93	0.25

Problem	v[RLT-NLP(SC)]	cpu secs.	$\frac{\text{used}}{\text{total}}$ constr.	v[LD-RLT-NLP(SC)]
BLP1	-1.089	0.091	15 / 21	-1.089
BLP2	-45.38	6.07	339 / 465	-45.62
BLP3	-806.53	32.84	387 / 564	-841.60
CQP1	-268.02	13.61	352 / 497	-269.68
CQP2	-40.10	2.54	261 / 320	-42.99

Legend: (m, n) = size of the problem QP, $v[\cdot]$ = optimal solution of problem ($\cdot$), cpu secs. = cpu seconds to solve the problem on an IBM 3090 computer.

problems. Nevertheless, the attractive computational times, especially when the problem size increases, makes this method our first choice to be used in the proposed branch-and-bound algorithm.

In adopting the Lagrangian dual approach of Section 3.6 for RLT-NLPE, the $2n(n-1)$ RLT constraints that are generated using the constraints $l \leq Pz \leq u$ in EQP were dualized. Implied simple bounds on z-variables were computed by minimizing and then maximizing each row of $z = P^t x$ over $l \leq x \leq u$ to obtain counterparts of both the constraints (3.25) and the subproblem (3.28) in z-space (see Section 4.5.4 for details). Ideally, we would like to use LD-RLT-NLPE as the bounding problem for the overall algorithm. However, due to its dense structure, LD-RLT-NLPE tends to perform poorly as for problems BLP2 and BLP3, although the bounds for CQP1 and CQP2 are slightly improved over those obtained via LD-RLT-NLP. Upon using the tightest simple bounds on the z-variables that contain the feasible region, the lower bound for BLP3 improved considerably to -1507.64, but it is still 81% lower than that obtained via LD-RLT-NLP.

Applying the constraint selection strategy of Section 3.7 on RLT-NLP we solved the reduced problem RLT-NLP(SC) with reduced computational effort, although at an expense of a 1.5% decrease in the lower bound for BLP3. (Roughly 18-30% of the constraints are deleted by this strategy.) For the same problem, v[LD-RLT-NLP(SC)] has also worsened by 1.5% compared to v[LD-RLT-NLP]. However, for the rest of the problems, LD-RLT-NLP(SC) actually improved the lower bounds. We have also observed that there is only a negligible increase in the required computational effort compared to that consumed by LD-RLT-NLP. Section 4.9 of Chapter 4 reports on computational experiments using RLT-NLP, LD-RLT-NLP, and LD-RLT-NLP(SC) within the proposed branch-and-bound algorithm.

Chapter IV

Implementation Issues for the Quadratic Programming Algorithm

In this chapter, we discuss implementation issues and strategies for a branch-and-bound algorithm for solving indefinite quadratic programming problems. In this algorithm the branching is performed based on the partitioning of the box constraints defined by the simple lower and upper bounds on each variable. Lower and upper bounds on QP are then derived using designed RLT relaxations of the previous chapter.

In the first section, we present a mild modification of the Lagrangian dual formulation proposed in Chapter 3. Besides being more stable, the new formulation also allows us to optimize certain dual variables exactly. The resulting Lagrangian dual problem is approximately solved by using a conjugate subgradient algorithm. The performance of this algorithm is considerably improved by employing a simple scaling of the original problem QP. Exploiting the linearity of the constraints and the separability of the Lagrangian dual subproblem and eigen-transformed problem's objective functions, various simple strategies are devised for range-restricting as well as for tightening the lower bounds in order to enhance the fathoming efficiency of the algorithm. To obtain good quality upper bounds, we designed various heuristic procedures. The branch-and-bound search strategy we adopt is a hybrid of the best-first and the depth-first strategies. Guidelines for other implementation details, along with computational experiments on test problems from the literature are also included in this chapter.

4.1 A Different Lagrangian Dual Formulation

To improve the Lagrangian dual scheme of Section 3.6, we consider a different formulation following the layering strategy proposed by Guignard and Kim (1987), where they showed that by duplicating some variables, a stronger Lagrangian relaxation can be obtained in the context of mixed-integer programming. Recall that in Section 3.6, the constraints (3.25) (or, equivalently (3.24.7) and (3.24.8)) were not dualized in order to obtain lower and upper linear bounding functions on the RLT variables w_{kl}, for all $1 \leq k < l \leq n$. For the same purpose, instead of (3.25), we could have as well chosen to retain the following constraints, which express the constraints (3.24.5) and (3.24.6) in open form, in the Lagrangian subproblem:

$$u_l x_k + u_k x_l - u_k u_l \leq w_{kl} \leq u_l x_k + l_k x_l - l_k u_l \quad \forall\ 1 \leq k < l \leq n \tag{4.1}$$

or for that matter, we could have employed any proper combination from (3.24.5)-(3.24.8) to be left undualized. In fact, we have observed that different choices of such bounding restrictions affect the empirical performance of the

Lagrangian dual optimization, and there does not seem to exist a clear criterion to guide this choice. To overcome this problem, we introduce a new set of duplicate variables

$$w'_{kl} = w_{kl} \qquad \forall 1 \leq k < l \leq n \tag{4.2}$$

in the constraints, and we rewrite (4.1) using these duplicated variables w'_{kl}. Subsequently, we dualize (4.2), while preserving (3.25) and (4.1) within the Lagrangian subproblem, to obtain separate lower and upper bounding functions on the variables w_{kl} and w'_{kl}, for all $1 \leq k < l \leq n$. We can now proceed as described in Section 3.6, but this time, we replace both w_{kl} and w'_{kl} variables with either their upper or lower bounding functions in order to obtain the reduced Lagrangian subproblem (3.26).

4.2 Subgradient Optimization Algorithm For Solving The Lagrangian Dual Problem

The Lagrangian dual problem is a nondifferentiable optimization problem. To solve this problem, we adopt a conjugate subgradient algorithm based on an algorithm due to Sherali and Ulular (1989), which uses an average direction subgradient deflection strategy along with a block-halving step-size strategy. At each iteration of this algorithm, a Lagrangian relaxation problem is solved, and a subgradient of the Lagrangian dual objective function is computed. As a compromise between computational effort and obtaining a good quality approximate solution to the Lagrangian dual problem, we set the iteration limit to 200.

For the sake of convenience in presentation, consider the primal problem stated in the generic form

$$\text{Minimize}\{cx\text{: } Ax \leq b,\, Dx = d,\, x \in X\} \tag{4.3}$$

The Lagrangian dual to this problem is given by

$$\text{Maximize}\{\theta(\omega)\ :\ \omega = (\omega^1, \omega^2),\ \ \omega^1 \geq 0,\ \ \omega^2 \text{ unrestricted}\} \tag{4.4}$$

where

$$\theta(\omega) = \text{Minimum}\{c^t x + \omega^1(b - Ax) + \omega^2(d - Dx)\ :\ x \in X\} \tag{4.5}$$

At $\omega = \omega_k$, let x_k solve (4.5) and denote $\theta_k \equiv \theta(\omega_k)$. Then a subgradient of $\theta(\omega)$ at $\omega = \omega_k$ is given by

$$\xi_k = \begin{pmatrix} b - Ax_k \\ d - Dx_k \end{pmatrix} \tag{4.6}$$

We assume that an upper bound on (4.4), say UB, is available. The step-size parameter β_k used in (4.8) below is adjusted according to the empirical recommendations of Sherali and Ulular (1989) as

$$\beta_k = \begin{cases} 0.75, & \text{if } k \leq 75 \\ 0.50, & \text{if } 76 \leq k \leq 156 \\ 0.25, & \text{if } k \geq 157 \end{cases} \tag{4.7}$$

In the algorithm, we keep track of the best (incumbent) θ value, the corresponding dual values, the norm of the subgradient, and the outgoing direction as the vector $(\theta^{inc}, \omega^{inc}, ||\xi^{inc}||, d^{inc})$. If θ^{inc} does not improve in 10 consecutive iterations, then we reset to the incumbent solution, and halve the other accompanying step-size parameter α_0 in (4.8). This parameter α_0 is initialized at 2.25. The counter l_{no} is used to keep track of the non-improving iterations. The algorithm is run for a maximum number of iterations of 200.

Conjugate Subgradient Optimization Algorithm:

Step 0. Initialization: Set the initial dual variables as $\omega_0 = 0$, and compute the corresponding θ_0 and ξ_0. Set $d_0 = \xi_0$, $(\theta^{inc}, \omega^{inc}, ||\xi^{inc}||, d^{inc}) = (\theta_0, \omega_0, ||\xi_0||, d_0)$, $\alpha = \alpha_0$, the iteration counter $k = 0$, and the non-improving iteration counter $l_{no} = 0$. Proceed to Step 1.

Step 1. Main Iteration: Compute the step-length as

$$\lambda_k = \frac{\alpha \beta_k [\mathrm{UB} - \theta^{inc}]}{||d_k||} \tag{4.8}$$

and determine the new dual values

$$\omega_{k+1} = P_{\omega^1 \geq 0}\left[\omega_k + \lambda_k \frac{d_k}{||d_k||}\right], \quad \text{where} \quad P_{\omega^1 \geq 0}[\omega^1, \omega^2] \equiv [\max\{0, \omega^1\}, \ \omega^2], \tag{4.9}$$

with the max{·} taken componentwise. Compute the corresponding dual objective value θ_{k+1} and the subgradient ξ_{k+1} via (4.5) and (4.6), respectively. The actual direction adopted in moving from ω_k to ω_{k+1} is denoted by $d_k^a = \omega_{k+1} - \omega_k$. The next direction at ω_{k+1} is obtained by using Sherali and Ulular's (1989) average direction deflection strategy as follows:

$$d_{k+1} = \xi_{k+1} + \phi_k d_k^a, \quad \text{where} \quad \phi_k = ||\xi_{k+1}|| \, / \, ||d_k^a|| \tag{4.10}$$

If $\theta_{k+1} > \theta^{inc}$ then go to Step 3. Otherwise, proceed to Step 2.

Note: At any stage of the above procedure, if any of the $||d_k||$, $||d_k^P||$, or $||\xi_k||$ becomes close to zero then the algorithm is terminated with the incumbent as a near-optimal solution.

Step 2. Non-Improving Iteration: $(\theta_{k+1} \leq \theta^{inc})$. Increment $l_{no} = l_{no} + 1$. If $l_{no} < 10$ then go to Step 4. Otherwise, reset $(\theta_{k+1}, \omega_{k+1}, ||\xi_{k+1}||, d_{k+1}) = (\theta^{inc}, \omega^{inc}, ||\xi^{inc}||, d^{inc})$, and halve the step-length factor $\alpha = \alpha/2$. Set $l_{no} = 0$, and go to Step 4.

Step 3. Improvement: $(\theta_{k+1} > \theta^{inc})$. Set $(\theta^{inc}, \omega^{inc}, ||\xi^{inc}||, d^{inc}) = (\theta_{k+1}, \omega_{k+1}, ||\xi_{k+1}||, d_{k+1})$, and reset the non-improving iteration counter $l_{no} = 0$. Proceed to Step 4.

Step 4. New Iteration: If $k \geq 200$, then stop. Otherwise, increment the iteration counter k by 1. If the value of β_k changes according to (4.7), then set $l_{no} = 0$. Go to Step 1.

4.3 Dual Ascent Procedures

After the conjugate subgradient algorithm terminates, to further improve the incumbent solution, we implement two different dual ascent procedures.

First procedure fixes the dual variables having relatively small magnitudes at their current level in the given incumbent solution, and polishes the remaining dual variable values. In our implementation, dual variables having a value less than half of the average magnitude of all the dual values, were fixed at their current values. In the projected space of the remaining dual variables, the conjugate subgradient algorithm is continued for another 50 iterations. The motivation here is to restrict the dimension of the search space to the dual variables that have been evidently predicted to have a greater influence on the problem. In this run, some of the parameters of the algorithm are changed to $\alpha_0 = 0.0125$, and $\beta_k = 1 \ \forall k$.

Finally, another improvement of the Lagrangian dual solution is conducted by finding optimal values for the dual variables associated with the duplicate variable constraints (4.2) one at a time, in a Gauss-Seidel fashion, given the remaining dual variable values. This can be executed with little additional effort, so it provides another quick dual ascent step. The derivation of this method is as follows.

Optimization of Selected Dual Variables:

Let $\gamma_{kl}, \ \forall 1 \leq k < l \leq n$ be the dual variables associated with the constraints (4.2). Considering only one γ_{kl} at a time for some $1 \leq k < l \leq n$, while other dual variables are fixed at their current values, we write the Lagrangian subproblem $\phi(\cdot)$ as a function of γ_{kl}, at the following stage

$$\phi(\gamma_{kl}) = \text{Minimum} \quad p_0 + \sum_{i=1}^{n} (p_{ii}x_i^2 + p_i x_i) + (s_{kl} - \gamma_{kl})w_{kl} + \gamma_{kl}w'_{kl} \tag{4.11.1}$$

$$\text{subject to} \quad l_l x_k + l_k x_l - l_k l_l \leq w_{kl} \leq l_l x_k + u_k x_l - u_k l_l \tag{4.11.2}$$

$$u_l x_k + u_k x_l - u_k u_l \leq w'_{kl} \leq u_l x_k + l_k x_l - l_k u_l \tag{4.11.3}$$

$$l_k \leq x_k \leq u_k \qquad k = 1, \ldots, n \tag{4.11.4}$$

Notice that we have used the nonconvex equality constraints (3.27) in the problem LD-RLT-NLP. After minimizing the part of the objective function that is constant with respect to γ_{kl}, we have

$$\phi(\gamma_{kl}) = \phi_0 + \text{Minimum} \quad p_{kk}x_k^2 + p_{ll}x_l^2 + p_kx_k + p_lx_l + (s_{kl} - \gamma_{kl})w_{kl} + \gamma_{kl}w'_{kl} \qquad \text{subject to} \quad (4.11.2),\ (4.11.3),\ (4.11.4) \tag{4.12}$$

where

$$\phi_0 = \underset{(4.11.4)}{\text{Minimum}}\left\{p_0 + \sum_{\substack{i=1\\ i \neq k,l}}^{n} (p_{ii}x_i^2 + p_ix_i)\right\} \tag{4.13}$$

By considering the negative and positive values of s_{kl} separately, we can eliminate the constraints (4.11.2) and (4.11.3) to obtain a more favorable form for $\phi(\gamma_{kl})$. Below, we consider only the case for $s_{kl} \leq 0$, since the other case $s_{kl} > 0$ can be derived in a similar fashion. Now, assuming that $s_{kl} \leq 0$, $\phi(\gamma_{kl})$ reduces to the following form.

If $\gamma_{kl} < s_{kl}$, then

$$\phi(\gamma_{kl}) = \phi_0 + \underset{(4.11.4)}{\text{minimum}}\{p_{kk}x_k^2 + p_{ll}x_l^2 + p_kx_k + p_lx_l + (s_{kl} - \gamma_{kl})(l_lx_k + l_kx_l - l_kl_l) + \gamma_{kl}(u_lx_k + l_kx_l - l_ku_l)\} \tag{4.14.1}$$

If $s_{kl} \leq \gamma_{kl} < 0$, then

$$\phi(\gamma_{kl}) = \phi_0 + \underset{(4.11.4)}{\text{minimum}}\{p_{kk}x_k^2 + p_{ll}x_l^2 + p_kx_k + p_lx_l + (s_{kl} - \gamma_{kl})(l_lx_k + u_kx_l - u_kl_l) + \gamma_{kl}(u_lx_k + l_kx_l - l_ku_l)\} \tag{4.14.2}$$

If $\gamma_{kl} \geq 0$, then

$$\phi(\gamma_{kl}) = \phi_0 + \underset{(4.11.4)}{\text{minimum}}\{p_{kk}x_k^2 + p_{ll}x_l^2 + p_kx_k + p_lx_l + (s_{kl} - \gamma_{kl})(l_lx_k + u_kx_l - u_kl_l) \quad (4.14.3)$$
$$+ \gamma_{kl}(u_lx_k + u_kx_l - u_ku_l)\}$$

Note that from (4.14), as well as from (4.12), $\phi(\gamma_{kl})$ is a piece-wise concave function of γ_{kl}. Hence, $\phi(\gamma_{kl})$ can be easily maximized by employing an appropriate one dimensional search algorithm where the sign of the subgradients of $\phi(\gamma_{kl})$ can also be utilized. In Appendix B, we propose an exact method for maximizing $\phi(\gamma_{kl})$.

Having optimized $\phi(\gamma_{kl})$, we set up (4.11) for another γ dual variable and optimize the resulting $\phi(\cdot)$. We continue in this fashion considering all the dual variables γ in turn. If there is no significant improvement after a complete cycle, this procedure stops. Otherwise, we start another cycle. A limit of 3 is set on the total number of cycles attempted.

4.4 Additional Features of the Proposed Algorithm

In each of the subsections below, we explain various strategies we have employed in the proposed branch-and-bound algorithm.

4.4.1 Scaling

Our experiments with the conjugate subgradient deflection algorithm have indicated that scaling plays an important role in the performance of this algorithm. Among several scaling methods we tried, including a sophisticated iterative method used in the package MINOS (see Murtagh and Saunders, 1987), a simple one seemed to work well. This method scales the variables such that the lower and upper bounds are mapped onto the hypersquare $[0, 1]^n$. In addition, a row scaling is employed that divides each constraint $a_i x \leq b_i$, $i \in \{1, \ldots, m\}$, by the l_∞-norm of (a_i, b_i). We employ this scaling scheme for each node subproblem in the branch-and-bound algorithm.

4.4.2 Branch-and-Bound Search Strategy

The depth-first search and the best-first search strategies are two well-known, and somewhat extreme, strategies in branch-and-bound methods. Implementing a best-first search is almost impossible due to the excessive amount of storage and bookkeeping requirements. On the other hand, the depth-first search is easy to implement, but it has the potential disadvantage of being unable to extricate itself from a poor portion of the tree, due to some inadvertent initial branching decisions. Sherali and Myers (1985/6) suggest a hybrid strategy, where at most a fixed *(MAXACT)* number of end nodes are kept active in the branch-and-bound tree. As a result of computational tests on integer programming problems, the recommended value for *MAXACT* is 5, but this should be kept as large as computationally feasible in practice, depending on the size of the problem. Hence, we used the largest value of *MAXACT* as permissible by storage limitations, depending on the size of the problem. Notice that the depth-first search corresponds to $MAXACT = 1$, and the best-first search corresponds to $MAXACT = \infty$. Therefore, our suggestion is to approach the best-first strategy performance as closely as possible, without violating the storage limitations.

In this approach, branching is performed by splitting the bound interval of a variable as stated in Section 4.6. Each node of the branch-and-bound tree is defined by associating with it a set of bounds on the *x*-variables. For the nodes at the same level of the branch-and-bound tree, these bound intervals are disjoint, except possibly at the boundaries. After solving the bounding problem, each (unfathomed) node has associated with it a lower bound on the optimum of the node subproblem. In order to define its two descendent nodes, a branching variable and a corresponding pair of subintervals for the branching variable are also associated with the current node. Notice that, except for the branching variable, the bounds on the other *x*-variables remain the same for the descendent nodes. Each child node resides within the parent node, and called as a *nonactive node*, until it is *activated* by solving its associated bounding problem. This subsequently defines nonactive child nodes for this node itself, in case it is not fathomed. Notice that the active end nodes have two nonactive child nodes, and according to the search strategy only *MAXACT* of them are allowed to exist at a given time. Each descendent node resides within the parent node as a *nonactive node*, until it is *activated* by solving its associated bounding problem. This subsequently defines nonactive descendents for this node itself, in case it is not fathomed. Notice that the active end nodes have two nonactive descendent nodes, and according to the search strategy, only *MAXACT* active end nodes are allowed to exist at a given time.

4.4.3 Optimality Criterion

To avoid undue excessive computations involved in sifting through alternative optimal solutions or close to global optimal solutions, we adopt the fathoming criterion

$$LB \geq UB - \varepsilon|UB| \tag{4.15}$$

where $0 < \varepsilon < 1$, and where LB is a valid lower bound at the current branch-and-bound node, and UB is the current best (incumbent) solution value for QP. Hence, when the algorithm stops, we can claim that the global minimum is within $100\varepsilon\%$ of the current best solution.

4.5 Range Reduction Strategies

The tightness of the lower and upper bounds that define the *box constraints* on the variables play a major role in the performance of the Lagrangian dual bounding scheme. Fast and effective procedures to improve simple bounds, known as "logical tests," have acquired a good reputation in (mixed) 0-1 integer programming. In addition to showing that these types of tests have their counterparts in global optimization, Hansen, Jaumard and Lu (1991) have also developed new tests using analytical and numerical methods. In the same spirit, we also propose some suitable strategies for improving lower and upper bounds on the variables at each node of the branch-and-bound tree. The first procedure is based on knapsack problems defined by individual linear functional constraints along with the box constraints. The second procedure is motivated by the number of constraints that have to be binding at optimality. The third and the fourth procedures are applied to cutting planes based on, respectively, the Lagrangian dual objective function, and the eigen-transformed separable objective function, both of which are constructed via an upper bound on QP. The latter three strategies are based on some appropriate optimality conditions, by which we can further tighten the box constraints beyond feasibility considerations, by discarding regions that cannot contain an optimal solution. Since the bounds are tightened based on optimality related considerations, they are possibly not implied by the already existing constraints, and in the light of Proposition 2, this helps to generate tighter RLT bounding problems as well.

4.5.1 Range Reduction Strategy 1

This is a strategy for tightening bounds on variables by the virtue of simple feasibility check. By minimizing the left-hand side expressions of the less than or equal to type functional constraints of QP in turn over the box constraints, we can obtain a maximum slack for each constraint. These values are a measure of feasibility for each constraint with respect to the box constraints. This procedure then discards that portion of the interval for each variable for which the maximum slack of some constraint would become negative. This so called *range-restriction* can therefore tighten the variable bounds. Below, we present the implementation details of this procedure using the original bounds (l, u) on x-variables for the clarity of the notation. In the branch-and-bound algorithm, this procedure is applied at each node using the associated bounds on x-variables.

Recalling that $Ax \leq b$ denotes the functional constraints in QP, we first identify the positive and negative entries of the matrix A separately for each constraint and for each variable.

$$J_i^+ = \{j \in \{1,\dots,n\} : a_{ij} > 0\}, \qquad J_i^- = \{j \in \{1,\dots,n\} : a_{ij} < 0\}, \qquad \forall i = 1,\dots,m \tag{4.16}$$

$$I_j^+ = \{i \in \{1,\dots,m\} : a_{ij} > 0\}, \qquad I_j^- = \{i \in \{1,\dots,m\} : a_{ij} < 0\}, \qquad \forall j = 1,\dots,n \tag{4.17}$$

We define a maximum slack, S_i, for each constraint $i \in \{1,\dots, m\}$, as follows.

$$\begin{aligned} S_i &= b_i - \min\left[\sum_{j=1}^{n} a_{ij}x_j : l_j \le x_j \le u_j,\ j = 1,\dots, n\right] \\ &= b_i - \sum_{j \in J_i^+} a_{ij} l_j - \sum_{j \in J_i^-} a_{ij} u_j \end{aligned} \tag{4.18}$$

Then, we modify the bounds for each variable x_j, $j \in \{1,\dots, n\}$.

$$\begin{aligned} u_j^{new} &= \min\{u_j,\ (S_i/a_{ij} + l_j),\ \forall i \in I_j^+\} \\ l_j^{new} &= \max\{l_j,\ (S_i/a_{ij} + u_j),\ \forall i \in I_j^-\} \end{aligned} \tag{4.19}$$

Modified bounds are subsequently used in (4.18) to update the maximum slacks S_i, $i = 1,\dots, m$. We use (4.18) and (4.19) in a cyclic fashion until there is no modification in a complete cycle. During this procedure, if an updated maximum slack value becomes negative, then we fathom the current branch-and-bound node.

4.5.2 Range Reduction Strategy 2

Denoting the number of nonpositive eigenvalues of the matrix Q by q, this procedure is based on the result that at optimality, at least q out of the $m + 2n$ constraints can be required to hold as equalities (see Phillips and Rosen, 1990, and Mueller, 1970). For example, for concave programming, where $q = n$, the global optimum occurs at an extreme point, at which there are at least n (linearly independent) binding hyperplanes, by definition. The proposed strategy computes a measure of redundancy for each functional constraint with respect to the box constraints by maximizing the left-hand side of the less than or equal to type functional constraints. If the number of nonredundant constraints thus detected, plus the number of x-variables that have an *original* lower or upper bound restricting its interval at the current node, is less than q, then we can fathom the current node. For any given variable, we can readily identify in closed-form the range of its interval for which the foregoing criterion would hold, if at all. This range is then discarded unless the remaining interval becomes disjoint, and the procedure continues with another variable in a cyclic fashion until no further restrictions are effected in a complete cycle.

Let (l', u') denote the bounds on the variables associated with the current node of the branch-and-bound tree, and let (l, u) denote the original bound restrictions of the problem QP as before. We determine a minimum slack value for each functional constraint as follows.

$$\begin{aligned} s_i^{\min} &= b_i - \max\left[\sum_{j=1}^{n} a_{ij}x_j \colon l'_j \le x_j \le u'_j,\ j = 1,\dots,n\right] \\ &= b_i - \sum_{j=1}^{n} \max\{a_{ij}l'_j,\ a_{ij}u'_j\}, \qquad i = 1,\dots,m \end{aligned} \tag{4.20}$$

Let $D \subseteq \{1,\dots,m\}$ denote the index set for the nonredundant constraints, i.e., $D \equiv \{i \colon s_i^{\min} \le 0,\ i = 1,\dots,m\}$. In addition to the functional constraints identified by D, the bounds from (l', u') that coincide with their counterparts in the original bound restrictions (l, u) should also be taken into account as potentially binding constraints at optimality. The resulting total number of nonredundant constraints establishes the following fathoming criterion.

$$\text{If } |D| + \sum_{j=1}^{n} \begin{bmatrix} 1, & \text{if } u'_j = u_j \text{ or } l'_j = l_j \\ 0, & \text{otherwise} \end{bmatrix} \leq q - 1 \text{ then we fathom the current node} \tag{4.21}$$

Now, suppose that the condition (4.21) does not hold. Then we can attempt to improve the bounds on the variables by using the same concept over the feasible range of each variable in turn. For this purpose, let us define the index set of nonredundant functional constraints at $x_k = \bar{x}_k$, for some given $k \in \{1, \ldots, n\}$, as follows,

$$D_k(\bar{x}_k) \equiv \{i \in D : s_i^{\min} + \max\{a_{ik} l'_k, a_{ik} u'_k\} - a_{ik}\bar{x}_k \leq 0\} \tag{4.22}$$

and the total number of nonredundant constraints is given by

$$T_k(\bar{x}_k) \equiv |D_k(\bar{x}_k)| + \sum_{\substack{j=1 \\ j \neq k}}^{n} \begin{bmatrix} 1, & \text{if } u'_j = u_j \text{ or } l'_j = l_j \\ 0, & \text{otherwise} \end{bmatrix} + \begin{bmatrix} 1, & \text{if } \bar{x}_k = u_k \text{ or } \bar{x}_k = l_k \\ 0, & \text{otherwise} \end{bmatrix} \tag{4.23}$$

Assuming $l'_k < u'_k$, to improve the lower bound on x_k, our objective is to solve the following problem:

$$\bar{x}_k = \min\{x_k : T_k(x_k) \geq q, \;\; l'_k \leq x_k \leq u'_k\} \tag{4.24}$$

and update l'_k as $l_k^{new} = \bar{x}_k$, if (4.24) has a feasible solution. Otherwise, we fathom the current node.

Notice that, in order to actually improve l'_k, the total number of nonredundant constraints at $x_k = l'_k$ has to be less than q. Otherwise, we move to another variable or try to improve the upper bound u'_k. Now, suppose that $T_k(l'_k) \leq q - 1$, then we increase the lower bound on x_k from l'_k towards u'_k in order to solve problem (4.24). During this process, we are interested in the points where the total number of nonredundant constraints, say T, is updated. To initialize T, we set $T = T_k(l'_k)$, unless $l'_k = l_k$, in which case, we set $T = T_k(l'_k) - 1$, since the constraint $x_k \geq l_k$ becomes redundant as soon as we increase the lower bound. The points at which T changes can be computed explicitly as follows.

Case 1: Some constraint $a_i x \leq b_i$, $i \in D$, that is redundant at $x_k = l'_k$ becomes nonredundant at some point $x_k^{(1)} \leq u'_k$. Such a constraint can be easily identified, since $s_i^{\min} + a_{ik}u'_k - a_{ik}l'_k > 0$, indicating that the i^{th} constraint is redundant at $x_k = l'_k$, and $s_i^{\min} \leq 0$, since $i \in D$. Notice that $a_{ik} > 0$, and consequently, $a_{ik}u'_k = \max\{a_{ik}u'_k, a_{ik}l'_k\}$. Once such a constraint is identified, the point where Case 1 occurs is computed as follows:

$$x_k^{(1)} = \frac{s_i^{\min}}{a_{ik}} + u'_k \tag{4.25}$$

Case 2: Some constraint $a_i x \leq b_i$, $i \in D$, that is redundant at $x_k = u'_k$ becomes nonredundant at some point $x_k^{(2)} > l'_k$. Such a constraint can be easily identified, since $s_i^{\min} + a_{ik}l'_k - a_{ik}u'_k > 0$, indicating that the i^{th} constraint is redundant at $x_k = u'_k$, and $s_i^{\min} \leq 0$, since $i \in D$. Notice that $a_{ik} < 0$, and consequently, $a_{ik}l'_k = \max\{a_{ik}u'_k, a_{ik}l'_k\}$. Once such a constraint is identified, the point where Case 2 occurs is computed as follows:

$$x_k^{(2)} = \frac{s_i^{\min}}{a_{ik}} + l'_k \tag{4.26}$$

After ordering all the points where Case 1 and Case 2 occurs in increasing order, we restrict our search on $(l'_k, u'_k]$ to solve (4.24) only on these points and on one more additional point of $x_k = u'_k$. While moving from l'_k to u'_k, if a Case 1 point is encountered, then we increase T by 1. If $T \geq q$ then the current point $x_k^{(1)}$ solves the problem (4.24), so after setting $l_k^{new} = x_k^{(1)}$, we quit the search. On the other hand, when a Case 2 point is encountered, we decrease T by 1, and continue with the search. If the search reaches the point $x_k = u'_k$ and $T_k(u'_k) \geq q$, then $\bar{x}_k = u'_k$, and so, we set $l_k^{new} = u'_k$. Otherwise, $T_k(u'_k) < q$, and so, we fathom the current branch-and-bound node, since there cannot be at least q constraint binding over the interval $[l'_k, u'_k]$.

After updating the lower bound, we try to improve the upper bound on x_k by a similar procedure, and then, the whole procedure continues with another variable in a cyclic fashion until there is no improvement in the bounds in a complete cycle.

4.5.3 Range Reduction Strategy 3

For each node, given the current bound restrictions on the x-variables, consider the Lagrangian dual subproblem (3.28) corresponding to the incumbent dual solution of the parent subproblem. Since this gives a valid lower bounding problem, we examine each variable in turn and identify a subinterval of its range for which, if the variable was so restricted, the resulting Lagrangian based bound would fathom the node. Any such range identified for a variable is discarded, unless the remaining interval becomes disjoint. In a similar fashion, after having solved the current node's problem via the Lagrangian dual scheme, we perform this range reduction before making any branching decision. This restriction also serves as an input for the immediate descendent nodes, for which this test will be applied after imposing a branching decision.

Let (l', u') be the simple bounds on x-variables associated with the current node. Then, consider the following Lagrangian subproblem where the objective function is constructed using the incumbent dual solution corresponding to *the parent node* of the current node.

$$\bar{c}_0 + \text{Minimize} \quad \sum_{k=1}^{n} \bar{c}_k x_k + \sum_{k=1}^{n} \bar{q}_{kk} w_{kk} + \sum_{k=1}^{n-1} \sum_{l=k+1}^{n} (\bar{q}_{kl} w_{kl} + \bar{\gamma}_{kl} w'_{kl}) \tag{4.27.1}$$

$$\text{subject to} \quad l'_l x_k + l'_k x_l - l'_k l'_l \le w_{kl} \le l'_l x_k + u'_k x_l - u'_k l'_l \quad 1 \le k < l \le n \tag{4.27.2}$$

$$u'_l x_k + u'_k x_l - u'_k u'_l \le w'_{kl} \le u'_l x_k + l'_k x_l - l'_k u'_l \quad 1 \le k < l \le n \tag{4.27.3}$$

$$w_{kk} = x_k^2, \qquad l'_k \le x_k \le u'_k \qquad k = 1, \ldots, n \tag{4.27.4}$$

Notice that the problem (4.27) gives a valid lower bound on the current node's subproblem. Proceeding as described in Sections 3.6 and 4.1 to eliminate w_{kl} and w'_{kl} variables from (4.27), consider the resulting problem at the stage when it becomes separable.

$$\hat{c}_0 + \text{Minimize} \quad \sum_{k=1}^{n} \hat{c}_k x_k + \sum_{k=1}^{n} \hat{q}_{kk} w_{kk}$$

$$\text{subject to} \quad w_{kk} = x_k^2 \qquad k = 1, \ldots, n$$

$$l'_k \le x_k \le u'_k \qquad k = 1, \ldots, n$$

For convenience, we have used the nonconvex constraint (3.27) above instead of its convex equivalent (3.23). Suppose that the current node is not fathomed, that is

$$\hat{c}_0 + \sum_{k=1}^{n} \min\{\hat{c}_k x_k + \hat{q}_{kk} x_k^2 : l'_k \leq x_k \leq u'_k\} < \text{UB} - \varepsilon|\text{UB}|. \tag{4.28}$$

By setting all but one x-variables to their minimum value, we obtain a quadratic inequality in one variable, for any $k \in \{1, \ldots, n\}$,

$$\begin{aligned} &\hat{c}_k x_k + \hat{q}_{kk} x_k^2 + \sum_{\substack{l=1 \\ l \neq k}}^{n} \min\{\hat{c}_l x_l + \hat{q}_{ll} x_l^2 : l'_l \leq x_l \leq u'_l\} + \hat{c}_0 - (\text{UB} - \varepsilon|\text{UB}|) \\ &\equiv \hat{c}_k x_k + \hat{q}_{kk} x_k^2 + C_k \leq 0. \end{aligned} \tag{4.29}$$

To modify bounds on x-variables by using (4.29), consider three different cases.

Case 1: $\hat{q}_{kk} = 0$.

- If $\hat{c}_k = 0$ then no update of bounds.
- If $\hat{c}_k > 0$ then $u_k^{new} = \min\{u'_k, -C_k/\hat{c}_k\}$.
- If $\hat{c}_k < 0$ then $l_k^{new} = \max\{l'_k, -C_k/\hat{c}_k\}$.

Case 2: $\hat{q}_{kk} > 0$. Let r_1 and r_2 $(r_1 \leq r_2)$ be the roots of the quadratic function in (4.29). Hence, (4.29) becomes $(x_k - r_1)(x_k - r_2) \leq 0$. Notice that r_1 and r_2 are real, because no fathoming occured in (4.28).

- Update bounds as, $l_k^{new} = \max\{l'_k, r_1\}$, $u_k^{new} = \min\{u'_k, r_2\}$.

Case 3: $\hat{q}_{kk} < 0$. Let r_1 and r_2 be the roots of the quadratic function in (4.29), so that (4.29) can be written as $(x_k - r_1)(x_k - r_2) \geq 0$.

- If r_1 and r_2 are not real, then (4.29) holds for all $x_k \in R$. Therefore, there is no update on bounds.
- If r_1 and r_2 are real, say $r_1 \leq r_2$, then we are interested in the interval $I_o = [l'_k, u'_k] - (r_1, r_2)$.
- Accordingly, if I_o is connected then modify the bounds as follows.
    - If $I_o = [l'_k, u'_k]$ then the bounds are left unaltered.
    - Otherwise, if $r_1 < l'_k \leq r_2$ then $[l_k^{new}, u_k^{new}] = [r_2, u'_k]$.
    - Otherwise, if $r_2 > u'_k \geq r_1$ then $[l_k^{new}, u_k^{new}] = [l'_k, r_1]$.
- On the other hand, if I_o is disconnected, then both r_1 and r_2 lie within the bounds, and this yields two disjunctive intervals of interest, namely, $[l'_k, r_1]$ and $[r_2, u'_k]$. Unfortunately, this is not helpful in our range reduction strategy framework, so we leave the bounds unaltered.

Notice that in above procedure, we did not check if the updated bounds are consistent or if $I_o = \emptyset$, since (4.28) takes care of these cases. To implement Range Reduction Strategy 3, we can do a single pass, or continue in a cyclic fashion until there is no improvement in the bounds for any of the x-variables.

Due to memory space restrictions, we may not be able to store the objective function coefficients in (4.27.1) associated with all the existing nodes of the branch-and-bound tree to be used later in Range Reduction Strategy 3, when their descendent nodes are activated. In such cases, we can construct (4.27.1) using the optimal dual solution of the root node. This maintains the validity of (4.27) as a lower bounding problem for all the nodes of the branch-and-bound tree, although at the expense of probably weakening Range Reduction Strategy 3.

Recall that each node subproblem is re-scaled using the scaling scheme of Section 4.4.1 before solving its associated Lagrangian dual problem. This creates a compatibility problem for the objective function coefficients in (4.27) passed on to the current node from its parent node whose associated optimal dual values reflect the effects of scaling. To overcome this problem, we unscale the "scaled" counterparts of the coefficients in (4.27.1) taking the root node (original) problem as the "unscaled" base case. Let $(\bar{l}, \bar{u})$ be the bounds associated with the parent node, and let (c^s, q^s, γ^s), be the scaled version of the objective function coefficient in (4.27.1) as they appear in the parent node. Following the definition of the scaled variables, denoted by the superscript "s" below,

$$x_k^s = \frac{x_k - \bar{l}_k}{\bar{u}_k - \bar{l}_k}, \quad w_{kk}^s = \frac{w_{kk} - 2\bar{l}_k x_k + \bar{l}_k^{\,2}}{(\bar{u}_k - \bar{l}_k)^2} \qquad k = 1, \ldots, n$$

$$w_{kl}^s = \frac{w_{kl} - \bar{l}_l x_k - \bar{l}_k x_l + \bar{l}_l \bar{l}_k}{(\bar{u}_k - \bar{l}_k)(\bar{u}_l - \bar{l}_l)}, \quad w'^s_{kl} = \frac{w'_{kl} - \bar{l}_l x_k - \bar{l}_k x_l + \bar{l}_l \bar{l}_k}{(\bar{u}_k - \bar{l}_k)(\bar{u}_l - \bar{l}_l)} \qquad 1 \le k < l \le n \tag{4.30}$$

we obtain the unscaled coefficients of (4.27.1) as follows.

$$\bar{c}_0 = c_0^s - \sum_{k=1}^{n} \frac{c_k^s \bar{l}_k}{(\bar{u}_k - \bar{l}_k)} + \sum_{k=1}^{n} \frac{q_{kk}^s \bar{l}_k^{\,2}}{(\bar{u}_k - \bar{l}_k)^2} + \sum_{k=1}^{n-1} \sum_{l=k+1}^{n} \frac{\bar{l}_k \bar{l}_l (q_{kl}^s + \gamma_{kl}^s)}{(\bar{u}_k - \bar{l}_k)(\bar{u}_l - \bar{l}_l)}$$

$$\bar{c}_k = \frac{c_k^s}{\bar{u}_k - \bar{l}_k} - \frac{2q_{kk}^s \bar{l}_k}{(\bar{u}_k - \bar{l}_k)^2} - \sum_{l=1}^{k-1} \frac{(q_{lk}^s + \gamma_{lk}^s)(\bar{l}_l x_k + \bar{l}_k x_l)}{(\bar{u}_k - \bar{l}_k)(\bar{u}_l - \bar{l}_l)} - \sum_{l=k+1}^{n} \frac{(q_{kl}^s + \gamma_{kl}^s)(\bar{l}_l x_k + \bar{l}_k x_l)}{(\bar{u}_k - \bar{l}_k)(\bar{u}_l - \bar{l}_l)},$$

$$\bar{q}_{kk} = \frac{q_{kk}^s}{(\bar{u}_k - \bar{l}_k)^2} \qquad \forall\ k = 1, \ldots, n$$

$$\bar{q}_{kl} = \frac{q_{kl}^s}{(\bar{u}_k - \bar{l}_k)(\bar{u}_l - \bar{l}_l)}, \quad \bar{\gamma}_{kl} = \frac{\gamma_{kl}^s}{(\bar{u}_k - \bar{l}_k)(\bar{u}_l - \bar{l}_l)} \qquad \forall 1 \le k < l \le n$$

Now, the problem (4.27) is compatible with the base problem, and we can proceed with Range Reduction Strategy 3.

REMARK 4: We can also employ this procedure *after* solving the current node's bounding problem before making branching decisions. In this case, the objective function is constructed using the incumbent dual solution of the current node. Consequently, the improved bounds are valid only for the successive nodes of the current node or, more formally, for the subtree rooted at the current node. The resulting improved bounds serve as an input for the immediate descendant nodes. Moreover, improved bounds are also expected to be useful in making advantageous branching and partitioning decisions.

4.5.4 Range Reduction Strategy 4

This strategy has the same motivation as that of Range Reduction Strategy 3 in that it is based on restricting the solution space to points for which the objective value is at least as good as the incumbent value UB. To obtain a favorable structure in $cx + x^tQx \leq \text{UB}$, this test is performed on the eigen-transformed problem EQP of Section 3.4. We first derive lower and upper bounds on the z-variables, say $[L, U]$, by minimizing and then maximizing for each $i = 1, \ldots, n$, the definition function $z_i = (P^t_{i\bullet})x$, where $(P^t_{i\bullet})$, is the i^{th} row of P^t, over the box constraints on the x-variables corresponding to the current node restrictions. Let z^0 minimize the separable eigen-transformed objective function over $L \leq z \leq U$. Then, considering one z-variable at a time, while fixing the others at their values in z^0, we determine a subinterval to be eliminated for that z-variable by finding the range for which the objective function value would exceed the incumbent solution value. We eliminate such a subinterval, unless the remaining interval becomes disjoint, and continue this procedure in a cyclic fashion until there is no further reduction in the bounding intervals for the z-variables. Let $[L^{new}, U^{new}]$ be the final bounds thus obtained. Then, by including the constraints $L^{new}_i \leq z_i = (P^t_{i\bullet})x \leq U^{new}_i$ for those z_i, $i = 1, \ldots, n$, for which an interval reduction has resulted, we perform Range Reduction Strategy 1 again to possibly further restrict the x-variable bounds. In the rest of this subsection, we present the details of this procedure.

Using the separable objective function (3.18.1) of EQP, we obtain the following restriction on the eigen-transformed variables, where $g \equiv cP$.

$$cPz + z^tDz = \sum_{i=1}^{n} g_iz_i + \lambda_iz_i^2 \leq \text{UB} \qquad (4.31)$$

To test the satisfiability of (4.31) by minimizing each quadratic univariate function separately in (4.31), we first derive simple bounds on the z-variables by using the bounds on the x-variables. From (3.17), for any $i \in \{1, \ldots, n\}$, we have,

$$z_i = \sum_{j=1}^{n} p_{ji} x_j = \sum_{j \in J_i^+} p_{ji} x_j + \sum_{j \in J_i^-} p_{ji} x_j \tag{4.32}$$

where $J_i^+ = \{j : p_{ji} > 0\}$ and $J_i^- = \{j : p_{ji} \leq 0\}$. Then,

$$\sum_{j \in J_i^+} p_{ji} l_j + \sum_{j \in J_i^-} p_{ji} u_j = L_i \leq z_i \leq \sum_{j \in J_i^+} p_{ji} u_j + \sum_{j \in J_i^-} p_{ji} l_j = U_i \tag{4.33}$$

Let z_i^o solve the problem to minimize$\{g_i z_i + \lambda_i z_i^2 : L_i \leq z_i \leq U_i\}$, $\forall i = 1, \ldots, n$. Then from (4.31), $\sum_{i=1}^{n} (g_i z_i^o + \lambda_i (z_i^o)^2)$ is a lower bound on the optimal value of the current node's subproblem. If we know that the current node problem cannot yield a (significantly) better solution than the incumbent one, then we can fathom this node, i.e.,

$$\text{if} \quad \sum_{i=1}^{n} (g_i z_i^o + \lambda_i (z_i^o)^2) \geq \text{UB} - \varepsilon |\text{UB}| \quad \text{then we fathom the current node.} \tag{4.34}$$

Suppose the condition (4.34), does not hold, and so, the current node is still active. By setting all but one z-variables to their above minimizing value z^o in (4.31), we obtain a quadratic inequality in one variable, which must hold if the node is to remain active. This inequality is given by

$$g_k z_k + \lambda_k z_k^2 + \sum_{\substack{i=1 \\ i \neq k}}^{n} (g_i z_i^o + \lambda_i (z_i^o)^2) - \text{UB} - \varepsilon|\text{UB}| \equiv g_k z_k + \lambda_k z_k^2 + C_k \leq 0 \;\; \forall k = 1,\dots,n. \quad (4.35)$$

To modify bounds on z-variables by using (4.35), consider three different cases.

Case 1: $\lambda_k = 0$.

- If $g_k = 0$ then no update of bounds.
- If $g_k > 0$ then $U_k^{new} = \min\{U_k, -C_k/g_k\}$.
- If $g_k < 0$ then $L_k^{new} = \max\{L_k, -C_k/g_k\}$.

Case 2: $\lambda_k > 0$. Let r_1 and r_2 $(r_1 \leq r_2)$ be the roots of the quadratic function in (4.35). Hence, (4.35) becomes $(z_k - r_1)(z_k - r_2) \leq 0$. Notice that r_1 and r_2 are real, because no fathoming occured in (4.34).

- Update bounds as, $L_k^{new} = \max\{L_k, r_1\}$, $U_k^{new} = \min\{U_k, r_2\}$.

Case 3: $\lambda_k < 0$. Let r_1 and r_2 be the roots of the quadratic function in (4.35), so that (4.35) can be written as $(z_k - r_1)(z_k - r_2) \geq 0$.

- If r_1 and r_2 are not real, then (4.35) holds for all $z_k \in R$. Therefore, there is no update on bounds.
- If r_1 and r_2 are real, say, $r_1 \leq r_2$, then we are interested in the interval $I_o = [L_k, U_k] - (r_1, r_2)$.
- Accordingly, if I_o is connected then modify the bounds as follows.
  - If $I_o = [L_k, U_k]$ then the bounds are left unaltered.

  - Otherwise, if $r_1 < L_k \leq r_2$ then $[L_k^{new}, U_k^{new}] = [r_2, U_k]$.
  - And if $r_2 > U_k \geq r_1$ then $[L_k^{new}, U_k^{new}] = [L_k, r_1]$.

- On the other hand, if I_o, is disconnected, then both r_1 and r_2 lie within the bounds, and this yields two disjunctive intervals of interest, namely, $[L_k, r_1]$ and $[r_2, U_k]$. Unfortunately, this is not helpful in our range reduction strategy framework, so we leave the bounds unaltered.

Notice that in above procedure we did not check if the updated bounds are consistent or if $I_o = \emptyset$, since (4.34) takes care of these cases. To implement Range Reduction Strategy 4, we can perform a single pass or continue in a cyclic fashion until there is no improvement in the bounds for the z-variables. Then, for each z_i, $i \in \{1, ..., n\}$, we include the following constraints in Range Reduction Strategy 1 only if the corresponding bound is updated.

$$L_i^{new} \leq \sum_{j=1}^{n} p_{ji} x_j \leq U_i^{new} \tag{4.36}$$

Using the augmented set of linear constraints, we perform Range Reduction Strategy 1 until there is no further improvement in the bound restrictions on x-variables.

Notice that we can apply Range Reduction Strategy 4 at each node of the branch-and-bound tree by replacing (l, u) with (l', u') throughout this subsection, where (l', u') denote the bounds on x-variables associated with the current node.

4.6 Branching Variable Selection

We prescribe below a branching rule that attempts to resolve the discrepancy between the values of the RLT variables and the corresponding nonlinear terms they represent. If there is no such discrepancy, that is, if (3.3) holds at an optimal solution to the bounding problem, then this solution is also optimal to QP. Otherwise, given an optimal solution $(\bar{x}, \bar{w})$ to the RLT bounding problem, to identify the RLT variables that contribute toward reducing the lower bound below the true QP objective value due to the discrepancies between their corresponding nonlinear terms, let us compute the quantity

$$d_1 = \underset{1 \leq k \leq l \leq n}{\text{minimum}} \left\{ \min\{0, \begin{bmatrix} q_{kl}, & \text{if } k = l \\ 2q_{kl}, & \text{if } k < l \end{bmatrix} (\bar{w}_{kl} - \bar{x}_k \bar{x}_l)\} \right\} \tag{4.37}$$

Notice that, by the inner minimum operation in (4.37), we have avoided the terms that yield discrepancies which work to our benefit by increasing the objective function value of the bounding problem beyond the actual corresponding quadratic value. In particular, $d_1 \leq 0$, and if $d_1 = 0$, then $\bar{x}$ solves the current quadratic branch-and-bound node subproblem. The next proposition proves this assertion, and therefore gives a sufficient condition under which the x-variable part of the solution to the RLT bounding problem optimally solves QP.

PROPOSITION 6: Suppose that $(\bar{x}, \bar{w})$ solves the RLT bounding problem of QP, with the corresponding objective value being $\bar{z}$. Let d_1 be computed as in (4.37). If $d_1 = 0$ then $\bar{x}$ solves QP with the corresponding objective value being $\bar{z}$.

Proof: By Proposition 2, the constraints of QP are implied in the RLT problem. (Alternatively, if a reduced RLT is being employed, then these constraints are directly present in the RLT problem, unless they are implied as described at the end of Section 3.7.) Therefore, $\bar{x}$ is feasible to QP, and its objective function value $c\bar{x} + \bar{x}^tQ\bar{x}$ gives an upper bound on the actual optimum $v[QP]$. Due to the inner minimization operation in (4.37), $d_1 = 0$ implies that $q_{kk}\overline{w}_{kk} \geq q_{kk}\bar{x}_k^2$ $\forall k = 1,\ldots, n$, and $q_{kl}\overline{w}_{kl} \geq q_{kl}\bar{x}_k\bar{x}_l$ $\forall 1 \leq k < l \leq n$. Hence, $\bar{z} \geq c\bar{x} + \bar{x}^tQ\bar{x}$. But since $\bar{x}$ is feasible to QP, and $\bar{z}$ is a lower bound on QP, we have, $c\bar{x} + \bar{x}^tQ\bar{x} \geq v[QP] \geq \bar{z} \geq c\bar{x} + \bar{x}^tQ\bar{x}$, which implies that equality holds throughout. Hence, $\bar{x}$, of objective value $\bar{z}$, solves QP, and this completes the proof. ■

Now, suppose that d_1 given by some indices $(\underline{k}, \underline{l})$ in (4.37) is negative. (If there are ties in (4.37), we break it in favor of the maximum discrepancy $\bar{x}_k\bar{x}_l - \overline{w}_{kl}$, with further ties broken arbitrarily.) In order to choose between $x_{\underline{k}}$ and $x_{\underline{l}}$ for a branching variable, using the same motivation as in (4.37), we compute for $t = \underline{k}$ and $\underline{l}$,

$$d_2(t) = \sum_{j=1}^{t-1} \min\{0, 2q_{jt}(\overline{w}_{jt} - \bar{x}_j\bar{x}_t)\} + \sum_{j=t+1}^{n} \min\{0, 2q_{tj}(\overline{w}_{tj} - \bar{x}_t\bar{x}_j)\} + \min\{0, q_{tt}(\overline{w}_{tt} - \bar{x}_t^2)\} \tag{4.38}$$

The branching variable index is then selected as follows:

$$br = \operatorname{argmin}\{d_2(t) : t = \underline{k}, \underline{l}\} \tag{4.39}$$

The foregoing branching scheme is based on an optimal solution $(\bar{x}, \overline{w})$ to the RLT bounding problem. However, note that since we are solving the Lagrangian dual of the enhanced first-level RLT problem, we do not directly obtain the optimal primal (x, w) variable values. For this purpose, we attempted to use a penalty function based primal recovery scheme (see Appendix C). However, the additional computational effort required to solve this problem turns out to be comparable to the computational effort required to solve the Lagrangian dual problem itself, which ruled out the use of penalty function based scheme in the overall algorithm. Based on Theorem 1 of the next chapter, we could solve a linear program obtained by surrogating all the constraints of RLT-LP using the optimal dual solution obtained via LD-RLT-NLP, except for the bound factor product constraints and the original constraints of QP, and guarantee convergence of the algorithm. However, this linear program itself might be a computational bottleneck. Hence, for theoretical purposes, one could resort to solving the latter linear program only finitely often along any branch of the branch-and-bound tree, but for the most part, resort to the following branching scheme that is motivated by the above theory.

Branching Scheme:

Step 1: Consider the Lagrangian subproblem associated with the dual incumbent solution to LD-RLT-NLP at the stage (3.28), where (l, u) are the x-variable bounds associated with the *current node*. Letting $f_k(x_k) = \hat{c}_k x_k + \hat{q}_{kk} x_k^2$, select a branching variable index as follows:

$$br = \underset{\substack{k \in \{1,\dots,n\} \\ \ni l_k < \bar{x}_k < u_k}}{\operatorname{argmax}} \left[\min\{|f_k(u_k) - f_k(\bar{x}_k)|, |f_k(l_k) - f_k(\bar{x}_k)|\} \right] \tag{4.40}$$

where $\bar{x}$ solves (3.28), breaking ties in favor of the variable that has the largest feasible interval at the current node. If all variables are at their bounds in $\bar{x}$, proceed to Step 2.

Step 2: Using the incumbent dual solution to LD-RLT-NLP as the starting solution, continue the conjugate subgradient procedure for 50 more iterations omitting the resetting strategy, and accumulate $(\bar{x}, \bar{w})$ as the average of the Lagrangian subproblem solutions. (In theory, Larsson and Liu (1989) show that this should ultimately converge to the optimal primal solution to RLT-NLP under a very restricted step-size strategy. More general results on recovering primal feasible and optimal solutions are presented in Sherali and Choi (1994). To aid this process, we also attempt to project $\bar{x}$, if infeasible, onto the feasible region of QP by taking a single step toward this region along a direction defined by the violated constraints.) Select a branching variable using (4.37)-(4.39) provided that $d_1 < 0$ in (4.37), and otherwise, if $d_1 = 0$, then proceed to Step 3.

Step 3: Select the x-variable that has the largest feasible interval in the current node problem as the branching variable. That is, let

$$br = \underset{k \in \{1, \ldots, n\}}{\operatorname{argmax}} \{(u_k - l_k)\} \tag{4.41}$$

and exit this procedure.

Partitioning Phase: If the branching variable x_{br} is selected using Step 1 or Step 2 of the above procedure, then split its current interval at the value $\bar{x}_{br}$, creating the partitions $[l_{br}, \bar{x}_{br}]$ and $[\bar{x}_{br}, u_{br}]$, provided that the length of each resulting partition is at least 5% of the length of the current interval $[l_{br}, u_{br}]$. Otherwise, partition the current interval of x_{br} by simply bisecting it.

4.7 *Finding Good Quality Feasible Solutions*

In the branch-and-bound algorithm, besides a tight lower bound, we should also actively seek good quality solutions early in the algorithm. Although RLT-(N)LP yields a feasible solution by construction, this may not be true for the case of LD-RLT-NLP. Therefore, we developed and tested the following three heuristic procedures, and composed them in the manner described below.

In the first procedure, we formulate an l_1-norm penalty function for the problem QP that incorporates absolute violations in the functional constraints into the penalty term, and then approximately minimize this penalty function using the conjugate subgradient algorithm described in Section 4.2. As the starting solution, we used the average of the x-variable part of the subproblem solutions obtained at improving iterations of the conjugate subgradient algorithm while solving the Lagrangian dual problem at the current branch-and-bound node. During this procedure, we attempt to project promising near feasible points onto the original feasible region of QP by taking a single step toward this region along a direction defined by the violated constraints. We consider a feasibility tolerance of 10^{-6} in Euclidean distance to be compatible with the default setting for MINOS (see Murtagh and Saunders, 1987).

In the second procedure, we simply apply MINOS to the original problem QP, using the resulting point of the foregoing penalty approach as the starting solution.

In the third procedure, at each node, having solved the Lagrangian dual problem, we formulate (3.26) corresponding to the incumbent dual solution, but now we also include the functional constraints of QP in this subproblem. The resulting convex program is then solved using MINOS. Notice that this procedure also happens to be a dual ascent step for the Lagrangian dual problem.

In the overall branch-and-bound algorithm, we implemented the following heuristic scheme. For the first 10 nodes of the branch-and-bound tree, we employed the second procedure, using the solution of the first procedure as a starting solution. At all subsequent nodes, we employed the first and the third procedures, except that whenever the incumbent solution improved, we executed the second procedure using this new incumbent point as the starting solution, to possibly further improve this incumbent solution.

The first and the third heuristic procedures, as well as the projection scheme for near feasible points, are presented in detail in separate subsections below.

4.7.1 Penalty Function Based Method

We adopt an l_1-norm penalty function for the problem QP, where the functional constraints are incorporated in the penalty function. The minimum of this penalty function is then sought over the rectangular region defined by the *original* bounds on the x-variables.

$$\underset{l \le x \le u}{\text{Minimize}} \left\{ cx + x^t Q x + \sum_{i=1}^{m} \mu \max(0, a_i x - b_i) \right\} \tag{4.42}$$

To solve the nondifferentiable problem (4.42), we modify the conjugate subgradient algorithm of Section 4.2 by omitting the resetting step, setting the iteration limit to 300, and consequently, changing the step-length parameters $\alpha = 0.25$, and β_k to

$$\beta_k = \begin{cases} 0.75, & \text{if } k \le 50 \\ 0.50, & \text{if } 51 \le k \le 75 \\ 0.25, & \text{if } 76 \le k \le 100 \\ 1.00, & \text{if } k \ge 101 \end{cases} \tag{4.43}$$

As an approximation to the optimal solution to QP, which is needed in the step-length calculations, we use the lower bound via the bounding problem of node 0 of the branch-and-bound tree. The initial penalty parameter is set to $\mu^{(0)} = 150$. During the first 100 iterations of the conjugate subgradient algorithm, the penalty parameter is increased according to $\mu^{(k+1)} = 1.2\mu^{(k)}$, where the superscripts denote the iteration number. For the iterations $k \ge 101$, the penalty parameter is fixed at $\mu^{(k)} = \mu^{(100)}$. As a starting solution, we use the average of x-variable part of the subproblem solutions obtained at improving iterations of the conjugate subgradient algorithm while solving the Lagrangian dual problem. Note that at each node of the branch-and-bound tree, the only change in the penalty problem run is the starting point used.

Let x^{inc} be the incumbent solution at some iteration of the conjugate subgradient algorithm. As a measure of infeasibility, we use the maximum Euclidean distance from the given point to the violated constraints given by $F(x) = \max_{i=1,\dots,m} \{0, a_i x - b_i\}/\|a_i\|$. During the first 100 iterations, if $F(x^{(k)})$ is small, the incumbent solution is updated using criteria that compromise between an improvement in the QP objective function value and a reduction in the measure of infeasibility. In particular, the incumbent solution is updated if at least one of the following conditions are satisfied.

$$\begin{aligned} &(i) \quad cx^{(k)} < cx^{inc} \text{ and } F(x^{(k)}) < F(x^{inc}) + 10^{-5} \\ &(ii) \quad cx^{(k)} < cx^{inc} + \frac{\varepsilon}{4}|cx^{inc}| \text{ and } F(x^{(k)}) < F(x^{inc}) - 0.5 \times 10^{-5} \end{aligned} \tag{4.44}$$

During the last 200 iterations, irrespective of the magnitude of $F(x^{(k)})$, the incumbent solution is updated whenever there is reduction in the measure of infeasibility. Also, whenever $F(x^{(k)}) < 0.3 \times 10^{-5}$ and $cx^{(k)}$ is less than the incumbent upper bound on QP, we also attempt to project $x^{(k)}$ onto the original feasible region of QP using the projection scheme of the next subsection. If the maximum infeasibility measure reduces by half at the projected point, then we attempt upto 5 more projections.

The above procedure is terminated if a feasible solution that improves the current incumbent upper bound on QP is found at some stage. Otherwise, when the limit on the number of iterations is reached, we attempt to project the final incumbent solution onto the feasible region using the projection scheme of the next subsection.

4.7.2 Projection of Near Feasible Points onto the Feasible Region

In this projection scheme, given a point $\bar{x}$ that is feasible to the imposed bounds but is infeasible to the functional constraints of QP, we take a single step towards the feasible region along a direction based on a weighted average of the negative gradients of the violated constraints. Our implementation of this scheme accommodates its iterative use as it is called for by the penalty function method of the previous subsection. We consider a feasibility tolerance of 10^{-6} to be compatible with the default setting for MINOS. That is, any variable, including the slack variables corresponding to functional constraints, can violate their closest bound by at most 10^{-6} in terms of Euclidean distance. Given the index set I_{infs} corresponding to the violated constraints, the direction of motion $d \in \mathbb{R}^n$ is computed as follows,

$$d = P_{l \le x \le u}\left[-\sum_{i \in I_{infs}} \left(\frac{a_i\bar{x} - b_i}{\|a_i^t\|^2} \right) a_i^t \right] \tag{4.45}$$

where $P_{\{\cdot\}}[\cdot]$ denotes projection on the set $\{\cdot\}$. Projection $P_{\{\cdot\}}[\cdot]$ sets those elements of the pre-projected direction to zero corresponding to the variables that are at either their lower or upper bounds, and these bounds would have been violated for any positive step taken along this direction. We then compute a step-length $\lambda > 0$ to be taken along direction d, aiming to satisfy all the violated constraints, while not violating any bounding restrictions.

$$\lambda = \min\left[\underset{k \in \{1,\dots,n\}}{\text{minimum}} \begin{Bmatrix} (\bar{x}_k - l_k)/ - d_k, & \text{if } d_k < 0 \\ (u_k - \bar{x}_k)/d_k, & \text{if } d_k > 0 \end{Bmatrix}, \ \underset{i \in I_{infs}}{\text{maximum}} \{(b_i - a_i\bar{x})/a_i d\} \right] \tag{4.46}$$

If the resulting point $\tilde{x} = \bar{x} + \lambda d$ is not feasible, then the procedure stops with an indication of *failure*. Otherwise, $\tilde{x}$ is the feasible point recovered by this procedure. This procedure works well in recovering a feasible solution when the infeasibilities are small in magnitude.

4.7.3 Convex Quadratic Programming Based Method

Another way to obtain a feasible solution involves solving a convex quadratic programming problem. At the end of the Lagrangian dual optimization procedure, we solve an additional Lagrangian subproblem corresponding to the incumbent dual solution. However, this time we also include the original functional constraints of QP in this problem.

Consider the reduced Lagrangian subproblem (3.26) corresponding to the incumbent dual solution at the stage when this subproblem becomes separable, which is repeated below.

$$\hat{c}_0 + \text{Minimize} \quad \sum_{k=1}^{n} \hat{c}_k x_k + \sum_{k=1}^{n} \hat{q}_{kk} w_{kk} \tag{4.47.1}$$

$$\text{subject to} \quad x_k^2 \leq w_{kk} \leq (u_k + l_k) - l_k u_k \qquad k = 1, \ldots, n \tag{4.47.2}$$

$$l_k \leq x_k \leq u_k \qquad k = 1, \ldots, n \tag{4.47.3}$$

We have used the convex constraint (3.23) above instead of the nonconvex constraint (3.27) in order to obtain a convex quadratic program upon eliminating the w_{kk}-variables. After eliminating the w_{kk}-variables by using (4.47.2) according to the sign of $\hat{q}_{kk}$-coefficients in (4.47.1), we include the original functional constraints $Ax \leq b$ of problem QP in (4.47) to obtain the following convex quadratic programming problem.

$$\hat{c}_0 + \text{Minimize} \quad \sum_{k=1}^{n} \hat{c}_k x_k + \sum_{k=1}^{n} \hat{q}_{kk} \begin{bmatrix} x_k^2, & \text{if } \hat{q}_{kk} \geq 0 \\ (u_k + l_k) - l_k u_k, & \text{if } \hat{q}_{kk} < 0 \end{bmatrix} \tag{4.48.1}$$

$$\text{subject to} \quad Ax \leq b \tag{4.48.2}$$

$$l_k \leq x_k \leq u_k \qquad k = 1, \ldots, n \tag{4.48.3}$$

We solve (4.48) by using MINOS. Then, we check if the solution of this problem improves the incumbent upper bound. Note that the optimal objective function value of this problem might also give a dual ascent, since we are actually solving a tighter Lagrangian subproblem corresponding to the incumbent dual solution.

4.8 Summary of the Algorithm

Combining the features described in the last two chapters, we present the proposed branch-and-bound algorithm as a step by step procedure.

Step 0. Initialization: Apply the heuristic procedure of Section 4.7 using $x^0 = Pz^0$ as the starting solution, where z^0 is defined as in Section 4.5.4, to obtain an initial incumbent solution. Initialize the branch-and-bound tree as node 0, and let the present set of bounds on the x-variables be as given in Problem QP. Flag node 0 and proceed to Step 1.

Step 1. Range Reductions: Designate the most recently flagged node as the current active node. For the given set of bounds, apply Range Reduction Strategies

1-4. If this indicates a fathoming of the current node, then go to Step 3. Otherwise, proceed to Step 2.

Step 2. Bounding and Branching Step: Scale the node subproblem using the scheme of Section 4.4.1, and solve the problem LD-RLT-NLP to obtain a lower bound on the node subproblem. (If LD-RLT-NLP(SC) is used, then prior to the constraint selection process, scale the problem (3.18) as described in Section 4.4.1 using the bounds $[L^{new}, U^{new}]$ on z-variables, which are readily available as a by-product of Range Reduction Strategy 4.) During the process of solving the Lagrangian dual problem, whenever the incumbent dual solution improves during the conjugate subgradient optimization iterations, check if the fathoming condition (4.15) is satisfied, and proceed to Step 3 if this is the case. Apply the heuristic procedure of Section 4.7, to possibly improve the incumbent solution. Again, if the fathoming rule (4.15) holds, then proceed to Step 3. Otherwise, apply Range Reduction Strategy 3 using the current incumbent dual solution, and select a branching variable according to the branching rule of Section 4.6. Accordingly, partition the current node subproblem by creating two nonactive descendent nodes corresponding to the resulting two sets of (revised) bounds on the branching variable x_{br}, and go to Step 4.

Step 3. Fathoming Step: Fathom the current node. If the sibling of the fathomed node is not active (see Section 4.4.2) then flag that node. Otherwise, flag the nonactive sibling of the highest level node on the path from the current node to the root node, and return to Step 1. If there is no such node, then either stop if there exist no active end nodes, or else, proceed to Step 4.

Step 4. Node Selection Step: If the incumbent solution has improved since the last time Step 4 has been visited, then fathom any active node that satisfies the criterion (4.15). If the number of active end nodes equals *MAXACT*, then select an active end node that has the least lower bound, and flag one of its descendent nodes. On the other hand, if the number of active end nodes is less than *MAXACT*, then along the branch of each such end node, find the lowest level node (closest to the root) that has at least one nonactive descendent node, and among these nodes, flag the nonactive descendant node of the one that has the least lower bound. Return to Step 1.

The convergence of the above algorithm follows from Theorem 1 of Chapter 5, where it is shown that for a more general procedure, any accumulation point of the sequence of solutions generated for the RLT relaxations along any infinite branch solves the Problem QP. Hence, finite convergence to an ε-optimal solution can be obtained.

4.9 Computational Results and Conclusions

We now evaluate the proposed algorithm using a set of test problems chosen from the literature. In addition to the five problems used in Section 3.9, six larger sized ($m = 10, n = 20$) standard test problems from Floudas and Pardalos (1990) (Problems 6-11 in Appendix A), and seven randomly generated problems using the generation scheme of Phillips and Rosen (1990) and Visweswaran and Floudas (1993) of size upto ($m = 20, n = 50$) are solved.

Table 2 presents the results using LD-RLT-NLP as the lower bounding problem. The optimality tolerance (see Section 4.4.3) is taken as 1% for the first five

Table 2. Performance of the Branch-and-Bound Algorithm Using LD-RLT-NLP.

Problem	(m, n)	Known $v[QP]$	v[B&B]	cpu secs.	No. of B&B nodes	Node 0 LB
BLP1	(2, 2)	-1.083	-1.083	0.71	1	-1.089
BLP2	(10, 10)	-45.38	-45.38	1.37	3	-46.02
BLP3	(13, 10)	-794.86	-794.86	2.66	5	-839.02
CQP1	(11, 10)	-267.95	-268.01	1.12	1	-270.68
CQP2	(5, 10)	-39.00	-39.00	1.61	5	-42.96
CQP3	(10, 20)	-394.75	-394.75	8.13	7	-439.03
CQP4	(10, 20)	-884.75	-884.75	2.54	1	-928.92
CQP5	(10, 20)	-8695.01	-8695.01	13.26	11	-9541.67
CQP6	(10, 20)	-754.75	-754.75	5.04	5	-803.31
CQP7	(10, 20)	-4105.28	-4150.41	27.00	25	-5262.57
IQP1	(10, 20)	49318.0	49317.97	2.61	3	45776.43

Table 3. Performance of the Branch-and-Bound Algorithm Using LD-RLT-NLP(SC).

Problem	(m, n)	Known $v[QP]$	v[B&B]	cpu secs.	No. of B&B nodes	Node 0 LB
BLP1	(2, 2)	-1.083	-1.083	0.71	1	-1.089
BLP2	(10, 10)	-45.38	-45.38	1.08	1	-45.81
BLP3	(13, 10)	-794.86	-794.86	3.02	5	-838.85
CQP1	(11, 10)	-267.95	-268.01	1.17	1	-270.69
CQP2	(5, 10)	-39.00	-39.00	1.72	5	-42.95
CQP3	(10, 20)	-394.75	-394.75	3.29	3	-423.32
CQP4	(10, 20)	-884.75	-884.75	2.61	1	-904.02
CQP5	(10, 20)	-8695.01	-8695.01	2.55	1	-9097.99
CQP6	(10, 20)	-754.75	-754.75	2.61	1	-787.60
CQP7	(10, 20)	-4105.28	-4150.41	15.94	11	-5126.68
IQP1	(10, 20)	49318.0	49317.97	2.73	3	44937.40

Table 4. Performance of the Branch-and-Bound Algorithm Using RLT-NLP.

Problem (m, n)		Known $v[QP]$	v[B&B]	cpu secs.	No. of B&B nodes	Node 0 LB
BLP1	(2, 2)	-1.083	-1.083	1.15	1	-1.089
BLP2	(10, 10)	-45.38	-45.38	12.90	1	-45.38
BLP3	(13, 10)	-794.86	-794.86	63.60	1	-794.86
CQP1	(11, 10)	-267.95	-268.01	15.25	1	-268.01
CQP2	(5, 10)	-39.00	-39.00	16.68	3	-39.82

Table 5. Performance of the Branch-and-Bound Algorithm For Randomly Generated Problems Using LD-RLT-NLP(SC).

Problem (m, n)	v[B&B]	Node 0 LB	cpu secs.	No. of B&B nodes
(20, 25)	365.91	349.05	1.71	1
(20, 25)	1170.54	1114.67	1.52	1
(20, 40)	-234.55	-245.38	3.58	1
(20, 40)	-1264.53	-1316.37	2.30	1
(20, 50)	-1310.40	-1373.07	3.84	1
(20, 50)	-1259.00	-1321.87	12.77	1
(20, 50)	-1215.62	-1275.34	4.98	1

Legend: Problem = problem name, (m, n) = size of the problem QP, v[QP] = known optimal (best known) solution of problem QP, v[B&B] = branch-and-bound algorithm incumbent value, cpu secs. = cpu seconds to solve the problem on an IBM 3090 computer, No. of B&B nodes = number of branch-and-bound nodes generated, Node 0 LB = lower bound on QP at root node (optimality criterion is 1% for the first 5 problems, and 5% for the rest of the problems in Tables 2, 3 and 4, and 5% for all the problems in Table 5).

problems and as 5% for the remaining ones. All the problems, except one, are solved under 14 cpu seconds. Problem CQP7 required by far the greatest effort, taking 27 cpu seconds to be solved. However, even when using an optimality criterion of 5%, a better solution than the best known one to CQP7 is found. (This previous best solution reported in Floudas and Pardalos (1990) has an objective value of -4105.2779.) Upon reducing the optimality criterion to 1% and then to 0.1%, CQP7 is solved in 64 cpu seconds and in 205 cpu seconds, respectively, and in both cases, the same solution of value -4150.4087 is obtained (non-zero variables are $x_3 = 1.0429$, $x_{11} = 1.746744$, $x_{13} = 0.4314709$, $x_{16} = 4.43305$, $x_{18} = 15.85893$, $x_{20} = 16.4869$). For both of these cases, 86% of the overall effort is spent in solving the Lagrangian dual problem, showing that this is the determining factor for the total computational effort required. The heuristic procedure of Section 4.7 performed well by identifying the incumbent solution at the root node for all the problems, except for Problem CQP2, for which the optimum was found at the second node. The Range Reduction Strategies of Section 4.5 prove to be very fast and effective; for example, if these reductions are not performed for Problem CQP7, the number of branch-and-bound nodes enumerated increases to 45 (requiring 57 cpu seconds).

Table 3 reports on results obtained using LD-RLT-NLP(SC) as the bounding problem in the branch-and-bound algorithm. Compared to Table 2, the run times have somewhat improved for the larger problems, all problems being solved under 16 cpu seconds. More importantly, although a more relaxed problem is being solved, we observe an improvement in several of the root node lower bounds, especially for larger sized problems. This is principally due to the reduction of the dual search space, which improves the performance of the conjugate subgradient algorithm, while not significantly sacrificing the tightness of the theoretical lower bound. However, when we set the optimality criterion to 1% for CQP7, the algorithm consumed 90 cpu seconds (as compared to 64 cpu seconds for LD-RLT-NLP), while for an accuracy tolerance of 0.1%, the algorithm was prematurely terminated after enumerating the preset limit of 200 nodes in 290 cpu seconds. (Recall that LD-RLT-NLP solved this last instance of the problem in 205 cpu seconds.) Therefore, if a higher degree of accuracy is required, we recommend using the non-reduced problem LD-RLT-NLP where, overall, the marginally tighter representation does play an important role.

Although by Proposition 2, the tightness of the implied bounds on the variables should not affect the result of the bounding problems, this seems to play an important role in the performance of the Lagrangian dual solution procedure. As originally stated, problems CQP3-CQP7 do not include upper bounds on the variables, and for the purpose of the branch-and-bound algorithm, we used the smallest hyperrectangle that contains the feasible region found by minimizing and maximizing each variable over the feasible region. As a comparison, upon using a looser upper bound of 40 on each variable, which is trivially implied by a generalized upper bounding type of constraint present in these problems, CQP3 and CQP7 are solved in 35 and 43 cpu seconds, respectively, using LD-RLT-NLP(SC) as the bounding problem.

Table 4 presents results for the smaller sized problems using RLT-NLP as the bounding problem and solving this by using MINOS 5.1, in lieu of the Lagrangian dual approach used in Table 2. We observe that although the computational time has increased for all problems, the first four problems are solved at the root node itself, while the fifth problem returns an initial lower bound of -39.82 at the root node, the optimum value being -39.00. Note that while there is some loss in the tightness of the bounds due to the inaccuracy in solving RLT-NLP via the

Lagrangian dual approach, the overall gain in efficiency is quite significant. Hence, there exists a great potential for further improvement if the lower bounding problem RLT-NLP could be solved more accurately by the Lagrangian dual scheme.

Using the random problem generator kindly shared by Visweswaran and Floudas (1993) and Phillips and Rosen (1990), we solved seven larger sized problems having upto 20 constraints and 50 variables using the bounding problem LD-RLT-NLP(SC), and an optimality criterion of 5%. These problems are of the form $\min\{\theta_1(0.5\sum_{i=1}^{n}\lambda_i(x_i - \overline{w}_i)^2)\colon Ax \leq b, x \geq 0\}$, where $\theta_1 = -0.001$, and the number of positive and negative components of λ are roughly equal. As reported in Table 5, all the problems are solved at the root node with a reasonable computational effort.

To summarize, in the last two chapters we have investigated RLT based relaxations embedded within a branch-and-bound algorithm for solving nonconvex quadratic programming problems. Tight nonlinear programming relaxations have been defined, and a suitable Lagrangian dual procedure has been designed to solve the relaxations efficiently. The proposed algorithm has been further enhanced by incorporating fast and effective range reduction procedures. Test problems from the literature having upto 20 variables, and randomly generated problems having upto 50 variables have been solved with a reasonable computational effort.

For implementation, we recommend the use of LD-RLT-NLP when a better than 5% accuracy is desired, and the use of the reduced relaxation LD-RLT-NLP(SC) otherwise. For specially structured QP's, especially in the light of Remark 1 and Proposition 2, we strongly suggest that specialized, reduced RLT relaxations be investigated. The eigen-space based relaxation LD-RLT-NLPE is also recommended to be used whenever it can be conveniently constructed, and if there is a significant gap observed between the lower bounds generated via RLT-NLP and RLT-NLPE. For large sized problems, we recommend that the heuristic of Section 4.7 be used, perhaps in concert with solving the RLT based relaxation LD-RLT-NLP(SC) at a limited number of nodes.

The material in Chapters 3 and 4 appears in Sherali and Tuncbilek (1994).

Chapter V

A Global Optimization Algorithm For Polynomial Programming Problems

5.1 *Introduction*

In this chapter, we address a polynomial programming problem which seeks a global minimum to a polynomial objective function subject to a set of polynomial constraint functions, all defined in terms of some continuous decision variables. We do not put any other convexity or generalized convexity restrictions on these functions, but we do assume that the feasible region is compact. A mathematical formulation of this problem is given below.

PP(Ω): Minimize$\{\phi_0(x) \ : \ x \in Z \bigcap \Omega\}$

where $Z = \{x : \phi_r(x) \geq \beta_r$ for $r = 1, \ldots, R_1, \quad \phi_r(x) = \beta_r$ for $r = R_1 + 1, \ldots, R\}$, and $\Omega = \{x : 0 \leq l_j \leq x_j \leq u_j < \infty$, for $j = 1, \ldots, n\}$, and where

$$\phi_r(x) \equiv \sum_{t \in T_r} \alpha_{rt} \left[\prod_{j \in J_{rt}} x_j \right] \quad \text{for } r = 0,1,\ldots, R. \tag{5.1}$$

Here, T_r is an index set for the terms defining $\phi_r(\cdot)$, and α_{rt} are real coefficients for the polynomial terms $(\Pi_{j \in J_{rt}} x_j)$, $t \in T_r$, $r = 0, 1,\ldots, R$. Note that we permit a repetition of indices within each set J_{rt}. For example, if $J_{rt} = \{1, 2, 2, 3\}$, then the corresponding polynomial term is $x_1 x_2^2 x_3$. In particular, let us denote $N = \{1,\ldots, n\}$, and let δ be the maximum degree of any polynomial term appearing in PP(Ω). Define $\overline{N} = \{N,\ldots, N\}$ to be composed of δ replicates of N. Then, each $J_{rt} \subseteq \overline{N}$, with $1 \leq |J_{rt}| \leq \delta$, for $t \in T_r$, $r = 0, 1,\ldots, R$.

In its general form as formulated above, polynomial programs have not received much attention. Notable exceptions are the recent papers by Floudas and Visweswaran (1991) and Shor (1990a). The first of these papers suggests a successive quadratic variable substitution strategy via Shor (1990a) to transform a given polynomial problem to one of minimizing a bilinear function subject to bilinear constraints. Following this, a generalized Benders type of approach is employed, where the Benders cuts are replaced by suitable implied linearized Lagrange functions. A branch and bound algorithm is developed to obtain an ε-optimal solution.

To solve PP(Ω), we propose a branch and bound algorithm which utilizes specially constructed linear bounding problems using a Reformulation Linearization Technique (RLT). In this approach, we generate nonlinear implied constraints by taking the products of bounding terms in Ω up to a suitable order, and also possibly products of other defining constraints of the problem. The resulting problem is subsequently linearized by defining new variables, one for each nonlinear term appearing in the problem. The straightforward mechanics of RLT automatically creates outer linearizations that approximate the closure of the convex hull of the feasible region. In the case of polynomial 0-1 integer programs and polynomial 0-1 mixed integer programs which are linear in the continuous variables when the 0-1 variables are fixed in values, Sherali and Adams (1990, 1994) obtain a hierarchy of such approximations leading to the exact convex hull representation of the feasible region, by using suitable applications of such an RLT procedure. For the jointly constrained biconvex programming problem, which is a special case of PP(Ω), Al-Khayyal and Falk (1983), and Al-Khayyal (1990) constructed linear bounding problems with the motivation to approximate the convex envelopes of the biconvex functions appearing in the constraints and in the objective function. By selecting proper products of lower/upper bounding constraints, their linear programming representation can equivalently be obtained via a restricted application of RLT. Sherali and Alameddine (1990) applied an extended version of RLT to derive stronger bounds in a branch and bound algorithm to solve bilinear programming problems. They also showed that for some special cases of jointly constrained bivariate bilinear programs, RLT yields the exact convex hull representation. In this chapter, we generalize their branch and bound algorithm for solving continuous polynomial programs. A similar partitioning strategy is employed, involving the splitting of the set Ω into smaller hyperrectangles at each stage of the algorithm. However, the main point of difference lies in the construction of an appropriate RLT procedure used in concert with a suitable partitioning of the hyperrectangle in order to ensure the convergence of the algorithm.

The remainder of this chapter is organized as follows. In the next section, we construct the RLT based linear bounding problem. Section 5.3 imbeds this problem in a branch and bound algorithm, and prescribes a partitioning strategy that guarantees the convergence of the overall algorithm. A third order polynomial problem is solved to illustrate the procedure in Section 5.4, and the final Section discusses implementation issues. This material appears in Sherali and Tuncbilek (1992b).

5.2 An RLT Based Linear Bounding Problem

Given Ω, in order to construct the linear programming bounding problem LP(Ω) using RLT, we begin by generating implied constraints using *distinct* products of *bound-factors* $(x_j - l_j) \geq 0$, $(u_j - x_j) \geq 0$, $j \in N$, taken δ at a time. These constraints are of the form

$$F_\delta(J_1, J_2) \equiv \prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j) \geq 0 \tag{5.2}$$

where $(J_1 \bigcup J_2) \subseteq \overline{N}$, $|J_1 \bigcup J_2| = \delta$. The number of distinct constraints of type (5.2) is given by

$$\sum_{k=0}^{\delta} \binom{n+k-1}{k} \binom{n+(\delta-k)-1}{(\delta-k)} = \binom{2n+(\delta-1)}{\delta}$$

After including the constraints (5.2) in the problem PP(Ω), let us substitute

$$X_J = \prod_{j \in J} x_j \qquad \forall J \subseteq \overline{N} \tag{5.3}$$

where the indices in J are assumed to be sequenced in nondecreasing order, and where $X_{\{j\}} \equiv x_j \ \forall\, j \in N$, and $X_\emptyset \equiv 1$. The number of X-variables defined here, besides $X_{\{j\}}, j \in N$, and $X_\emptyset$ is $\binom{n+\delta}{\delta} - (n+1)$. Note that each distinct set J produces one distinct X_J variable, and that when we write $X_{(i,j,k)}$ or $X_{(J_1 \cup J_2)}$ for example, we assume that the indices within $(\cdot)$ are sequenced in nondecreasing order.

REMARK 5: Tighter Linear Programming Representations. Evidently, the constraints of type (5.2) are implied by the set Ω prior to the linearization. However, following the linearization process, these constraints impose useful interrelationships among the product variables X_J. In a likewise manner, we can also generate additional implied constraints in the form of polynomials of degree less than or equal to δ, by taking suitable products of *constraint-factors* $\phi_r(x) - \beta_r \geq 0$, $r = 1, \ldots, R_1$ and/or products of bound-factors with constraint-factors wherever possible. In addition, we can multiply the equality constraints $\phi_r(x) = \beta_r$, $r = R_1 + 1, \ldots, R$, defining Z, by sets of products of variables of the type $\Pi_{j \in J_1} x_j$, $J_1 \subseteq \overline{N}$, so long as the resulting polynomial expression is of degree no more than δ. Incorporating these additional constraints in LP(Ω) after making the substitutions (5.3), produces a tighter linear programming representation. Although this is theoretically admissible and may be computationally advantageous, it is not necessarily required for the results presented in this chapter. Nevertheless, we permit the inclusion of such constraints within LP(Ω) as a user or application driven option.

Proposition 7 below verifies that LP(Ω) is indeed a relaxation of PP(Ω), and gives an important characterization of LP(Ω). Henceforth, we will let $\nu[\cdot]$ denote the value at optimality of the corresponding problem $[\cdot]$.

PROPOSITION 7: $\nu[\text{LP}(\Omega)] \leq \nu[\text{PP}(\Omega)]$. Moreover, if the optimal solution (x^*, X^*) obtained for LP(Ω) satisfies (5.3) for all $J \in \bigcup_{r=0}^{R} \bigcup_{t \in T_r} \{J_{rt}\}$, then x^* solves Problem PP(Ω).

Proof: For any feasible solution $\bar{x}$ to PP(Ω), there exists a feasible solution $(\bar{x}, \bar{X})$ to LP(Ω) with the same objective function value which is constructed using the definition (5.3). Hence, $\nu[\text{LP}(\Omega)] \leq \nu[\text{PP}(\Omega)]$. Moreover, if (5.3) holds for an optimal solution (x^*, X^*) to LP(Ω), for all $J \in \bigcup_{r=0}^{R} \bigcup_{t \in T_r} \{J_{rt}\}$, then x^* is feasible to PP(Ω), and $\nu[\text{LP}(\Omega)] = \Sigma_{t \in T_0} \alpha_{0t} X^*_{J_{0t}} = \Sigma_{t \in T_0} \alpha_{0t} [\Pi_{j \in J_{0t}} x^*_j]$, which equals the objective value of PP(Ω) at $x = x^*$. Hence , x^* solves PP(Ω). ∎

Notice that LP(Ω) does not explicitly contain any constraint that can be generated by constructing products of bound-factors taken less than δ at a time. The next Proposition shows that such constraints are actually implied by those already existing in LP(Ω).

PROPOSITION 8: Let $[f(\cdot)]_\ell$ denote the linearized version of a polynomial function $f(\cdot)$ after making the substitutions (5.3). Then, the constraints $[F_{\delta'}(J_1, J_2)]_\ell \geq 0$, where $(J_1 \cup J_2) \subseteq \overline{N}$, $|J_1 \cup J_2| = \delta'$, $1 \leq \delta' < \delta$, are all implied by the constraints $[F_\delta(J_1, J_2)]_\ell \geq 0$ generated via (5.2), for all distinct ordered pairs $(J_1 \cup J_2) \subseteq \overline{N}$, $|J_1 \cup J_2| = \delta$.

Proof: For any δ', $1 \leq \delta' < \delta$, consider the surrogate of the constraints $[F_{\delta'+1}(J_1 \cup \{t\}, J_2)]_\ell \geq 0$ and $[F_{\delta'+1}(J_1, J_2 \cup \{t\})]_\ell \geq 0$, where $(J_1 \cup J_2) \subseteq \overline{N}$, $|J_1 \cup J_2| = \delta'$, and $t \in N$.

$$[F_{\delta'+1}(J_1 \cup \{t\}, J_2)]_\ell + [F_{\delta'+1}(J_1, J_2 \cup \{t\})]_\ell = [(x_t - l_t)F_{\delta'}(J_1, J_2)]_\ell + [(u_t - x_t)F_{\delta'}(J_1, J_2)]_\ell$$

$$= [x_t F_{\delta'}(J_1, J_2)]_\ell - l_t[F_{\delta'}(J_1, J_2)]_\ell$$

$$+ u_t[F_{\delta'}(J_1, J_2)]_\ell - [x_t F_{\delta'}(J_1, J_2)]_\ell$$

$$= (u_t - l_t)[F_{\delta'}(J_1, J_2)]_\ell \geq 0$$

Since $(u_t - l_t) \geq 0$, it follows that $[F_{\delta'}(J_1, J_2)]_\ell \geq 0$ is implied by $[F_{\delta'+1}(J_1 \cup \{t\}, J_2)]_\ell \geq 0$ and $[F_{\delta'+1}(J_1, J_2 \cup \{t\})]_\ell \geq 0$. The required result follows by the principle of induction, and this completes the proof. ■

The next result establishes an important interrelationship between the newly defined variables X_J, and prompts a partitioning strategy which drives the convergence argument.

PROPOSITION 9: Let $(\bar{x}, \bar{X})$ be a feasible solution to LP(Ω). Suppose that $\bar{x}_p = l_p$. Then

$$\bar{X}_{(J \cup p)} = l_p \bar{X}_J \quad \forall\, J \subseteq \bar{N}, \quad 1 \leq |J| \leq \delta - 1. \tag{5.4}$$

Similarly, if $\bar{x}_p = u_p$, then

$$\bar{X}_{(J \cup p)} = u_p \bar{X}_J \quad \forall\, J \subseteq \bar{N}, \quad 1 \leq |J| \leq \delta - 1. \tag{5.5}$$

Proof: First, consider the case $\bar{x}_p = l_p$. For $|J| = 1$, consider any $q \in N$ (possibly, $q \equiv p$). By Proposition 8, the following constraints are implied by $[(5.2)]_\ell$, where as before, $[\cdot]_\ell$ denotes the linearization of $[\cdot]$ under the substitution (5.3).

$$[(x_p - l_p)(x_q - l_q)]_\ell = X_{(p,q)} - l_q x_p - l_p x_q + l_p l_q \geq 0$$
$$[(x_p - l_p)(u_q - x_q)]_\ell = - X_{(p,q)} + u_q x_p + l_p x_q - l_p u_q \geq 0 \tag{5.6}$$

Hence, we get

$$l_q(x_p - l_p) + l_p x_q \leq X_{(p,q)} \leq l_p x_q + u_q(x_p - l_p). \tag{5.7}$$

By evaluating (5.7) at $(\bar{x}, \bar{X})$, we have $\bar{X}_{(p,q)} = l_p \bar{x}_q$.

Now, let us inductively assume that (5.4) is true for $|J| = 1,\ldots,(t-1)$, and consider $|J| = t$, where $2 \leq t \leq \delta - 1$. For any $q \in J$ (possibly $q \equiv p$), by Proposition 8, the following constraints are implied by $[(5.2)]_\ell$.

$$\left[(x_p - l_p)(x_q - l_q)\prod_{j\in J-q}(x_j - l_j)\right]_\ell \geq 0$$
$$\left[(x_p - l_p)(u_q - x_q)\prod_{j\in J-q}(x_j - l_j)\right]_\ell \geq 0 \tag{5.8}$$

Let us write $\Pi_{j\in J-q}(x_j - l_j) = \Pi_{j\in J-q} x_j + f(x)$, where $f(x)$ is a polynomial in x of degree no more than $t-2$. Then, from (5.8), we have,

$$\begin{aligned}(X_{(J\cup p)} - l_p X_J) &\geq l_q(X_{(J+p-q)} - l_p X_{(J-q)}) + [l_p x_q f(x) - x_p x_q f(x)]_\ell \\ &\quad + l_q[x_p f(x) - l_p f(x)]_\ell \\ (X_{(J\cup p)} - l_p X_J) &\leq u_q(X_{(J+p-q)} - l_p X_{(J-q)}) + [l_p x_q f(x) - x_p x_q f(x)]_\ell \\ &\quad + u_q[x_p f(x) - l_p f(x)]_\ell .\end{aligned} \tag{5.9}$$

Let $(\cdot)|_{(\bar{x},\bar{X})}$ denote the function $(\cdot)$ being evaluated at $(\bar{x}, \bar{X})$. By the induction hypothesis, $\bar{X}_{(J+p-q)} = l_p \bar{X}_{(J-q)}$, $[x_p x_q f(x)]_\ell|_{(\bar{x},\bar{X})} = l_p[x_q f(x)]_\ell|_{(\bar{x},\bar{X})}$, and $[x_p f(x)]_\ell|_{(\bar{x},\bar{X})} = l_p[f(x)]_\ell|_{(\bar{x},\bar{X})}$. Hence, when we evaluate (5.9) at $(\bar{x}, \bar{X})$, the right hand sides of both the inequalities become zero, and this gives $\bar{X}_{(J \cup p)} = l_p \bar{X}_J$.

The case for $\bar{x}_p = u_p$ can be similarly proven by using

$$\begin{aligned}
&\left[(u_p - x_p)(u_q - x_q)\prod_{j \in J-q}(x_j - l_j)\right]_\ell \geq 0 \\
&\left[(u_p - x_p)(x_q - l_q)\prod_{j \in J-q}(x_j - l_j)\right]_\ell \geq 0
\end{aligned} \qquad (5.10)$$

in place of (5.8), and this completes the proof. ■

5.3 A Branch and Bound Algorithm

We are now ready to imbed LP(Ω) in a branch and bound algorithm to solve PP(Ω). The procedure involves the partitioning of the set Ω into sub-hyperrectangles, each of which is associated with a node of the branch and bound tree. Let $\Omega' \subseteq \Omega$ be such a partition. Then, LP(Ω′) gives a lower bound for the node subproblem PP(Ω′). In particular, if $(\bar{x}, \bar{X})$ solves LP(Ω′) and satisfies (5.3) for all $J \in \bigcup_{r=0}^{R} \bigcup_{t \in T_r} \{J_{rt}\}$, then by Proposition 7, $\bar{x}$ solves PP(Ω′), and being feasible to PP(Ω′), the value $v[\text{PP}(\Omega')] \equiv v[\text{LP}(\Omega')]$ provides an upper bound for the problem PP(Ω). Hence, we have a candidate for possibly updating the incumbent solution x^* and its value v^* for PP(Ω). In any case, if $v[\text{LP}(\Omega')] \geq v^*$, we can fathom the node associated with Ω′. Hence, at any stage k of the branch and bound algorithm, we have a set of non-fathomed or *active nodes* denoted as (k, t), for t belonging to some index set T_k, each associated with a corresponding partition $\Omega^{k,t}$ of Ω. We now select an active node (k, t^*), $t^* \in \operatorname{argmin}\{v[\mathrm{LP}(\Omega^{k,t})], t \in T_k\}$, and proceed by decomposing the corresponding $\Omega^{k,t}$ into two sub-hyperrectangles, based on a *branching variable* x_p selected according to the following rule. (Here, $(\bar{x}, \bar{X})$ denotes the optimal solution obtained for $\mathrm{LP}(\Omega^{k,t})$.)

Branching Rule:

$$p \in \operatorname*{argmax}_{j \in N}\{\theta_j\},$$

$$\text{where } \theta_j = \operatorname*{maximum}_{t=1,\ldots,\delta-1} \operatorname*{maximum}_{\substack{J \subseteq \bar{N} \\ |J|=t}} \{|\bar{X}_{(J \cup j)} - \bar{x}_j \bar{X}_J|\} \text{ for each } j \in N \tag{5.11}$$

A formal statement of this procedure is given below.

Step 0: Initialization. Initialize the incumbent solution $x^* = \emptyset$, and let the incumbent objective value $v^* = \infty$. Set $k = 1$ and $T_k = \{1\}$. Denoting $\Omega^{1,1} \equiv \Omega$, solve $\mathrm{LP}(\Omega^{1,1})$ to obtain an optimal solution $(\bar{x}, \bar{X}) \equiv (x^{1,1}, X^{1,1})$, and hence determine a branching variable x_p by using (5.11). If $\theta_p = 0$, then stop; by Proposition 7, $x^{1,1}$ solves the original problem $\mathrm{PP}(\Omega)$. Otherwise, set $t^* = 1$, and proceed to Step 1.

Step 1: Partitioning Step (stage k, $k \geq 1$). Having the active node (k, t^*) to be partitioned, let x_p be its associated branching variable determined via (5.11). Since $\theta_p > 0$, by Proposition 9, $l_p^{k,t^*} < x_p^{k,t^*} < u_p^{k,t^*}$, where $(l_p^{k,t^*}, u_p^{k,t^*})$ are the bounds on x_p in the hyperrectangle Ω^{k,t^*}. Accordingly, partition the set Ω^{k,t^*} into two sub-hyperrectangles

$$\begin{aligned} \Omega^{k,t_1} &= \Omega^{k,t^*} \cap \{x : l_p^{k,t^*} \leq x_p \leq x_p^{k,t^*}\} \\ \Omega^{k,t_2} &= \Omega^{k,t^*} \cap \{x : x_p^{k,t^*} \leq x_p \leq u_p^{k,t^*}\} \end{aligned} \tag{5.12}$$

by picking indices t_1, $t_2 \notin T_k$. After setting $T_k = (T_k - \{t\}) \bigcup \{t_1, t_2\}$, proceed to Step 2.

Step 2: Bounding Step. Solve the linear program $\mathrm{LP}(\Omega^{k,t_1})$ to obtain an optimal solution $(\bar{x}, \bar{X}) \equiv (x^{k,t_1}, X^{k,t_1})$ of objective value $\mathrm{LB}_{k,t_1} \equiv v[\mathrm{LP}(\Omega^{k,t_1})]$. Using this optimal solution in (5.11), determine the corresponding branching variable x_p. If $\theta_p = 0$, then x^{k,t_1} solves the node subproblem $\mathrm{PP}(\Omega^{k,t_1})$. In this case, if $v^* > v[\mathrm{LP}(\Omega^{k,t_1})]$, then update the incumbent solution $x^* = x^{k,t_1}$, and $v^* = \mathrm{LB}_{k,t_1}$. Else, $\theta_p > 0$, and so store the branching variable index p to be possibly used later. Repeat Step 2 after replacing t_1 by t_2, and then proceed to Step 3.

Step 3: Fathoming Step. Fathom any nonimproving nodes by setting $T_{k+1} = T_k - \{t \in T_k : \mathrm{LB}_{k,t} \geq v^*\}$. If $T_{k+1} = \emptyset$, then stop. Otherwise, update $\Omega^{k+1,t} \equiv \Omega^{k,t}$, $(x^{k+1,t}, X^{k+1,t}) \equiv (x^{k,t}, X^{k,t})$, and $\mathrm{LB}_{k+1,t} \equiv \mathrm{LB}_{k,t}$ for all $t \in T_{k+1}$. Increment k by 1, and proceed to Step 4.

Step 4: Node Selection Step. Select an active node (k, t^*), where $t^* \in \operatorname{argmin}\{\mathrm{LB}_{k,t},\ t \in T_k\}$, associated with the least lower bound $\mathrm{LB}_k \equiv \mathrm{LB}_{k,t^*}$ over the active nodes at stage k. Return to Step 1.

THEOREM 1: (Convergence Result) The above algorithm either terminates finitely with the incumbent solution being optimal to $\mathrm{PP}(\Omega)$, or else an infinite sequence of stages is generated such that along any infinite branch of the branch and bound tree, any accumulation point of the x-variable sequence of the linear programming iterates generated at the nodes solves $\mathrm{PP}(\Omega)$.

Proof: The case of finite termination is clear. Hence, suppose that an infinite sequence of stages is generated. Consider any infinite branch of the branch and bound tree, and denote the associated nested sequence of partitions as $\{\Omega^{k,t(k)}\}$, $t(k) \in T_k$ for each $k \in K$, where the indices $(k, t(k))$ used to represent the nodes are selected so that

$$\mathrm{LB}_k = \mathrm{LB}_{k,\,t(k)} \equiv v[\mathrm{LP}(\Omega^{k,\,t(k)})] \quad \forall k \in K. \tag{5.13}$$

By taking any convergent subsequence of $\{x^{k,\,t(k)}\}$, if necessary, assume without loss of generality that

$$\{x^{k,\,t(k)}\}_K \to \bar{x}. \tag{5.14}$$

We must show that $\bar{x}$ solves PP(Ω). First of all, note that since $t(k)$ corresponds to the particular $t^* \in T_k$, for each $k \in K$, and since LB_{k,t^*} is the least lower bound at stage k, we have,

$$v[\mathrm{PP}(\Omega)] \geq \mathrm{LB}_{k,\,t(k)} = \phi_L(x^{k,\,t(k)}, X^{k,\,t(k)}) \quad \forall k \in K, \tag{5.15}$$

where $\phi_L(x, X)$ is the objective function of LP(Ω).

Next, observe that over the infinite sequence of nodes $\Omega^{k,\,t(k)}$, $k \in K$, there exists a variable x_p that is branched on infinitely often via the choice (5.11). Associated with x_p, there must be some index set $J_\infty \subseteq \bar{N}$, $1 \leq |J_\infty| \leq \delta - 1$, which occurs along with p infinitely often in determining θ_p. Let $K_1 \subseteq K$ be the subsequence over which $\max\limits_{j \in N} \theta_j = \theta_p = |\bar{X}_{J_\infty \cup p} - \bar{x}_p \bar{X}_{J_\infty}|$ in (5.11). Then, by (5.11), for each $k \in K_1$, we have

$$|X^{k,\,t(k)}_{(J_\infty \cup p)} - x_p^{k,\,t(k)} X^{k,\,t(k)}_{J_\infty}| \geq |X^{k,\,t(k)}_{(J \cup j)} - x_j^{k,\,t(k)} X_J^{k,\,t(k)}| \quad \forall J \subseteq \bar{N},\ |J| = 1, \ldots, \delta - 1, \\ j = 1, \ldots, n. \tag{5.16}$$

Now, by the boundedness of all sequences generated, there exists a subsequence $K_2 \subseteq K_1$, such that

$$\{x^{k,\,t(k)},\ X^{k,\,t(k)},\ l^{k,\,t(k)},\ u^{k,\,t(k)}\}_{K_2} \to (\bar{x},\ \bar{X},\ \bar{l},\ \bar{u}), \tag{5.17}$$

so that $\{\Omega^{k,t(k)}\}_{K_2} \to \overline{\Omega}$. Hence $(\bar{x}, \overline{X})$ is feasible to LP$(\overline{\Omega})$. Moreover, by virtue of the partitioning (12), we know that for each $k \in K_2$, $x_p^{k,t(k)} \notin (l_p^{k',t(k')}, u_p^{k',t(k')})$ for all $k' \in K_2$, $k' > k$, while $\bar{x}_p \in [\bar{l}_p, \bar{u}_p]$. Hence, we must have $\bar{x}_p = \bar{l}_p$ or $\bar{x}_p = \bar{u}_p$. By Proposition 9, we get

$$\overline{X}_{(J_\infty \cup p)} = \bar{x}_p \overline{X}_{J_\infty} \tag{5.18}$$

But this means from (5.16) that as $k \to \infty$, $k \in K_2$, we have

$$\overline{X}_{(J \cup j)} = \bar{x}_j \overline{X}_J \quad \forall\, J \subseteq \overline{N},\ |J| = 1, \ldots, \delta - 1, \text{ and } j = 1, \ldots, n. \tag{5.19}$$

Hence, the definitions (5.3) hold true for $(\bar{x}, \overline{X})$. Therefore, $\bar{x}$ is feasible to PP(Ω), and moreover,

$$\phi_0(\bar{x}) = \phi_L(\bar{x}, \overline{X}) \geq v[\text{PP}(\Omega)] \tag{5.20}$$

Noting that (5.15) implies upon taking limits as $k \to \infty$, $k \in K_2$, that $v[\text{PP}(\Omega)] \geq \phi_L(\bar{x}, \overline{X})$, we deduce that $v[\text{PP}(\Omega)] = \phi_0(\bar{x})$, and so $\bar{x}$ is optimal to PP(Ω). This completes the proof. ■

REMARK 6: Special Cases. Note that in the spirit of the foregoing algorithmic scheme, we can retain the flexibility of exploiting certain inherent special structures in designing admissible, convergent variants of this procedure. For example, consider a trilinear programming problem (see Zikan (1990) for an application in the context of tracking trajectories) in which $\delta = 3$, with any third order cross product terms being of the type $x_i x_j x_k$ for $1 \leq i \leq n_1$, $n_1 + 1 \leq j \leq n_2$, and $n_2 + 1 \leq k \leq n$, and similarly, any second order cross product term being of the form $x_i x_j$ for i and j lying in two different index sets $\{1, \ldots, n_1\}$, $\{n_1 + 1, \ldots, n_2\}$, and $\{n_2 + 1, \ldots, n\}$. For this problem, we would need to generate in (5.2) only those bound-factor products of order 3 that involve one variable from each index set. Then, Proposition 7 holds as stated, and in Proposition 8, the corresponding second order bound-factor product constraints generated by indices from two different sets are also implied. Consequently, Proposition 9 holds with $(J \bigcup p)$ having at most one index per index set. Accordingly, in (5.11), we only need to consider those $(J \bigcup j)$ which have at most one index from each index set, and the convergence of the resulting algorithm continues to hold by Theorem 1.

REMARK 7: Alternate Branching Variable Selection Rule. In light of Proposition 7 and the proof of Theorem 1, observe that we could have restricted in (5.11) the evaluation of only those $|\overline{X}_{(J \bigcup j)} - \overline{x}_j \overline{X}_J|$ quantities for which the product $\Pi_{i \in (J \bigcup j)} x_i$ appears in some term, or as a subset of some term, in the problem. Then, by the argument evolving around (5.16) in the proof of Theorem 1 and Proposition 7, the convergence of the algorithm would continue to hold.

5.4 An Illustrative Example

To illustrate the branch and bound algorithm of the previous section, we will solve the following nonconvex polynomial program of order $\delta = 3$.

PP(Ω): Minimize $\phi_0(x) = x_1x_2x_3 + x_1^2 - 2x_1x_2 - 3x_1x_3 + 5x_2x_3 - x_3^2 + 5x_2 + x_3$

$$\begin{aligned} &\text{subject to } 4x_1 + 3x_2 + x_3 \le 20 \\ &x_1 + 2x_2 + x_3 \ge 1 \\ &2 \le x_1 \le 5,\ \ 0 \le x_2 \le 10,\ \ 4 \le x_3 \le 8 \end{aligned} \tag{5.21}$$

At stage $k = 1$, $T_1 = \{1\}$, $v^* = \infty$, and $\Omega^{1,1} \equiv \Omega = \{x : 2 \le x_1 \le 5,\ 0 \le x_2 \le 10,\ 4 \le x_3 \le 8\}$. The corresponding linear program LP(Ω) has 56 constraints of type (5.2), linearized by using the substitution (5.3). Two of these constraints are given below as an example.

(*i*) $J_1 = \{1, 2, 3\}, J_2 = \emptyset : [(x_1 - 2)(x_2)(x_3 - 4)]_\ell \ge 0$

$$\rightarrow X_{123} - 4X_{12} - 2X_{23} + 8x_2 \ge 0$$

(*ii*) $J_1 = \{1, 3\}, J_2 = \{2\} : [(x_1 - 2)(x_3 - 4)(10 - x_2)]_\ell \ge 0$

$$\rightarrow X_{123} - 4X_{12} - 10X_{13} - 2X_{23} + 40x_1 + 8x_2 + 20x_3 \le 80$$

Besides the newly generated constraints, LP($\Omega^{1,1}$) contains the original functional constraints of PP(Ω) linearized via (5.3), along with $\Omega^{1,1}$ itself, in its constraint set, and has the objective function

$$\phi_L(x, X) = X_{123} + X_{11} - 2X_{12} - 3X_{13} + 5X_{23} - X_{33} + 5x_2 + x_3.$$

Note that the entire set of variables (x, X) in the problem is given by

$$(x, X) \equiv (x_1, x_2, x_3, X_{11}, X_{12}, X_{13}, X_{22}, X_{23}, X_{33},$$

$$X_{111}, X_{112}, X_{113}, X_{122}, X_{123}, X_{133}, X_{222}, X_{223}, X_{233}, X_{333}).$$

Upon solving LP($\Omega^{1,1}$), we obtain,

$$(x^{1,1}, X^{1,1}) = (3, 0, 8, 8, 0, 24, 0, 0, 64, 20, 0, 64, 0, 0, 192, 0, 0, 0, 512),$$

$$v[\mathrm{LP}(\Omega^{1,1})] = -120.$$

Note that since the constraints of (5.21) are linear, $\bar{x}$ is feasible to (5.21), so that $\phi_0(x^{1,1}) = -119$ is an upper bound on the optimum to (5.21). Hence, the current incumbent solution is $x^* = (3, 0, 8)$ and $v^* = -119$. Using (5.11), we have, $\theta_1 = |X_{113}^{1,1} - x_1^{1,1}X_{13}^{1,1}| = 8$, and $\theta_2 = \theta_3 = 0$. (If we use Remark 7 given at the end of the previous section, then $\theta_1 = |X_{11}^{1,1} - x_1^{1,1}x_1^{1,1}| = 1$). With x_1 as the branching variable ($p = 1$), we partition $\Omega^{1,1}$ as,

$$\Omega^{1,2} = \{x : 2 \le x_1 \le 3, 0 \le x_2 \le 10, 4 \le x_3 \le 8\}$$

$$\Omega^{1,3} = \{x : 3 \le x_1 \le 5, 0 \le x_2 \le 10, 4 \le x_3 \le 8\},$$

and set $T_1 = \{2, 3\}$ at Step 1. Then at Step 2, for node (1, 2), LP($\Omega^{1,2}$) gives

$$(x^{1,2}, X^{1,2}) = (3, 0, 8, 9, 0, 24, 0, 0, 64, 27, 0, 72, 0, 0, 192, 0, 0, 0, 512),$$

$$v[\mathrm{LP}(\Omega^{1,2})] = -119,$$

and for node (1, 3), LP($\Omega^{1,3}$) gives the same solution as for LP($\Omega^{1,2}$). By using this common solution in (5.11), we get $\theta_1 = \theta_2 = \theta_3 = 0$. Hence, this solution is feasible to PP(Ω), but it does not improve the incumbent value. At the fathoming step (Step 3), we fathom the nodes (1, 2) and (1, 3), and since the list of active nodes is now empty, the solution $x^* = (3, 0, 8)$ solves the given problem (5.21).

Finally, let us illustrate the comment given in Remark 5. Suppose that in addition to the bound-factor constraints generated above, we also generate the following constraints:

$$[(u_i - x_i)(u_j - x_j)(20 - 4x_1 - 3x_2 - x_3)]_\ell \geq 0 \text{ for } 1 \leq i \leq j \leq 3.$$

Note that this is a particular restricted set of additional constraints generated in the spirit of Remark 5. Then, it so happens that the augmented linear program LP($\Omega^{1,1}$) itself yields the optimal solution to (5.21), with no branching required in this instance.

5.5 Conclusions and Further Considerations

In this chapter we have presented a generic algorithm for globally optimizing polynomial programming problems based on the use of linear programming relaxations generated via a Reformation Linearization Technique. By incorporating appropriate bound-factor products in this RLT scheme, and employing a suitable partitioning technique that is prompted by the discrepancy between the new variables and the products they represent, a convergent branch and bound algorithm has been developed.

As suggested in Remark 5, and evidenced by the foregoing illustrative example, there is a considerable flexibility, and therefore opportunity, in designing a favorable RLT process for this problem. Several types of implied constraints, or subsets, or surrogates thereof can be generated and added in a linearized form to LP(Ω), thereby tightening its representation at the expense of an increase in size. This poses an obvious question of compromise that needs to be resolved. In the very least, a successive quadratic substitution can be used as in Shor (1990a) to convert the problem into an equivalent quadratically constrained quadratic problem, and then a pairwise bound-factor product RLT can be employed to generate an admissible variant of our algorithm. Moreover, as pointed out in Remark 7, different admissible branching variable selection schemes exist that need to be computationally evaluated. Also, as evident from Remark 6, special classes of polynomial programming problems might possess particular structures that can be exploited in designing special variants of the proposed algorithm. Our motivation here has been to present the basic machinery and methodology. Further investigation and experimentation is necessary to glean an adequate understanding of how best to implement this approach, depending on the actual type of problem being solved. Such issues are pursued in the following chapters.

Chapter VI

Design of RLT-Based Bounding Problems for Polynomial Programming Problems

By far, the most crucial element of the branch-and-bound algorithm proposed in Chapter 5 is the RLT relaxation problem. The strength of this relaxation is a principal determinant of the fathoming power of the algorithm. In this chapter, we address various implementation issues regarding the derivation of this RLT-based lower bounding problem.

As referred to in Chapter 2, Shor (1990a) proposed a method to equivalently transform a polynomial programming programming into a quadratically constrained quadratic programming problem using variable redefinitions. The question of applying RLT to the original polynomial programming problem versus to its equivalent "quadrified" representation is resolved in Section 6.1. Both theoretically and through computational comparisons using test problems from the literature, it is shown that the former scheme yields tighter lower bounds than the latter. In this chapter, we also investigate various RLT constraint generation schemes for one-dimensional and multi-dimensional polynomial programming problems. Our computational experiments show that some special classes of RLT constraints can produce very tight bounds for one-dimensional polynomial programming problems. For the multi-dimensional case, we have also derived additional RLT constraint generation schemes, and demonstrate their application on example problems from the literature.

6.1 Original Versus Quadrified Problems in the RLT Context

This section compares the strategies of applying RLT to the original problem versus applying it to Shor's equivalent quadrified problem. We prove that by applying RLT directly to the original polynomial program, we derive a bound that dominates the value obtained when RLT is applied to the quadratic problem that is constructed by using Shor's quadrification method on the original problem. We also compare these strategies practically through computational tests on some of the problems presented in Appendix A.

In this section, we continue to use the notation introduced in the previous chapter, other than referring to problem PP(Ω) simply as PP whenever it is convenient to do so. Recall that δ is the maximum degree of any polynomial term in PP, and $\overline{N} = \{N,..., N\}$ is composed of δ replicates of $N \equiv \{1,..., n\}$.

Using Shor's notation, let the highest degree of each variable in PP be S_j, $j = 1,..., n$. Let $s_j = |S_j/2|^+$, where $|\alpha|^+$ denotes the operation of rounding up $\alpha \in \mathbb{R}$ to the smallest integer greater than or equal to it. Consider integer vectors

$a = (a_1,\dots, a_n)$ having nonnegative elements, and associate with any such vector, a monomial of the type

$$R[a] = \prod_{j=1}^{n} x_j^{a_j}, \quad \text{where } 0 \le a_j \le s_j,\ a_j \in \mathbb{Z}, \quad j = 1,\dots, n. \tag{6.1}$$

Then, the following system of identity relations are obtained.

$$R[a^{(1)}]R[a^{(2)}] - R[a^{(3)}]R[a^{(4)}] = 0 \tag{6.2}$$

for all sets of four vectors $\{a^{(1)}, a^{(2)}, a^{(3)}, a^{(4)}\}$, where

$$0 \le a^{(1)} + a^{(2)} = a^{(3)} + a^{(4)} \le S \equiv (S_1,\dots, S_n)^t. \tag{6.3}$$

Now, each polynomial function in PP can be written (non-uniquely) in terms of (6.1) as follows:

$$\phi_r(x) = \sum_{t \in T_r} \alpha_{rt} R[a^{(t,1)}]R[a^{(t,2)}] \quad \text{for } r = 0,1,\dots, R \tag{6.4}$$

where

$$\prod_{j=1}^{n} x_j^{(a_j^{(t,1)} + a_j^{(t,2)})} = \prod_{j \in J_{rt}} x_j \quad \forall t \in T_r, \quad r = 0, 1,\dots, R. \tag{6.5}$$

Notice that, in (6.4), $R[a]$ for each distinct a, where $0 \leq a_j \leq s_j$, $a_j \in \mathbb{Z}$, is a single variable that represents a monomial as defined in (6.1). Using successive substitutions of the type (6.2), the polynomial terms in x-variables can be reduced to quadratic terms in the $R[\cdot]$-variables. A particular reduction of this type can be conducted by setting $a^{(2)} = 0$ in (6.2), and increasing the degree of the underlying polynomial term each time by one. (Notice that $R[0] = 1$, according to (6.1).) For example, the term $x_1^3 x_2^2 x_3^4$ can be reduced to a quadratic form by using the following series of identity relations.

$x_1 \equiv R[1, 0, 0], \quad x_2 \equiv R[0, 1, 0], \quad x_3 \equiv R[0, 0, 1]$

$x_3^2 \equiv R[0, 0, 2] = R[0, 0, 1]R[0, 0, 1]$

$x_2 x_3^2 \equiv R[0, 1, 2] = R[0, 0, 2]R[0, 1, 0]$

$x_1 x_2 x_3^2 \equiv R[1, 1, 2] = R[0, 1, 2]R[1, 0, 0]$

$x_1^2 x_2 x_3^2 \equiv R[2, 1, 2] = R[1, 1, 2]R[1, 0, 0]$

$x_1^3 x_2^2 x_3^4 \equiv R[1, 1, 2]R[2, 1, 2]$

Using the last identity, the polynomial term $x_1^3 x_2^2 x_3^4$ is expressed as a quadratic term in terms of $R[\cdot]$-variables. To enforce the inter-relationship among the $R[\cdot]$-variables, the other quadratic equality constraints given above are also included in the problem. Notice that this successive quadrification scheme is not uniquely defined; for example, instead of writing $x_2 x_3^2$ as $(x_2)(x_3^2)$ as essentially done above, we could have as well used $x_2 x_3^2 = (x_3)(x_2 x_3)$ as follows:

$x_2 x_3 \equiv R[0, 1, 1] = R[0, 1, 0]R[0, 0, 1]$

$x_2 x_3^2 \equiv R[0, 1, 2] = R[0, 1, 1]R[0, 0, 1].$

Therefore, using (6.4), and including the identity relations (6.2) in PP, we obtain an equivalent quadratically constrained quadratic programming problem. We refer to this process as "quadrification" and denote the resulting (quadratic polynomial programming) problem by **QPP**. In the following discussion, we show that applying RLT to PP gives a tighter relaxation than applying RLT to QPP. Hence, if **LP(PP)** and **LP(QPP)** are the resulting linear programming relaxations produced, we have, $\nu[\text{LP(PP)}] \geq \nu[\text{LP(QPP)}]$.

To generate RLT constraints for QPP by using bound-factor products, we need to derive simple bounds on each $R[\cdot]$-variable of the type given in (6.1). For this purpose, using the specified bounds $0 \leq l_j \leq x_j \leq u_j < \infty$, we deduce lower and upper bounds $L[\cdot]$ and $U[\cdot]$, respectively, on the $R[\cdot]$-variables as follows:

$$L[a] = \prod_{j=1}^{n} l_j^{a_j}, \quad U[a] = \prod_{j=1}^{n} u_j^{a_j}, \quad \forall\, a, \text{ where } 0 \leq a_j \leq s_j, \quad a_j \in \mathbb{Z}, \quad j = 1, \ldots, n. \tag{6.6}$$

Using these bounds, we now generate the following bound-product based RLT constraints for QPP.

$$\left[(R[a^{(1)}] - L[a^{(1)}])(R[a^{(2)}] - L[a^{(2)}])\right]_\ell \geq 0, \tag{6.7.1}$$

$$\left[(U[a^{(1)}] - R[a^{(1)}])(U[a^{(2)}] - R[a^{(2)}])\right]_\ell \geq 0, \tag{6.7.2}$$

$$\left[(R[a^{(1)}] - L[a^{(1)}])(U[a^{(2)}] - R[a^{(2)}])\right]_\ell \geq 0, \tag{6.7.3}$$

$$\left[(U[a^{(1)}] - R[a^{(1)}])(R[a^{(2)}] - L[a^{(2)}])\right]_\ell \geq 0, \tag{6.7.4}$$

for all $R[a^{(1)}]$ and $R[a^{(2)}]$ generated by the quadrification scheme (6.1) – (6.4) such that $a^{(1)} \neq a^{(2)}$ for (6.7.4).

In the linearization phase, each nonlinear term is then replaced by a single variable according to the following definition.

$$W[a^{(1)}, a^{(2)}] \equiv R[a^{(1)}]R[a^{(2)}] \tag{6.8}$$

Using (6.8) in the identity relation constraints (6.2), we observe that only a single RLT variable $W[a^{(1)}, a^{(2)}]$ is needed for all quadratic terms $R[a^{(p)}]R[a^{(q)}]$ such that $a^{(p)} + a^{(q)} = a^{(1)} + a^{(2)}$.

Notice that the generated $R[\cdot]$-variables, and consequently the RLT constraints (6.7) depend on the particular choice of identity relations (6.2) used in the quadrification scheme. To eliminate the possible effect of employing alternative quadrification schemes on the resulting RLT problem, let us generate the identity relation constraints (6.2) *for all* $0 \le a^{(1)} + a^{(2)} = a^{(3)} + a^{(4)} \le S$. Additional constraints generated in this fashion are redundant for the original problem QPP. However, the resulting set of identity relation constraints introduces all possible $R[\cdot]$-variables that can appear in any quadrification scheme to obtain a representation QPP. Furthermore, upon linearization under (6.8), the constraints $[(6.2)]_\ell$ uniquely associate a single RLT variable $W[\cdot]$ with its underlying polynomial term in x-variables, regardless of the particular way in which this polynomial term has been represented in terms of the $R[\cdot]$-variables. Therefore, we can discard these constraints by simply defining a single RLT variable for each distinct polynomial term in the linearization phase. This result also enables us to omit the intermediate step of quadrification, and directly generate the bound-factor products that would have involved all possible $R[\cdot]$-variables arising via alternative quadrification schemes, in terms of the x-variables as follows:

$$\left(\prod_{j=1}^{n} x_j^{a_j^{(1)}} - \prod_{j=1}^{n} l_j^{a_j^{(1)}}\right)\left(\prod_{j=1}^{n} x_j^{a_j^{(2)}} - \prod_{j=1}^{n} l_j^{a_j^{(2)}}\right) \geq 0 \qquad (6.9.1)$$

$$\left(\prod_{j=1}^{n} u_j^{a_j^{(1)}} - \prod_{j=1}^{n} x_j^{a_j^{(1)}}\right)\left(\prod_{j=1}^{n} u_j^{a_j^{(2)}} - \prod_{j=1}^{n} x_j^{a_j^{(2)}}\right) \geq 0 \qquad (6.9.2)$$

for all distinct *unordered* pairs $(a^{(1)}, a^{(2)}) \in \mathbb{Z}^{2n}$, such that $0 \leq a_j^{(1)}, a_j^{(2)} \leq s_j$, $j = 1, \ldots, n$, and $\sum_{j=1}^{n} (a_j^{(1)} + a_j^{(2)}) \leq \delta$,

$$\left(\prod_{j=1}^{n} x_j^{a_j^{(1)}} - \prod_{j=1}^{n} l_j^{a_j^{(1)}}\right)\left(\prod_{j=1}^{n} u_j^{a_j^{(2)}} - \prod_{j=1}^{n} x_j^{a_j^{(2)}}\right) \geq 0 \qquad (6.9.3)$$

for all distinct *ordered* pairs $(a^{(1)}, a^{(2)}) \in \mathbb{Z}^{2n}$, such that $0 \leq a_j^{(1)}, a_j^{(2)} \leq s_j$, $j = 1, \ldots, n$, and $\sum_{j=1}^{n} (a_j^{(1)} + a_j^{(2)}) \leq \delta$.

Including (6.9) in PP, and then defining a single variable for each distinct *polynomial term* appearing in the constraints (6.9) along with the original PP constraints in the set Z, we obtain an all encompassing linear programming RLT relaxation for any quadrified problem. We denote this lower bounding problem by **LP($\overline{\text{QPP}}$)**. Note that the linearized constraints from Z are common to both problems LP(PP) and LP($\overline{\text{QPP}}$). Hence, in order to establish the stated dominance, we need to show that the RLT constraints $[(6.9)]_\ell$ in LP($\overline{\text{QPP}}$) are all implied by the regular RLT constraints of LP(PP). For convenience, the latter constraints are repeated below.

$$\left[\prod_{j\in J_1}(x_j - l_j)\prod_{j\in J_2}(u_j - x_j)\right]_\ell \geq 0 \quad \text{where } (J_1 \cup J_2) \subseteq \overline{N}, \quad |J_1 \cup J_2| = \delta \tag{6.10}$$

Note that, if $n_o = |\{j\in\{1,\ldots, n\} : S_j \text{ odd}, S_j \neq 1\}| > 0$, where $|\{\cdot\}|$ denotes the cardinality of the set $\{\cdot\}$, we need to replace δ by $\delta + n_o$ in (6.9). In this case, the commensurate regular RLT constraints (6.10) we consider for comparison will also be of order $\delta + n_o$. In order to not further complicate the notation, we continue to use δ in (6.9) and (6.10), and assume that δ should be replaced by $(\delta + n_o)$ below whenever $n_o > 0$.

In order to prove the dominance result, we first show that the defined terms of the type $\left(\prod_{j=1}^{n} x_j^{a_j^{(1)}} - \prod_{j=1}^{n} l_j^{a_j^{(1)}}\right)$ and $\left(\prod_{j=1}^{n} u_j^{a_j^{(1)}} - \prod_{j=1}^{n} x_j^{a_j^{(1)}}\right)$ as well as their products, can each be expressed as a sum of nonnegative multiples of ordinary bound-factor and nonnegative variable products. Hence, a linearization of products of such compound factors can likewise be expressed as a sum of linearizations of the latter type of ordinary nonnegative bound-factor products. Then, by showing that such latter products are themselves implied by the RLT constraints defining LP(PP), we will establish the dominance result.

PROPOSITION 10: The terms $\prod_{j=1}^{n}(x_j^{p_j} - l_j^{p_j})$ and $\prod_{j=1}^{n}(u_j^{p_j} - x_j^{p_j})$, where $p\in\mathbb{Z}^n$, $\sum_{j=1}^{n} p_j \leq \delta$, as well as the term $\prod_{j=1}^{n}(x_j^{p_j} - l_j^{p_j})\prod_{j=1}^{n}(u_j^{q_j} - x_j^{q_j})$, where $p, q\in\mathbb{Z}^n$, $\sum_{j=1}^{n}(p_j + q_j) \leq \delta$, are each comprised of nonnegative multiples of terms of the type

$$\prod_{j\in J_1}(u_j-x_j)\prod_{j\in J_2}(x_j-l_j)\prod_{j\in J_3}x_j \text{ where } (J_1\cup J_2\cup J_3)\subseteq\overline{N},\ |J_1\cup J_2\cup J_3|\le\delta \tag{6.11}$$

Proof: By the binomial expansion, we know that,

$$(y+a)^p=\binom{p}{0}y^pa^0+\binom{p}{1}y^{p-1}a^1+\ldots+\binom{p}{p-1}y^1a^{p-1}+\binom{p}{p}y^0a^p. \tag{6.12}$$

Hence, putting $a=l_j$, $y=(x_j-l_j)$, $p=p_j$, for $j\in N$ in (6.12), we get

$$(x_j^{p_j}-l_j^{p_j})=(x_j-l_j)^{p_j}+p_jl_j(x_j-l_j)^{p_j-1}+\frac{p_j(p_j-1)}{2}l_j^2(x_j-l_j)^{p_j-2}+\ldots+p_jl_j^{p_j-1}(x_j-l_j). \tag{6.13}$$

Therefore, $\prod_{j=1}^{n}(x_j^{p_j}-l_j^{p_j})$, $\sum_{j=1}^{n}p_j\le\delta$, is comprised of the sum of nonnegative multiples of terms of the type $\prod_{j\in J_2}(x_j-l_j)$, $J_2\subseteq\overline{N}$, $|J_2|\le\delta$, noting that $J_1=J_3=\emptyset$ in this case.

Now, putting $a=x_j$, $y=(u_j-x_j)$, $p=p_j$, for $j\in N$ in (6.12), we get

$$(u_j^{p_j}-x_j^{p_j})=(u_j-x_j)^{p_j}+p_jx_j(u_j-x_j)^{p_j-1}+\frac{p_j(p_j-1)}{2}x_j^2(u_j-x_j)^{p_j-2}+ \\ \ldots+p_jx_j^{p_j-1}(u_j-x_j). \tag{6.14}$$

Hence, $\prod_{j=1}^{n}(u_j^{p_j}-x_j^{p_j})$, $\sum_{j=1}^{n}p_j\le\delta$, is comprised of the sum of nonnegative multiples of terms of the type $\prod_{j\in J_1}(u_j-x_j)\prod_{j\in J_3}x_j$, $(J_1\cup J_3)\subseteq\overline{N}$, $|J_1\cup J_3|\le\delta$, with $J_2=\emptyset$ in this case. Finally, from (6.13) and (6.14), it can be easily seen that $\prod_{j=1}^{n}(x_j^{p_j}-l_j^{p_j})\prod_{j=1}^{n}(u_j^{q_j}-x_j^{q_j})$, where $p,q\in\mathbb{Z}^n$, $\sum_{j=1}^{n}(p_j+q_j)\le\delta$, is comprised of the sum of nonnegative multiples of terms of the type $\prod_{j\in J_1}(u_j-x_j)\prod_{j\in J_2}(x_j-l_j)\prod_{j\in J_3}x_j$ where $(J_1\cup J_2\cup J_3)\subseteq\overline{N}$, $|J_1\cup J_2\cup J_3|\le\delta$. This completes the proof. ■

PROPOSITION 11: For any integer vector $p \in \mathbb{Z}^n$, such that $p_j \geq 0$, $j = 1,\ldots, n$, and $\sum_{j=1}^{n} p_j \leq \delta$, the terms of the type $\left(\prod_{j=1}^{n} x_j^{p_j} - \prod_{j=1}^{n} l_j^{p_j}\right)$ and $\left(\prod_{j=1}^{n} u_j^{p_j} - \prod_{j=1}^{n} x_j^{p_j}\right)$ can each be expressed as a sum of nonnegative multiples of terms of the following form:

$$\prod_{j \in J_1} (u_j - x_j) \prod_{j \in J_2} (x_j - l_j) \prod_{j \in J_3} x_j \tag{6.15}$$

where $(J_1 \cup J_2 \cup J_3) \subseteq \overline{N}$, $|J_1 \cup J_2 \cup J_3| \leq \delta$.

Proof: We first show that $\left(\prod_{j=1}^{n} x_j^{p_j} - \prod_{j=1}^{n} l_j^{p_j}\right)$ can be expressed in terms of nonnegative multiples of products of factors $(x_j^{p_j} - l_j^{p_j})$, $j = 1,\ldots, n$. We will use induction on the number of variables n. For $n = 1$, the result is trivial. Hence, assume that the induction hypothesis is true for $n = 1,\ldots, k-1$, $k < \infty$, and consider the case $n = k$.

$$\begin{aligned}\left(\prod_{j=1}^{k} x_j^{p_j} - \prod_{j=1}^{k} l_j^{p_j}\right) = &\left(\prod_{j=1}^{k-1} x_j^{p_j} - \prod_{j=1}^{k-1} l_j^{p_j}\right)(x_k^{p_k} - l_k^{p_k}) + (x_k^{p_k} - l_k^{p_k}) \prod_{j=1}^{k-1} l_j^{p_j} \\ &+ l_k^{p_k}\left(\prod_{j=1}^{k-1} x_j^{p_j} - \prod_{j=1}^{k-1} l_j^{p_j}\right)\end{aligned} \tag{6.16}$$

By the induction principle, the right-hand side of (6.16) is therefore also the sum of nonnegative multiples of products of factors $(x_j^{p_j} - l_j^{p_j})$, $j = 1,\ldots, n$. Hence, by (the proof of) Proposition 10, and using the inductive expansion as in the right-hand side of (6.16), $\left(\prod_{j=1}^{n} x_j^{p_j} - \prod_{j=1}^{n} l_j^{p_j}\right)$ can be expressed as a sum of nonnegative multiples of terms of the form $\prod_{j \in J_2} (x_j - l_j)$, where $J_2 \in \overline{N}$, and $|J_2| \leq \sum_{j=1}^{n} p_j$.

In a similar fashion, we now show that $\left(\prod_{j=1}^{n} u_j^{p_j} - \prod_{j=1}^{n} x_j^{p_j}\right)$ can be expressed in terms of nonnegative multiples of products of factors $(u_j^{p_j} - x_j^{p_j})$ and $x_j^{p_j}$, $j = 1,\ldots, n$.

We will use induction on the number of variables n. For $n = 1$, the result is trivial. Hence, assume that the induction hypothesis is true for $n = 1,\ldots, k-1$, $k < \infty$, and consider the case $n = k$.

$$\left(\prod_{j=1}^{k} u_j^{p_j} - \prod_{j=1}^{k} x_j^{p_j}\right) = u_k^{p_k}\left(\prod_{j=1}^{k-1} u_j^{p_j} - \prod_{j=1}^{k-1} x_j^{p_j}\right) + (u_k^{p_k} - x_k^{p_k})\prod_{j=1}^{k-1} x_j^{p_j} \tag{6.17}$$

By the induction principle, the right-hand side of (6.17) is therefore also the sum of nonnegative multiples of products of factors $(u_j^{p_j} - x_j^{p_j})$ and $x_j^{p_j}$, $j = 1,\ldots, n$. Hence, by (the proof of) Proposition 10, and using the inductive expansion as in the right-hand side of (6.17), $\left(\prod_{j=1}^{n} u_j^{p_j} - \prod_{j=1}^{n} x_j^{p_j}\right)$ can be expressed as a sum of nonnegative multiples of terms of the form $\prod_{j\in J_1}(u_j - x_j)\prod_{j\in J_3} x_j$, where $(J_1 \cup J_3) \subseteq \overline{N}$, $|J_1 \cup J_3| \leq \delta$. This completes the proof. ■

To prove the dominance result, we need one additional intermediary step, relating the constraints of type $[(6.15)]_\ell \geq 0$ with the regular RLT constraints (6.10). Let us first define the following family of sets of constraints to identify all possible constructs of type (6.15).

$$\Omega_{s,\delta'} \equiv \left[\begin{array}{l} \left[\prod_{j\in J_1}(u_j - x_j)\prod_{j\in J_2}(x_j - l_j)\prod_{j\in J_3} x_j\right]_\ell \geq 0 \\ \forall \text{ distinct ordered triplets } (J_1, J_2, J_3) \text{ where} \\ (J_1 \cup J_2 \cup J_3) \subseteq \overline{N},\ |J_1| + |J_2| + |J_3| = \delta',\ |J_3| = s \end{array}\right] \tag{6.18}$$

for each $0 \leq s \equiv |J_3| \leq \delta' \leq \delta$. Notice that $\Omega_{0,\delta}$ is the set of regular RLT constraints (6.10). Although there is a direct relationship between the set of constraints (6.18) and the feasible region defined by these constraints, to prevent any confusion, let $\overline{\Omega}_{s,\delta'}$ denote the feasible region defined by the constraint set $\Omega_{s,\delta'}$.

PROPOSITION 12: **(a)** For any $0 \le s < \delta$, $\overline{\Omega}_{s,\delta'} \supseteq \overline{\Omega}_{s,\delta'+1}$ for all $\delta' \in \{s, \ldots, \delta - 1\}$. **(b)** For any $\delta' \in \{s, s+1, \ldots, \delta\}$, $\overline{\Omega}_{s-1,\delta'} \subseteq \overline{\Omega}_{s,\delta'}$ for all $1 \le s \le \delta$.

Proof: For any $0 \le s = |J_3| < \delta$ and $\delta' \in \{s, \ldots, \delta - 1\}$, consider the following constraint from $\Omega_{s,\delta'}$:

$$\left[\prod_{j \in J_1}(u_j - x_j)\prod_{j \in J_2}(x_j - l_j)\prod_{j \in J_3}x_j\right]_\ell \ge 0. \tag{6.19}$$

Surrogating the following constraints from the set $\Omega_{s,\delta'+1}$,

$$\left[(x_t - l_t)\prod_{j \in J_1}(u_j - x_j)\prod_{j \in J_2}(x_j - l_j)\prod_{j \in J_3}x_j\right]_\ell \ge 0 \tag{6.20.1}$$

$$\left[(u_t - x_t)\prod_{j \in J_1}(u_j - x_j)\prod_{j \in J_2}(x_j - l_j)\prod_{j \in J_3}x_j\right]_\ell \ge 0 \tag{6.20.2}$$

where $t \in \{1, \ldots, n\}$, we obtain,

$$(6.20.1) + (6.20.2) = \left[(u_t - l_t)\prod_{j \in J_1}(u_j - x_j)\prod_{j \in J_2}(x_j - l_j)\prod_{j \in J_3}x_j\right]_\ell \ge 0.$$

Since $(u_t - l_t) > 0$, constraint (6.19) is implied by $\Omega_{s,\delta'+1}$. Since this is true for any constraint from $\Omega_{s,\delta'}$, we have $\overline{\Omega}_{s,\delta'} \supseteq \overline{\Omega}_{s,\delta'+1}$. This proves part (a).

To prove part (b), for any $1 \leq s \leq \delta' \leq \delta$ and $t \in \{1, \ldots, n\}$, consider the following constraint from $\Omega_{s,\delta'}$:

$$\left[x_t \prod_{j \in J_1} (u_j - x_j) \prod_{j \in J_2} (x_j - l_j) \prod_{j \in J_3} x_j \right]_\ell \geq 0 \tag{6.21}$$

for any $|J_3| = s - 1$, $J_1 \bigcup J_2 \bigcup J_3 \bigcup \{t\} \subseteq \overline{N}$, $|J_1| + |J_2| + |J_3| + 1 = \delta'$. Notice that (6.21) is well defined, since $s = |J_3 \bigcup \{t\}| \geq 1$. We can obtain (6.21) by a particular surrogate of the following constraint from $\Omega_{s-1,\delta'}$,

$$\left[(x_t - l_t) \prod_{j \in J_1} (u_j - x_j) \prod_{j \in J_2} (x_j - l_j) \prod_{j \in J_3} x_j \right]_\ell \geq 0 \tag{6.22.1}$$

with the following constraint from $\Omega_{s-1,\delta'-1}$, which is shown to be implied by the constraints in $\Omega_{s-1,\delta'}$ in part (a) of this proposition,

$$\left[\prod_{j \in J_1} (u_j - x_j) \prod_{j \in J_2} (x_j - l_j) \prod_{j \in J_3} x_j \right]_\ell \geq 0. \tag{6.22.2}$$

Using $l_t \geq 0$ as the weight for (6.22.2), we have (6.22.1) + l_t(6.22.2) = (6.21). Hence, each constraint in $\Omega_{s,\delta'}$ is implied by those in $\Omega_{s-1,\delta'}$, and this completes the proof.

■

Having established that the regular RLT constraints $(6.10) \equiv \Omega_{(0,\delta)}$ imply all the constraints in the set $\{\Omega_{(s,\delta')}, \forall\ 0 \leq s \leq \delta' \leq \delta\}$, we are finally ready to establish the dominance result.

THEOREM 2: $v[\text{LP(PP)}] \geq v[\text{LP}(\overline{\text{QPP}})]$.

Proof: By Proposition 10 and 11, the compound factors $\left(\prod_{j=1}^{n} x_j^{\alpha_j} - \prod_{j=1}^{n} l_j^{\alpha_j}\right)$ and $\left(\prod_{j=1}^{n} u_j^{\alpha_j} - \prod_{j=1}^{n} x_j^{\alpha_j}\right)$ can be expressed as sums of nonnegative multiples of the terms of the form (6.15). Hence, the pairwise products of these compound factors, namely (6.9.1)-(6.9.3), can also be expressed as sums of nonnegative multiples of the terms of the form (6.15). Consequently, upon linearization, $[(6.9.1)]_\ell$, $[(6.9.2)]_\ell$ and $[(6.9.3)]_\ell$ can be obtained as surrogates of the constraints from the combined set $\{\Omega_{s,\delta'},\ 0 \leq s \leq \delta' \leq \delta\}$. Since $\overline{\Omega}_{0,\delta} \subseteq \overline{\Omega}_{s,\delta'}$ for all $0 \leq s \leq \delta' \leq \delta$ by Proposition 12, the RLT constraints of Problem LP($\overline{\text{QPP}}$), that is, the constraints $[(6.9.1)]_\ell$, $[(6.9.2)]_\ell$ and $[(6.9.3)]_\ell$, are implied by $\Omega_{0,\delta}$, which are the RLT constraints of problem LP(PP). Furthermore, the remaining linearized original constraints from set Z, and the linearized objective function are identical in both problems. Therefore, LP(PP) yields a tighter relaxation than LP($\overline{\text{QPP}}$), and consequently, v[LP(PP)] $\geq v$[LP($\overline{\text{QPP}}$)]. This completes the proof. ■

Notice that in establishing the above result, we have not used the restriction $a_j^{(1)}, a_j^{(2)} \leq s_j$, $j = 1,\ldots, n$, for generating the bound-factor products (6.9). Removal of this restriction from (6.9) results in the generation of additional RLT constraints to be included in LP($\overline{\text{QPP}}$) upon linearization. Therefore, the above dominance result holds even when considering a potentially tighter linear programming relaxation than LP($\overline{\text{QPP}}$).

We compare v[LP(PP)] and v[LP($\overline{\text{QPP}}$)] empirically below, using some problems from Appendix A. All the problems are scaled so that $(l, u) = (0, 1)$, where l, u are the vectors of lower and upper bounds, respectively, on the variables. The results in Table 6 show that the lower bounds via LP($\overline{\text{QPP}}$) are worsened compared to those obtained via LP(PP); in particular, the difference is significant for Problems 13, 16 and 20. In accordance with our earlier remark, fourth order RLT constraints are used in LP(PP) for Problem 12, where the resulting lower

Table 6. Comparison of LP(PP) versus LP($\overline{QPP}$).

Problem	Known $v[PP]$	v[LP(PP)]	v[LP($\overline{QPP}$)]
Problem 12	-4.5	-4.5	-9
Problem 13	0	-875	-6375.0
Problem 16	7.0	-17385.0	-2,292,825.0
Problem 19	-5.50796	-6.750	-6.9867
Problem 20	-118.705	-8108.0	-54764.0
Problem 21	-16.7389	-28.5	-29.0

Legend: Problem = Problems from Appendix A, v[PP] = Known optimal solution of problem PP (best), v[LP(PP)] = Lower bound via LP(PP), v[LP($\overline{QPP}$)] = Lower bound via LP($\overline{QPP}$).

bound matches the global minimum. Even when we use RLT constraints of order three, LP(PP) yields a lower bound of -6, which still dominates the lower bound obtained via LP($\overline{\text{QPP}}$).

To examine the consequences of generating a stronger relaxation LP($\overline{\text{QPP}}$) as mentioned after the proof of Theorem 2, we removed the restriction $a_j^{(1)}, a_j^{(2)} \leq s_j$, $j = 1, \ldots, n$, from (6.9) for Problems 13 and 20. Although the inclusion of the consequent additional constraints improved the lower bounds to -5241.67 and -46654 for Problems 13 and 20, respectively, the difference from v[LP(PP)] still remains significant.

6.2 One-Dimensional Polynomial Programming Problems

In this section, we investigate various bounding strategies for one-dimensional polynomial programming problems. Motivated by our earlier development for quadratic programming problems, we include simple nonlinear convex constraints in the regular bound-factor product RLT representation. These constraints impose certain lower bounding types of restrictions on some individual RLT variables. In the second bounding scheme, we transform the one-dimensional polynomial programming problem into a DC-programming problem (see Horst and Tuy, 1993) by partitioning the objective function into convex and concave parts. This enables us to generate constraint-factors that are based on first-order approximations of the convex part of the objective function. Taking products of these constraint-factors along with the bound-factors, we generate an RLT bounding problem. In the next three consecutive alternative bounding schemes, we derive various classes of implied polynomial constraints and subsequently linearize them to augment the convex RLT bounding problem. We also discuss a recent exponential transformation based approach by Maranas and Floudas (1993). As we present these bounding schemes in separate subsections below, we use Problem 16 from Appendix A to illustrate their characteristics. Finally, we compare the performance of the foregoing bounding schemes using various example problems from the literature (see Visweswaran and Floudas (1992)).

For a single variable, a polynomial programming problem reduces to minimizing (or maximizing) a polynomial function over a compact interval. For numerical stability purposes, given a problem $\min\{f^o(y): l \leq y \leq u\}$, we scale the variable y as follows,

$$x = \frac{y-l}{u-l} \qquad \text{so that} \qquad y = (u-l)x + l, \tag{6.23}$$

to obtain an equivalent problem over the unit interval. Note that we retain any resulting constants so that

$$\min\{f^o(y):\ l \leq y \leq u\} \equiv \min\{f(x):\ 0 \leq x \leq 1\}. \tag{6.24}$$

For example, after scaling (6.23), Problem 16 is expressed as

$$\begin{aligned}\text{Minimize } \{&10^6x^6 - 30(10^5)x^5 + 360(10^4)x^4 - 2200(10^3)x^3 + 7152(10^2)x^2 \\ &- 115200x + 7175 :\ 0 \leq x \leq 1\}.\end{aligned} \tag{6.25}$$

For this scaled problem, the global optimal solution is $x^* \in \{0.2 \text{ or } 0.8\}$, where $f(x^*) = 7$. As the graph of the objective function of (6.25) in Figure 2 shows, the steepness of the objective function, especially around the boundaries of the feasible interval, makes it difficult to construct tight lower bounding approximations for this problem.

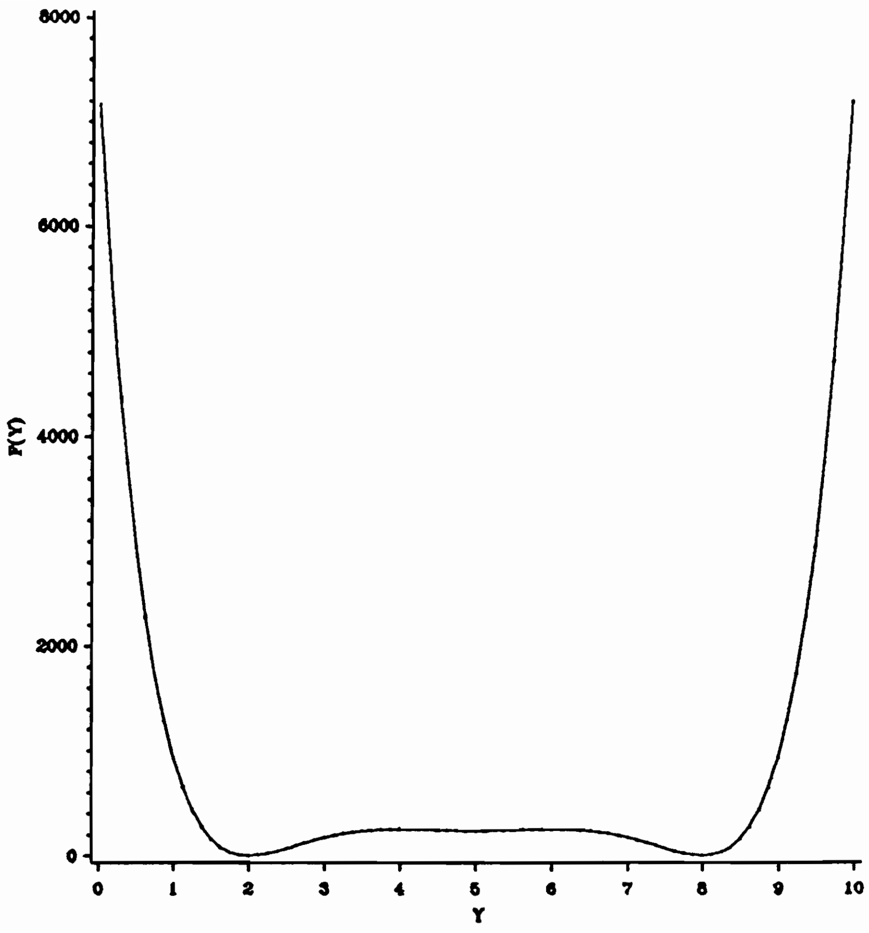

Figure 2. Graph of the objective function in problem (16), where the feasible interval is scaled to be $0 \leq y \leq 10$.

Before proceeding with our presentation, let us introduce a piece of notation that will be used in various places in this section. Let $|\alpha|^-$ denote the greatest integer that is less than or equal to $\alpha \in \mathbb{R}$ (floor of α).

6.2.1 Convex RLT Bounding Problem

Recall that in Sections 3.4 and 3.5, we have introduced certain simple non-linear convex constraints within the linear RLT lower bounding problem in order to improve the resulting lower bound, while causing no additional computational burden on the Lagrangian dual subproblem solution procedure. These constraints can be easily generalized for polynomial programming problems. Considering the *scaled* one-dimensional polynomial programming problem (6.24) of degree δ over the unit interval, we can generate the following convex constraints.

$$\{[x^j]_\ell\}^k \leq [x^{jk}]_\ell, \quad k = 2, \ldots, |\delta/j|^-, \text{ for } j = 1, \ldots, |\delta/2|^-. \tag{6.26}$$

Notice that some of the constraints in (6.26) are implied. For example, $x^4 \leq [x^4]_\ell$ is implied by $x^2 \leq [x^2]_\ell$ and $\{[x^2]_\ell\}^2 \leq [x^4]_\ell$. In general, considering all possible convex constraints of the type (6.26) that involve some nonlinear term x^p, $2 \leq p \leq \delta$, the constraint $\{[x^a]_\ell\}^{p/a} \leq [x^p]_\ell$ is implied if this set of constraints include a constraint $\{[x^b]_\ell\}^{p/b} \leq [x^p]_\ell$ such that $b > a$, and a is an exact divisor of b, where a and b are integral factors of p. Note also that (6.26) imposes a lower bounding restriction on individual RLT variables $[x^{jk}]_\ell$, and this appears to be more relevant for the terms x^{jk} having a positive coefficient in the objective function. Based on our experience, we recommend using this criteria to generate selected constraints from (6.26). Generating the remaining constraints in (6.26), in addition to such selected ones, usually yields only a marginal improvement, if any, in the resulting lower bound.

For Problem 16, we first solved the regular RLT linear bounding problem described in the previous chapter, which is based on only linearized bound-factor product RLT constraints. Since the objective function is of order 6, we consider all possible distinct sixth order products of bound-factors $(1-x) \geq 0$, and $x \geq 0$. This yields 7 RLT constraints. The resulting linear lower bounding problem has the optimal objective function value of -17,385. Next, we also included the nonlinear convex constraints $x^2 \leq [x^2]_\ell$, $[x^2]_\ell^2 \leq [x^4]_\ell$, and $[x^3]_\ell^2 \leq [x^6]_\ell$ in the above bounding problem. The resulting *convex RLT* bounding problem returned an improved lower bound of -15,202. Including the other possible constraints of type (6.26) in the bounding problem did not further improve this lower bound.

6.2.2 Convex-Concave Partitioning of the Objective Function

This bounding approach is based on partitioning the objective function into convex and concave parts to formulate the problem as a DC-programming problem (Horst and Tuy, 1993). An additional variable is then introduced in the problem to represent the convex part of the objective function. This enables us to derive linear supporting constraints for the epigraph of the convex component, and these in turn are subsequently used as constraint-factors to generate new RLT constraints.

Let $f(x) = g(x) - f_v(x)$ be a decomposition of the objective function such that $g(x)$ and $f_v(x)$ are convex over $0 \leq x \leq 1$. Notice that this partitioning is not unique. In the particular partitioning scheme we adopt, terms of $f(x)$ that have positive coefficients are gathered to define $g(x)$ in addition to the affine term, and the remaining terms collectively define $-f_v(x)$. Having the DC-programming representation **P1**: $\min\{g(x) - f_v(x): 0 \leq x \leq 1\}$, we define a new variable to represent $g(x)$, and write the equivalent problem as follows:

P2: Minimize $\quad z - f_v(x) \qquad (6.27.0)$

subject to $\quad g(x) - z \equiv G(x,z) \leq 0 \qquad (6.27.1)$

$\qquad 0 \leq x \leq 1, \quad z_l \leq z \leq z_u \qquad (6.27.2)$

where the bounds (z_l, z_u) on z can be computed via the bounds on x using the relationship $z = g(x)$, since we know that this must hold true in P2. (Note, in particular, that $z \leq z_u$ is not implied by (6.27.1).) Next, let us select a set of grid points $\{x_1, x_2,..., x_p\}$ over the interval $0 \leq x \leq 1$, along with the corresponding values $\{z_1, z_2,..., z_p\}$, where $z_i = g(x_i)$, so that $G(x_i, z_i) = 0$, $\forall\, i = 1,..., p$. By the convexity of $G(x, z)$, we have,

$$\begin{aligned} 0 \geq G(x, z) &\geq G(x_i, z_i) + (x - x_i, z - z_i)^t \nabla G(x_i, z_i) \\ &= (x - x_i)g'(x_i) - (z - z_i) \qquad \forall\, i = 1,.., p. \end{aligned} \tag{6.28}$$

We can now obtain a bounding problem by generating sixth-order RLT constraints by taking products of constraint-factors from (6.28) along with the bound-factors (6.27.2) six at a time. Including these RLT constraints in (6.27), and then linearizing the resulting representation via the usual substitution process yields such a lower bounding problem.

For Problem 16, selecting three uniform grid points (0.25, 0.5, 0.75), we generated a total of 924 RLT constraints, and the resulting lower bounding problem yielded -7269.633 as its optimal solution, an improvement from the previous lower bound of -15,202. Using alternative grid points (0.2, 0.6, 0.8), (0.2, 0.4, 0.8) and (0.2, 0.5, 0.8), we obtained respective lower bounds of -6182.095, -6182.095 and

$$f''(x) \geq f''_{+}(a_{i-1}) + f''_{-}(a_i) \quad \text{for} \quad a_{i-1} \leq x \leq a_i, \quad \forall\, i = 1, \ldots, q. \tag{6.32}$$

Then, taking the minimum of the right-hand sides of the inequalities in (6.32), we obtain a lower bound on $f''(x)$ over $0 \leq x \leq 1$ as follows.

$$f''(x) \geq f''_{\min} = \min_{i=1,\ldots,q} \{f''_{+}(a_{i-1}) + f''_{-}(a_i)\} \tag{6.33}$$

A similar constraint can be derived for the fourth-order mean value theorem as follows.

$$f(x) \geq f(\bar{x}) + (x - \bar{x})f'(\bar{x}) + \frac{(x - \bar{x})^2}{2} f''(\bar{x}) + \frac{(x - \bar{x})^3}{3!} f^{(iii)}(\bar{x}) + \frac{(x - \bar{x})^4}{4!} f^{(iv)}_{\min} \tag{6.34}$$

The constraints (6.30) and (6.34) are generated using a discretized grid of $\bar{x}$ values, and are subsequently linearized and included in the convex RLT lower bounding problem of Section 6.2.1.

For Problem 16, we computed $f''_{\min} = -373061$ by taking $q = 300$, and we obtained $f^{(iv)}_{\min} = -3{,}600{,}000$ by simply minimizing the quadratic function $f^{(iv)}(x)$ over $0 \leq x \leq 1$. We first considered 11 uniform grid points (including 0 and 1) for the values of $\bar{x}$ to generate a total of 22 linearized constraints of the type (6.30) and (6.34). The resulting augmented convex RLT lower bounding problem yielded an optimal value of -5105.094, an improvement from the previous best value of -6182. This optimal value did not change when we increased the number of grid points to 60. Considering 120 grid points, but this time generating constraints using only (6.34), we obtained only a marginally improved lower bound of -5104.903.

6.2.4 Squared Grid Point Factors

In this constraint generation scheme, given the grid points $\{x_1, x_2, \ldots, x_p\}$ over $0 < x < 1$, we construct products of all possible distinct combinations of squared grid-factors $(x - x_i)^2 \geq 0$, $i = 1, \ldots, p$, $|\delta/2|^-$ at a time. If the degree δ of the objective function is odd, then we multiply the resulting constraints separately by the bound-factors $x \geq 0$ and $(1 - x) \geq 0$. These additional constraints are subsequently linearized to be included in the convex RLT bounding problem of Section 6.2.1.

For example, for Problem 16, which is of degree $\delta = 6$, taking the products of squared grid point factors three at a time, the following constraints are generated.

$$[(x - x_i)(x - x_j)(x - x_k)]_\ell^2 \geq 0 \qquad \forall \ 1 \leq i \leq j \leq k \leq p \tag{6.35}$$

For a choice of 9 uniform grid points given by $\{0.1, 0.2, \ldots, 0.9\}$, we included the resulting 124 constraints of the type (6.35) in the overall bounding problem of Section 6.2.1. The optimal solution of this bounding problem gave a value of 7, which happens to be the global optimum for Problem 16. However, our additional computational experiments revealed that the strength of the constraints (6.35) strongly depends on the choice of the grid points, as might be expected. Solving the bounding problems separately using 3, 5 and 7 uniform grid points over $0 < x < 1$, yielded solutions of objective values -82.71, -23.33, and -61.68, respectively. In spite of the increase in the number of grid points from 5 to 7, the worsening of the lower bound from -23.33 to -61.68 shows the sensitivity of this bounding scheme to the particular choice of grid points.

6.2.5 Squared Lagrangian Interpolation Polynomial Based Constraints

This class of constraints are based on approximating the objective function by using Lagrange's interpolation polynomials over some grid points, and then squaring these approximations to obtain valid implied constraints, which are subsequently linearized. If δ is even, then we generate Lagrangian interpolation polynomials of order $\delta/2$. Otherwise, the squared Lagrangian interpolation polynomials of order $(\delta - 1)/2$ are further multiplied separately by each of the bound-factors $x \geq 0$ and $(1 - x) \geq 0$. For $\delta = 6$, and considering grid points $\{x_1, x_2, \ldots, x_p\}$, these constraints are characterized as follows:

$$\left[f(x_i)\frac{(x-x_j)(x-x_k)(x-x_l)}{(x_i-x_j)(x_i-x_k)(x_i-x_l)} + f(x_j)\frac{(x-x_i)(x-x_k)(x-x_l)}{(x_j-x_i)(x_j-x_k)(x_j-x_l)} + f(x_k)\frac{(x-x_i)(x-x_j)(x-x_l)}{(x_k-x_i)(x_k-x_j)(x_k-x_l)} \right.$$

$$\left. + f(x_l)\frac{(x-x_i)(x-x_j)(x-x_k)}{(x_l-x_i)(x_l-x_j)(x_l-x_k)} \right]_{\ell}^{2} \geq 0 \quad \forall\ 1 \leq i < j < k < l \leq p. \tag{6.36}$$

For Problem 16, using the 9 uniform grid points $\{0.1, 0.2, \ldots, 0.9\}$, 126 constraints of the type (6.36) were included in the convex RLT bounding problem of Section 6.2.1. The resulting problem yielded the optimal value of 6.935, which is very close to the actual global minimum value of 7. Considering 5, 7, and 8 uniform grid points separately, 5, 35, and 70 constraints, respectively, of the type (6.36) were generated. The resulting lower bounding problems yielded the optimal values of -6541.467, -280.25, and -142.870, respectively. According to these results, we expect this bounding scheme to be more stable with respect to the choice of grid points compared to the squared grid point factor based bounding scheme of the previous section.

6.2.6 *Exponential Transformation*

Maranas and Floudas (1994) recently suggested an exponential transformation procedure to solve generalized geometric programming problems, where the exponents appearing in the problem are allowed to be noninteger. To apply this method, the original problem is scaled so that the feasible interval of the scaled variable becomes $1 \le y' \le e^{\beta}$, where $\beta > 0$. Then, using the exponential transformation $y' = e^{t}$, the scaled problem is written in terms of variable t as $\min\{f^{e}(t) \equiv \alpha_0 + \sum_{i=1}^{\delta} \alpha_i e^{it}: 0 \le t \le \beta\}$. According to the sign of the coefficients α_i, $i = 1, \ldots, \delta$, the transformed objective function $f^{e}(t)$ is partitioned into convex and concave parts. Subsequently, for each term $\alpha_i e^{it}$ such that $\alpha_i < 0$, a linear underestimator of $-e^{it}$ that coincides with $-e^{it}$ at the end points of the interval $0 \le t \le \beta$ is constructed. The convex portion of the objective function, that is the collection of terms for which the α-coefficients are nonnegative, is left as it is in exponential form. The resulting convex bounding problem is not invariant to scaling, and the tightness of the approximation depends on the choice of the value of β. Maranas and Floudas (1993) recommend using $\beta < 0.2$, based on the largest magnitude of the error in approximating $-e^{t}$. However, in our computational experiments below, we have obtained better lower bounds for Problem 16 using larger values of β. This is due to the increase in the magnitude of the α-coefficients as β gets smaller, which is not satisfactorily compensated by the tighter linear approximations of $-e^{it}$ as β gets smaller.

At least as good approximations of $-e^{it}$ can be constructed using quadratic underestimating functions, and then applying the first-level RLT to obtain a convex lower bounding problem. Applying this quadratic approximation scheme, we solved convex lower bounding problems for Problem 16, using different values of the scaling parameter β. For β equal to 1/6, 3 and 5, we obtained the lower bounds $-5.50946(10^{10})$, $-5.51968(10^{6})$ and $-4.39422(10^{6})$, respectively. Although the lower bound improves as β increases, we cannot continue to increase β as it causes numerical problems; for example, setting $\beta = 10$, the magnitudes of some of the α-coefficients become less than 10^{-20}, which are indistinguishable from zero in many software packages. As a final note, although this bounding scheme is directly applicable to multidimensional problems, its performance is not competitive compared to any of the lower bounding schemes of the earlier sections.

6.2.7 Computational Experience

In this section, we compare the performance of the foregoing bounding schemes using 7 problems from the literature (see Appendix A), having objective functions of degree δ varying from 3 to 6. All the bounding problems are convex programming problems, and they are solved using GAMS along with the solver MINOS 5.2. The resulting lower bounds obtained via these initial bounding problems are presented in Table 7.

In the convex RLT bounding problem, we used selected constraints from (6.26) as recommended in Section 6.2.1. For the convex-concave partitioning based bounding scheme of Section 6.2.2, we calculated z_l by solving the problem $\min\{g(x)\colon 0 \le x \le 1\}$ using GAMS/MINOS 5.2. Furthermore, z_u was set to $\max\{g(0), g(1)\}$, since an optimal solution of maximizing a convex function over a nonempty polytope occurs at an extreme point of the feasible region. We used three grid points, namely, $\{0.2, 0.6, 0.8\}$, for all the problems in Table 7, and consequently generated 924 constraints for $\delta = 6$, 462 constraints for $\delta = 5$, 210 constraints for $\delta = 4$, and 84 constraints for $\delta = 3$.

Table 7. Comparison of Bounding Schemes for One-Dimensional Polynomial Programming Problems.

Problem	Known Global Optimum	(1) Convex RLT	(2) Convex-Concave Partition	(3) Mean Value Theorem Cuts
12 ($\delta = 3$)	-4.5	-5.625	-4.680	-5.000
13 ($\delta = 4$)	0	-752.778	-212.288	-29.984
14 ($\delta = 4$)	-7.5	-715.278	-216.995	-72.483
15 ($\delta = 5$)	-443.67	-1648.603	-534.445	-822.296
16 ($\delta = 6$)	7	-15202.355	-6182.095	-5105.094
17 ($\delta = 6$)	0	-2565.202	-1271.991	-376.258
18 ($\delta = 6$)	-29763.233	-34598.489	-34338.017	-34519.071

Problem	(4) Squared Grid-Factor Products	(5) Squared Lagrange Interpolation	(6) Exponential Transformation	cpu secs. for (4)
12 ($\delta = 3$)	-4.5	-4.5	-1311.02	0.07
13 ($\delta = 4$)	0	0	-22354.842	0.13
14 ($\delta = 4$)	-7.5	-8.220	-20992.360	0.16
15 ($\delta = 5$)	-457.736	-444.324	-14831.195	0.38
16 ($\delta = 6$)	7	6.964	-5519681.0	0.49
17 ($\delta = 6$)	-0.417	-0.568	-757599.883	0.63
18 ($\delta = 6$)	-30415.791	-34598.482	-1526435.0	0.52

Legend: Problem = Problems from Appendix A having an objective function of degree δ, Columns (1)-(6) = Lower bounds obtained via bounding schemes of Sections 6.2.1-6.2.6, respectively, cpu secs for (4) = cpu seconds required to solve the bounding problem in column (4).

For the mean value theorem based constraints, we used 60 uniform grid points for the values of $\bar{x}$, generating a total of 120 constraints via (6.30) and (6.34). Although not given in Table 7, we also solved bounding problems using 11 uniform grid points, and observed that the resulting lower bounds did not significantly change. The largest change occurred for Problem 15, where the lower bound decreased from -822.296 to -858.057 when the number of grid points was reduced from 60 to 11. Therefore, we recommend using only a limited number (e.g., 10-15) of grid points, for generating mean value theorem based constraints.

For generating both squared grid point factor based and squared Lagrangian interpolation polynomial based constraints, we used 9 uniform grid points, namely $\{0.1, 0.2, \ldots, 0.9\}$. In the first bounding scheme, we generated 125 constraints for $\delta = 6$, 90 constraints for $\delta = 5$, 45 constraints for $\delta = 4$, and 18 constraints for $\delta = 3$. In the second bounding scheme, we generated 126 constraints for $\delta = 6$, 168 constraints for $\delta = 5$, 84 constraints for $\delta = 4$, and 72 constraints for $\delta = 3$. The squared grid point factor based bounding scheme performed better than the squared Lagrangian interpolation polynomial based bounding scheme, except for Problem 15, where the latter performed only slightly better. Aside from Problem 18, both schemes produced comparatively close lower bounds. However, for Problem 18, the squared Lagrangian interpolation polynomial based constraints did not improve the convex RLT bounding problem, even when we tried some different choices of grid points.

The exponential transformation based method performed poorly in comparison to any other bounding scheme, but it has the merit of containing the least number of constraints. However, the objective function has nonlinear (convex) exponential terms. In a specialized branch-and-bound application for solving one-dimensional polynomial programming problems, we recommend using the squared grid factor or the squared Lagrangian interpolation polynomial based RLT constraints, starting with a manageable number of grid points at the root node and generating new ones while dropping some old ones at subsequent nodes based on the solution of the lower bounding problem.

6.3 Multi-Dimensional Polynomial Programming Problems

In this section, we extend our investigation on various bounding strategies to multi-dimensional polynomial programming problems. We use the mathematical statement of the problem PP(Ω) as given in the previous chapter. However, we now scale all the problems for numerical stability purposes so that the simple bound restrictions on each variable is mapped onto the unit interval. That is, we assume that $\Omega \equiv \{x{:}0 \leq x_j \leq 1,\ \text{for}\ j = 1,\ldots, n\}$. The problem parameters that are frequently referred to in this section are the number of variables n, and the maximum polynomial degree δ.

Slightly modifying our presention format, computational experiments and related comments follow the development of each bounding scheme in separate subsections below. Test problems are chosen from the literature, and most of them are application oriented. A complete formulation of these problems are also collectively presented in Appendix A, since at times, the particular structure of some problems becomes the subject of our discussion. All the bounding schemes we discuss yield either linear or convex programming problems that are solved using GAMS/MINOS 5.2 on an IBM 3090 mainframe computer.

6.3.1 Regular and Convex RLT Bounding Problems

The first type of bounding problem is obtained by a direct application of the RLT scheme proposed in the previous chapter. Although this regular RLT bounding problem is weaker than the forthcoming bounding schemes, it is still sufficient to be used in a convergent branch-and-bound algorithm along with appropriate branching rules. Recall that this method constructs a linear programming bounding problem by including the products of bound-factors $x_i \geq 0$, and $(1 - x_i) \geq 0$, $i = 1,\dots, n$, taken δ at a time, to the original polynomial programming problem, and subsequently linearizing the nonlinear terms in the augmented problem by defining a new variable for each distinct nonlinear term. The number of newly generated constraints is given by $\binom{2n + (\delta - 1)}{\delta}$.

The regular RLT bounding problem can be augmented by using simple convex constraints to further relate certain RLT variables with their corresponding nonlinear terms. This scheme is a direct extension of the convex RLT bounding problem that was used for solving one-dimensional polynomial programming problems in Section 6.2.1. The following convex constraints, generated separately for each variable, are simply the constraints in (6.26):

$$\{[x_i^j]_\ell\}^k \leq [x_i^{jk}]_\ell \quad \text{for} \quad k = 2,\dots, \lfloor \delta/j \rfloor, \quad j = 1,\dots, \lfloor \delta/2 \rfloor, \text{ for each } i = 1,\dots, n. \tag{6.37}$$

Table 8 presents the lower bounds obtained by solving the foregoing regular and convex RLT bounding problems, as well as the associated computational effort required and the number of generated constraints. For problems QPP1 and PP2, the regular RLT scheme yields lower bounds within as low as 0.44% and 8% of the actual global minimum, respectively; however, the bounds for the remaining problems, though reasonable, are not as satisfactory. The convex RLT scheme

Table 8. Regular and Convex RLT Bounding Schemes for Multi-Dimensional Polynomial Programming Problems.

Prob.	n	δ	Known Global Optimum	Regular RLT lower bound	number of RLT constrs.	cpu secs.
PP1	2	4	-16.7389	-28.5	35	0.06
PP2	4	4	17.014173	15.5909	330	1.63
PP3	3	7	-5.684802	-27.4742	792	9.13
QPP1	5	2	-30665.539	-30802.756	55	0.07
QPP2	8	2	7049.25	2533.10	136	0.12
QPP4	9	2	-400.0	-500.0	171	0.24

Prob.	Convex RLT lower bound	number of convex constrs.	cpu secs.
PP1	-16.7389	4	0.17
PP2	15.5909	12	2.66
PP3	-27.4742	18	40.55
QPP1	-30765.086	5	0.10
QPP2	2533.20	8	0.16
QPP4	-500.0	9	0.25

Legend: Prob. = Problems from Appendix A having n variables and a maximum polynomial degree of δ, constrs. = constraints generated, cpu secs. = cpu seconds required to solve the corresponding bounding problem, using GAMS/MINOS 5.2 on an IMB 3090 computer.

improves on the lower bound for PP1, closing the gap from 70% of optimality to an exact global optimum. For problem QPP1, the lower bound is also improved from 0.44% to within 0.33% of the actual global optimum. Further improvement of the lower bounds for these two problems will not be pursued in the following subsections. In terms of the computational effort, the degree of the problem δ plays a more decisive role than the number of variables n. For example, three problems having $\delta = 2$ and with n ranging from 5 to 8, are solved well below 0.5 cpu seconds, while the problem PP3, having $\delta = 7$ and $n = 3$, consumed 41 cpu seconds. This outcome is expected, examining the number of RLT constraints generated for these problems. In the forthcoming bounding schemes, after improving the lower bounds, we also suggest and test a procedure for reducing the number of RLT constraints. In particular, the sparsity of distinct nonlinear terms in problem PP3 enables us to cut down the number of RLT constraints drastically, without significantly reducing the resulting lower bound.

6.3.2 Using Available Constraint-Factors

In Remark 5 of Chapter 5, we suggested the use of constraint-factors to generate additional RLT constraints. This strategy is restricted to using the constraints having a polynomial degree smaller than δ so that the products of these constraint-factors with other constraint/bound-factors do not yield polynomials of degree exceeding δ. In this subsection, we make use of such constraint-factors, and discuss their implementation for each test problem.

In problem QPP2, having $\delta = 2$, we use 3 linear inequality constraints as constraint-factors. We take products of these constraint-factors with each other and with bound-factors, generating 54 additional RLT constraints to be included in the convex RLT problem. The resulting problem improved the lower bound to 3339.04.

Problem QPP4 is also of degree $\delta = 2$, and has 3 linear equality type constraints. As per Remark 1 of Chapter 3, we take products of these constraint-factors with each variable to generate 27 equality type RLT constraints. However, no improvement in the resulting lower bound is observed. As a reminder to be kept in mind over the next two sections, these 27 RLT constraints imply any valid additional RLT constraint that can be generated using the above 3 available constraint-factors, since they are of the equality type.

In problem PP2, having $\delta = 4$, there is a single quadratic constraint. We take products of this constraint with all distinct second-order bound-factor products to generate 36 RLT constraints of degree 4 before linearization. These additional constraints improve the lower bound to within 1.2% of the actual global optimum, down from 8.3%. Here, because this constraint is an equality, we can generate equivalent, but fewer, number of RLT constraints by taking products of it with all possible quadratic and linear terms. The resulting 14 constraints will be used in the computational experiments of the next chapter.

Problem PP3 has $\delta = 7$ and two constraints of polynomial degree 3 and 4, respectively. Using the corresponding constraint-factors, we generate a total of 189 constraints by taking the following products:

(constraint-factor of degree $3)^2 \times$ (bound-factors)

(constraint-factor of degree 3) $\times$ (bound-factor products of degree 4)

(constraint-factor of degree 4) $\times$ (bound-factor products of degree 3)

(constraint-factor of degree 4) $\times$ (constraint-factor of degree 3).

These constraints are included in the convex RLT problem, and the resulting problem is solved in 38.96 cpu seconds, yielding a lower bound of -5.6848, which matches the actual global minimum. (Recall that the above previous best lower bound was -27.4742.)

The sparsity in the matrix of distinct nonlinear terms in problem PP3 prompts a scheme for eliminating most of the RLT constraints generated above. The signs of the coefficients for all the three nonlinear terms appearing in the problem indicate that the corresponding RLT variables are expected to attain lower values than the nonlinear terms they represent at an optimum to the bounding problem. Therefore, we suggest retaining only those RLT constraints that impose a lower bounding type of restriction on these three RLT variables. For example, we preserve all the constraints of the type $[x_1^2 x_2]_\ell \geq$ (rest of the constraint). In general, arranging all the constraints in "≥ 0" form, we retain those RLT constraints that have a positive coefficient for at least one of the above three nonlinear terms. This scheme retains 25 of the 189 constraint-factor based restrictions, and 107 of the 792 bound-factor product based RLT constraints. The resulting problem is solved in 1.408 cpu seconds without worsening the lower bound of -5.6848. When the convex constraints (6.37) that do not involve any of the nonlinear terms appearing in the problem are also eliminated, the computational effort to solve the resulting linear program is reduced to 0.613 cpu seconds, again yielding the same lower bound that matches the global optimum.

6.3.3 Mean Value Theorem (MVT) Based Constraint-Factors for Quadratic Constraints

The use of constraint-factors in the previous subsection depends on the availability of constraints having a polynomial degree less than δ. In this subsection, for problems having degree $\delta = 2$, we propose a method that enables us to obtain linear constraint-factors from constraints of degree 2, by using the Mean Value Theorem (MVT). This enables us to derive tighter linear relaxations for the given quadratic constraints.

First Order Expansion: Let $f(x) \geq 0$ be a nonlinear constraint of degree $\delta = 2$. Considering some convex set $\hat{Z}$ such that $Z \cap \Omega \subseteq \hat{Z} \subseteq \Omega$, for a given point $\bar{x} \in \hat{Z}$, by the Mean Value Theorem, we have

$$0 \leq f(x) = f(\bar{x}) + (x - \bar{x})^t \nabla f(\hat{x}) \tag{6.38}$$

where $\hat{x} = \lambda x + (1 - \lambda)\bar{x}$ for some $0 < \lambda < 1$. To derive an upper bounding linear function on $f(x)$, we next construct a constant vector $\hat{f}$ such that $(x - \bar{x})^t \hat{f} \geq (x - \bar{x})^t \nabla f(\hat{x})$ $\forall x \in \hat{Z}$. We first partition the index set $\{1, \ldots, n\}$ of variables as I_N and I_L such that the variables x_i, $\forall i \in I_N$, are those that appear in some nonlinear term of $f(x)$, and x_i, $\forall i \in I_L$, are the remaining variables that appear linearly in $f(x)$. Noting that $[\nabla f(\cdot)]_i$ is constant for $i \in I_L$, we set $\hat{f}_i = [\nabla f(\hat{x})]_i$, for all $i \in I_L$. In order to manipulate the signs of the components of $(x - \bar{x})$ over $x \in \Omega$, we fix $\bar{x}$ at some corner point of the hypercube $\Omega = \{x : 0 \leq x \leq 1\}$. Without loss of generality, we set $\bar{x}_i = 0$ for all $i \in I_L$. For each $i \in I_N$, we minimize or maximize $[\nabla f(\hat{x})]_i$ over $\hat{x} \in \hat{Z}$ according to the value of $\bar{x}_i$ as follows:

$$\hat{f}_i = \begin{cases} \max\{[\nabla f(x)]_i : x \in \hat{Z}\} & \text{if } \bar{x}_i = 0 \\ \min\{[\nabla f(x)]_i : x \in \hat{Z}\} & \text{if } \bar{x}_i = 1. \end{cases} \tag{6.39}$$

Noting that $(x - 1) \leq 0$ and $(x - 0) \geq 0$, $\forall x \in \Omega$, from (6.39), we have $(x - \bar{x})^t \hat{f} \geq (x - \bar{x})^t \nabla f(\hat{x})$ for all $x \in \hat{Z}$. Hence, this yields the following *MVT-cut* from (6.38).

$$f(\bar{x}) + (x - \bar{x})^t \hat{f} \geq 0 \tag{6.40}$$

As far as selecting the fixed values $\bar{x}_i$, $\forall i \in I_N$, is concerned, we consider all possible distinct combinations of 0's and 1's, which (possibly) result in a different vector $\hat{f}$ for each combination, yielding $2^{|I_N|}$ MVT-cuts of type (6.40). Of course, this assumes that $|I_N|$ is reasonably small in magnitude. Otherwise, some smaller restricted number of MVT-cuts ought to be generated. The issue of choosing an appropriate $\hat{Z}$ can be resolved by setting $\hat{Z} = \Omega$, but we suggest including an additional relevant linear constraint in $\hat{Z}$ to obtain a tighter set. In this case, the resulting knapsack problems in (6.39) can be solved without too much of an additional computational burden.

Second Order Expansion: We can improve the above MVT-cuts by using a second order Mean Value Theorem expansion of $f(x)$.

$$0 \leq f(x) = f(\bar{x}) + (x - \bar{x})^t \nabla f(\bar{x}) + \frac{1}{2}(x - \bar{x})^t H(x - \bar{x}) \tag{6.41}$$

In a similar fashion as above, to obtain a linear upper bounding function for $f(x)$, we minimize or maximize component-wise the vector $H(x - \bar{x})$ according to the sign of the corresponding component of $(x - \bar{x})^t$. Denoting the i'th row of H by $H_{i\bullet}$, we define the constant vector h by setting $h_i = 0$, $\forall i \in I_L$, and setting

$$h_i = \begin{cases} \max\{H_{i\bullet}(x - \bar{x}) : x \in \hat{Z}\}, & \text{if } \bar{x}_i = 0 \\ \min\{H_{i\bullet}(x - \bar{x}) : x \in \hat{Z}\}, & \text{if } \bar{x}_i = 1 \end{cases} \qquad \forall\ i \in I_N. \tag{6.42}$$

Then, from (6.41) and (6.42), we obtain the following MVT-cut:

$$f(\bar{x}) + (x - \bar{x})^t \nabla f(\bar{x}) + \frac{1}{2}(x - \bar{x})^t h \geq 0. \tag{6.43}$$

For Problem QPP2 that has 3 quadratic constraints, we generated 19 distinct second-order MVT-cuts as additional constraint-factors. By including the resulting 605 RLT constraints generated by taking products of all 22 constraint-factors with each other and with bound-factors within the convex RLT bounding problem of Section 6.3.1, the accompanying lower bound was improved to 4665.93. This is an increase from the value 3339.40 that was obtained in the previous section by using only available constraint-factors. Problem QPP4 contains two inequality and one equality type quadratic constraints. After expressing the equality constraint in terms of two inequality constraints, we generated 20 second-order MVT-cuts, that were subsequently used to generate 570 additional RLT constraints. Notice that, aside from the 27 RLT constraints generated in the previous section using the 3 equality type available constraint-factors, these 3 constraint-factors are not involved in generating any of the above 570 additional RLT constraints. The augmented bounding problem yields a lower bound that matches the actual global minimum, while consuming 2.51 cpu seconds.

6.3.4 Degree Reduction of Constraints

In this section, we present another method to derive constraint-factors from constraints of degree δ by reducing their degree by one. We first consider the case $\delta = 2$, and then generalize this idea for any $\delta > 2$.

Consider a quadratic term $x_i x_j$, $1 \leq i \leq j \leq n$. The bound-factor products (in nonlinear form) that involve this term $x_i x_j$ produce the following implied constraints:

$$(x_i - l_i)(u_j - x_j) \geq 0 \quad \Rightarrow \quad x_ix_j \leq u_jx_i + l_ix_j - u_jl_i \tag{6.44.1}$$

$$(u_i - x_i)(x_j - l_j) \geq 0 \quad \Rightarrow \quad x_ix_j \leq l_jx_i + u_ix_j - l_ju_i \tag{6.44.2}$$

$$(x_i - l_i)(x_j - l_j) \geq 0 \quad \Rightarrow \quad x_ix_j \geq l_jx_i + l_ix_j - l_jl_i \tag{6.44.3}$$

$$(u_i - x_i)(u_j - x_j) \geq 0 \quad \Rightarrow \quad x_ix_j \geq u_jx_i + u_ix_j - u_ju_i. \tag{6.44.4}$$

(We will refer to the constraints generated via bound-factor products simply as *bound-factor product constraints*, due to its frequent use in section.) Let $\alpha_{ij} \neq 0$ be the coefficient of x_ix_j in some constraint $\phi_r(x) \geq 0$ of degree $\delta = 2$. We eliminate the quadratic term x_ix_j from $\phi_r(x) \geq 0$ as follows.

If $\alpha_{ij} > 0$, then replace x_ix_j by (6.44.1) or (6.44.2), and

if $\alpha_{ij} < 0$, then replace x_ix_j by (6.44.3)or (6.44.4).

The resulting constraint is a relaxation of $\phi_r(x) \geq 0$, and therefore, it is a valid constraint for the given problem. By eliminating all the quadratic terms appearing in $\phi_r(x) \geq 0$ in this manner, we obtain a valid linear inequality, that can now serve as a constraint-factor. Notice that each quadratic term can be replaced by two different linear functions, except for the case $i = j$ where (6.44.1) and (6.44.2) are identical. If $\phi_r(x)$ has q quadratic terms, then the total number of distinct constraint-factors that can be obtained from $\phi_r(x) \geq 0$ is upper bounded by 2^q.

This procedure can also be viewed as a surrogating scheme where $\phi_r(x) \geq 0$ and q appropriately chosen bound-factor product constraints are surrogated in order to eliminate all the quadratic terms using the absolute value of the quadratic term coefficients in $\phi_r(x)$ as weights on the corresponding bound-factor product constraints. This perspective helps us to generalize the above method for $\delta \geq 2$.

Let $\phi_r(x) \geq 0$ be a constraint of degree $\delta \geq 2$ that contains q terms of degree δ. Observe that by construction, any bound-factor product constraint of degree δ

contains exactly one term of degree δ. We can now surrogate the constraint $\phi_r(x) \geq 0$ with q appropriate bound-factor product constraints of degree δ in order to eliminate the terms of degree δ. Each such term is replaced by a function of degree $\delta - 1$, to obtain a valid relaxed version of the original constraint having a degree $\delta - 1$. This can now be used as a constraint-factor. Of course, the number of distinct constraint-factors that can be obtained in this fashion increases rapidly as δ increases. On the average, this number is upper bounded by $2^{q(\delta - 1)}$. However, we first need to assess the effectiveness of using these constraint-factors in tightening RLT relaxations. For this purpose, we applied this method on our remaining two test problems, both of which happen to be of degree two.

For Problem QPP2 having 3 quadratic constraints, we obtained 10 degree-reduced constraint-factors. Along with the 3 original linear constraints, taking products of the total 13 constraint-factors with each other and with the bound-factors, we generated 299 RLT constraints to be included in the convex RLT bounding problem of Section 6.3.1. The augmented problem yielded a lower bound of 4214.65 in 3.06 cpu seconds. This result improves the lower bound of 3339.04 determined in Section 6.3.2; however, it is weaker than the previous section's lower bound of 4665.93. Problem QPP4 has 2 inequality type and 1 equality type quadratic constraints. After expressing its equality type quadratic constraint in terms of 2 equivalent inequality constraints, we obtained 12 constraint-factors using this degree reduction method. A total of 294 RLT constraints were generated using these constraint-factors, and these were included in the convex RLT bounding problem of Section 6.3.1, along with the 27 RLT constraints generated in Section 6.3.2. The optimal solution value of the augmented problem matched the actual global optimum of value -400.0, while consuming 2.21 cpu seconds.

6.3.5 Conclusions

In Section 6.3, we have designed and compared various strategies to strengthen the RLT relaxation representation for multi-dimensional polynomial programming problems. Related results on test problems can be collectively found in Tables 8 and 9.

For the case of quadratic polynomial programming problems, MVT-cut based RLT constraints produce the strongest lower bounds. Competing lower bounds are obtained using degree-reduction based RLT constraints. This method is also applicable for higher degree polynomial programming problems. Both these methods allow constraints of degree δ to contribute to the RLT constraint generation scheme, which were otherwise not used. Hence, they produce a tighter resulting relaxation. However, although computational experiments confirm that these methods tighten the RLT relaxation considerably, it is not practical to generate all possible constraint-factors using these methods. A plausible way of applying these methods might be to generate selected constraint-factors at each node of the branch-and-bound tree based on the incumbent solution and/or the solution of the lower bounding problem associated with its parent node. Suggestions on such applications are forthcoming in Section 8.1. As an alternative strategy to reduce the constraint-factors obtained using the degree-reduction method, we can maximize a given linearized constraint of degree δ subject to a set of appropriately chosen linearized bound-factor constraints along with the simple bound restrictions on the original variables. Then, the terms of degree δ can be eliminated using the resulting optimal dual solution to surrogate this constraint with the bound-factor constraints. When we applied this method for prob-

Table 9. Using Available Constraint-Factors, Mean Value Theorem Cuts and Degree Reduction in the RLT Context.

Prob.	n	δ	Known Global Optimum	L.B. using available constraint-factors	no. of additional RLT constraints	cpu secs.
PP2	4	4	17.014173	16.8042	36	4.66
PP3	3	7	-5.684802	-5.6848	189	38.96
QPP2	8	2	7049.25	3339.40	54	0.37
QPP4	9	2	-400.0	-500.0	27	0.32

Prob.	L.B. using MVT-cuts	no. of additional RLT constraints	cpu secs.
QPP2	4665.93	605	9.56
QPP4	-400.0	597	2.51

Prob.	L.B. using degree-reduced constraint-factors	no. of additional RLT constraints	cpu secs.
QPP2	4214.65	299	3.06
QPP4	-400.0	321	2.21

Legend: Prob. = Problems from Appendix A having n variables and a maximum polynomial degree of δ, L.B. = lower bound obtained using the corresponding method, no. of additional RLT constraints = number of additional RLT constraints generated by the corresponding method to be included in convex RLT bounding problem, cpu secs. = cpu seconds required to solve the corresponding bounding problem using GAMS/MINOS 5.2 on an IMB 3090 computer.

lems QPP2 and QPP4, we obtained the lower bounds of 3416.78 and -400.0 within 0.864 and 0.569 cpu seconds, respectively.

At this stage, it is worthwhile to test the performance of the branch-and-bound algorithm using the next best lower bounding problem, which uses only the constraint-factors that are available in the original problem as described in Section 6.3.2. This lower bounding problem itself appears to be stronger than the exponential transformation based relaxation of Maranas and Floudas (1994). For example, for problem QPP2, the lower bound derived by Maranas and Floudas' method is 2100.00 (for this problem, variables are mapped onto $1 \leq x \leq 2$), as compared with the value 3339.40 obtained by our scheme. Our motivation is to deal with the efficiency issues regarding the implementation of the overall branch-and-bound algorithm, that would remain relevant even when tighter lower bounding problems are used. Such issues include a formal design of a constraint selection strategy, and an investigation of alternative branching variable selection methods. These topics are discussed in the following chapter.

Chapter VII

Implementation of a Polynomial Programming Algorithm

In this chapter, we implement the branch-and-bound algorithm of Chapter 5 for solving multi-dimensional Polynomial Programming Problems, following the development of the RLT-based lower bounding problems in the previous chapter. In constructing the lower bounding problem, we used available constraint-factors as described in Section 6.2.2. The results of this implementation will also form a basis of comparison for the performance of the algorithm when lower bounding problems of Sections 6.3.3 and 6.3.4 are used. To improve the performance of the algorithm, we address various implementation issues. Based on our experience in Chapter 4, we propose alternative branching variable selection rules. Furthermore, the computational experiments of Chapter 6 help us to develop a constraint selection strategy in order to reduce the size of the bounding problems. A range reduction strategy, a heuristic to find feasible solutions, and a definition of optimality and feasibility criteria, all complement the discussed implementation issues. Finally, computational experience using application oriented test problems from the literature closes this chapter.

7.1 Alternative Branching Variable Selection Rules

In this subsection, based on our experience in solving quadratic programming problems in Chapter 4, we propose a branching variable selection scheme to potentially improve the computational efficiency of the overall branch-and-bound algorithm. To preserve the convergence property of the algorithm, the underlying purpose of any branching variable selection rule should be to close the gap between the values of RLT variables and their corresponding nonlinear terms at the optimality of the bounding problem.

The branching variable selection rule (5.11) of Chapter 5 considers all possible distinct nonlinear terms of degree upto and including δ. By not making any *a priori* distinction between nonlinear terms, for example on the basis of whether or not they appear in the original problem, we might possibly waste effort on closing the above mentioned gap for irrelevant nonlinear terms. In Remark 7 of Chapter 5, we improved this rule to a certain extent by eliminating some of the irrelevant nonlinear terms from consideration. To further improve this rule, besides concentrating on the terms that appear in the problem, we now take into account the relative importance of the terms within the original problem.

The two alternative branching variable selection methods presented below generalize the criteria (4.37) and (4.38) of Chapter 4. We assume that all the constraints are arranged in "≥ 0" form, expressing equality constraints in terms of two inequality constraints, if necessary. Having the optimal solution $(\overline{X}, \overline{x})$ to the bounding problem, we select the branching variable x_p using one of the following methods.

Branching Variable Selection Method 1: We first determine the RLT variable whose associated gap with its corresponding nonlinear term deteriorates most the tightness of the relaxation, using the following criterion.

$$\begin{aligned} D_J = \text{minimum}&\left\{0, \begin{bmatrix} \alpha_{0t}\left(\overline{X}_{J_{0t}} - \prod_{j \in J_{0t}} \overline{x}_j\right) & \text{if } J = J_{0t} \text{ for some } t \in T_0 \\ 0 & \text{otherwise} \end{bmatrix}\right\} \\ + \sum_{r=1}^{R} \text{minimum}&\left\{0, \begin{bmatrix} -\alpha_{rt}\left(\overline{X}_{J_{rt}} - \prod_{j \in J_{rt}} \overline{x}_j\right) & \text{if } J = J_{rt} \text{ for some } t \in T_r \\ 0 & \text{otherwise} \end{bmatrix}\right\} \end{aligned} \tag{7.1}$$

where $J \in \overline{N}$. By construction $D_J \leq 0$, for all $J \in \overline{N}$; moreover, for any nonlinear term $\prod_{j \in J} \overline{x}_j$ that does not appear in the original problem, we define $D_J \equiv 0$. Therefore, we compute (7.1) only for the terms appearing in the original problem.

The minimization operators in (7.1) penalize the gap between the values of the RLT variables and their corresponding nonlinear terms only in a single direction determined by the sign of their associated coefficients, as opposed to penalizing the absolute value of the gap. To see the rationale behind this, let us examine the terms in the objective function and in the constraints separately.

In the objective function, for some $t \in T_0$,

if $\alpha_{0t} > 0$, then we penalize the event

$$(\overline{X}_{J_{0t}} - \prod_{j \in J_{0t}} \overline{x}_j) < 0,$$

and if $\alpha_{0t} < 0$, then we penalize the event

$$(\bar{X}_{J_{0t}} - \prod_{j \in J_{0t}} \bar{x}_j) > 0,$$

since a gap in the reverse direction increases the objective function value to our benefit by tightening the associated lower bound. Considering a constraint associated with the index number $r \in \{1, \ldots, R\}$, and some nonlinear term $\prod_{j \in J_{rt}} \bar{x}_j$, $t \in T_{rt}$, in this constraint,

if $\alpha_{rt} > 0$, then we penalize the event

$$(\bar{X}_{J_{rt}} - \prod_{j \in J_{rt}} \bar{x}_j) > 0,$$

and if $\alpha_{rt} < 0$, then we penalize the event

$$(\bar{X}_{J_{rt}} - \prod_{j \in J_{rt}} \bar{x}_j) < 0,$$

since a gap in the reverse direction works in favor of tightening the constraint restriction. Consequently, if $\underset{J \in \bar{N}}{\text{minimum}}\, D_J = 0$, then the lower bounding problem solves the original problem.

To continue with the selection of a branching variable, suppose that $D_{\bar{J}} = \underset{J \in \bar{N}}{\text{minimum}}\, D_J < 0$. Candidates for the branching variable are now restricted to the variables that construct the nonlinear term associated with the index set $\bar{J}$. Based on the same motivation as in (7.1), we compute a cumulative penalty for each candidate variable x_k, $k \in \bar{J}$, as follows.

$$\begin{aligned} \theta_k = & \sum_{\substack{t \in T_0 \\ \ni k \in J_{0t}}} \text{minimum} \left\{ 0, \alpha_{0t} \beta_{J_{0t}}^{(k)} \left(\bar{X}_{J_{0t}} - \prod_{j \in J_{0t}} \bar{x}_j \right) \right\} \\ & + \sum_{r=1}^{R} \sum_{\substack{t \in T_r \\ \ni k \in J_{rt}}} \text{minimum} \left\{ 0, -\alpha_{rt} \beta_{J_{rt}}^{(k)} \left(\bar{X}_{J_{rt}} - \prod_{j \in J_{rt}} \bar{x}_j \right) \right\} \end{aligned} \tag{7.2}$$

where $\beta_{J_{rt}}^{(k)}$ denotes the number of times the index k appears in J_{rt}. These weights penalize the variables that introduce a higher degree of nonlinearity in the terms of interest. Finally, the variable that yields the minimum value for θ_k is selected as the branching variable, that is, we branch on x_p, where,

$$p \in \underset{k \in \bar{J}}{\operatorname{argmin}} \{\theta_k\} \tag{7.3}$$

Branching Variable Selection Method 2: This method is a modification of the above method in which we skip the criterion (7.1) to directly compute θ_k in (7.2) for the entire set of variable indices $k \in N$, and subsequently select the branching variable index p over the candidate set N according to,

$$p \in \underset{k \in N}{\operatorname{argmin}} \{\theta_k\}. \tag{7.4}$$

In (7.3) and (7.4), we break ties in favor of the variable that has the largest feasible interval at the current branch-and-bound node.

Partitioning Phase: Having selected the branching variable x_p, we split its current interval $[l_p, u_p]$ at the value $\bar{x}_p$, creating the partitions $[l_p, \bar{x}_p]$ and $[\bar{x}_p, u_p]$.

Scaling for Branching Variable Selection Methods: In our explanation of the role played by the objective function and constraint coefficients used in (7.1) and (7.2), the focus was on the sign of these α-coefficients. In both the above methods, the magnitudes of the objective function and the constraint coefficients help to differentiate among the candidate branching variables, along with the magnitudes of the gaps between the values of RLT variables and their associated nonlinear terms. However, the magnitudes of the α-coefficients can be manipulated by applying different scaling schemes on the original problem, which may result in a selection of different branching variables for an equivalent bounding problem solution. The particular scaling scheme we propose aims to achieve a "fair" comparison of the relative importance of the above mentioned gaps.

Recall that we scale the original problem by projecting the feasible interval of each variable onto the unit interval. This variable (column) scaling dictates that each nonlinear term, and hence each RLT variable, is also bounded over the unit interval, consequently making the RLT variables to be commensurate with each other. To obtain a compatible basis of comparison across the constraints and the objective function, we then divide each linearized original problem constraint and the linearized objective function by its Euclidean norm.

7.2 Constraint Selection Strategy

In this section, we propose a strategy to reduce the number of constraints in the bounding problem. At the optimal solution of an RLT bounding problem, we observe that only a fraction of all the generated RLT constraints turns out to be useful. This motivates us to try predicting the constraints that might be active at optimality, and eliminate the rest, in a manner that does not significantly deteriorate the resulting lower bound. By using such a reduced bounding problem, the computational effort required to obtain a good quality lower bound will be considerably reduced.

The constraint selection strategy we propose is conducted in two stages. In the first stage, we concentrate on the RLT variables that correspond to the nonlinear terms appearing in the original problem. We select those RLT constraints that impose appropriate restrictions on such RLT variables that serve to oppose the direction in which the definitions $X_J = \prod_{j \in J} x_j$ are more likely to be violated. This is done using the sign of the associated problem coefficients. The selected constraints in the first stage possibly contain some RLT variables that do not correspond to the nonlinear terms appearing in the original problem. Such RLT variables are considered in the second stage, where a second set of RLT constraints are selected to reinforce, in turn, the definition of these RLT variables. We now formalize the constraint selection strategy. (First stage of this strategy was successfully used in Section 6.3.2 of Chapter 6 on example problem PP3.)

Constraint Selection Strategy:

Let us denote the available set of RLT constraints as $f_i(x, X) \geq 0$, $i \in M$. Without loss of generality, we assume that all the RLT constraints as well as the all the original constraints have been arranged in $" \geq 0"$ form.

Stage 1: To remove the distinction between the nonlinear terms appearing in the objective function and those appearing in the constraints, consider the coefficient of a nonlinear term associated with the index set J_{rt}, $t \in T_r$, $r \in \{0, \ldots, R\}$, and define

$$\bar{\alpha}_{rt} \equiv \begin{cases} \alpha_{rt}, & \text{if } r \in \{1, \ldots, R\} \\ -\alpha_{rt}, & \text{if } r = 0. \end{cases} \tag{7.5}$$

- If $\bar{\alpha}_{rt} > 0$, select all the RLT constraints that impose an upper bounding restriction on X_{rt}; that is, select all the RLT constraints in which X_{rt} has a negative coefficient. (Notice that if some RLT constraint is of the form $f_i(x, X) = -X_{rt} + [\text{the rest of the constraint}] \geq 0$, then this constraint imposes an upper bounding restriction on X_{rt} as $X_{rt} \leq [\text{the rest of the constraint}]$.)

- If $\overline{\alpha}_{rt} < 0$, select all the RLT constraints that impose a lower bounding restriction on X_{rt}; that is, select all the RLT constraints in which X_{rt} has a positive coefficient.

Let the RLT constraints selected above be denoted by $f_i(x, X) \geq 0$, $i \in S$, and let the remaining (eliminated) RLT constraints be denoted by the index set $E = M - S$. Then, proceed to Stage 2.

Stage 2: Consider a selected constraint $f_s(x, X) \geq 0$, $s \in S$, and an eliminated constraint $f_e(x, X) \geq 0$, $e \in E$. For any term in f_s having a nonzero coefficient, excluding linear terms (x-variable terms) and the terms appearing in the original problem (which have already been considered in Stage 1 above), if f_e has a nonzero coefficient of opposite sign, then we say that f_e provides a (partial) cover for f_s. Let f_e provide a cover for f_i for all $i \in C_e \subseteq S$. We now select additional constraints from $e \in E$ such that $|C_e| \geq 0.5|S|$, that is, each constraint selected in Stage 2 provides a cover for at least 50% of the constraints selected in Stage 1.

Recall that by construction of the RLT constraints, corresponding to each possible nonlinear term of degree upto and including δ, a separate RLT variable is present in the bounding problem. The above constraint selection procedure, besides reducing the number of constraints, may also reduce the number of RLT variables in the bounding problem. However, the selected constraints in the reduced problem may no longer guarantee the upper and lower bound restrictions on the entire set of variables of this problem, which was guaranteed when the entire set of RLT constraints generated via the linearized bound-factor products of degree δ was present in the bounding problem, due to Proposition 8. Therefore, we now explicitly include the following lower and upper bound restrictions on each RLT variable X_J that is present in the reduced bounding problem,

$$\prod_{j \in J} l_j \leq X_J \leq \prod_{j \in J} u_j \tag{7.6}$$

and also, the bound restrictions on the original problem variables, $l_j \leq x_j \leq u_j$, $j = 1, \ldots, n$. As far as the convex constraints (6.37) are concerned, we generate those that provide a cover for at least one constraint in the reduced problem after applying the above constraint selection strategy.

7.3 Various Implementation Issues

In this section, we describe various other strategies that we have employed in the proposed branch-and-bound algorithm. The branch-and-bound search strategy is a hybrid of the best-first and depth-first strategies that maintains at most *MAXACT* active end nodes in the branch-and-bound tree. The details of this strategy are explained in Section 4.4.2. The strategy used to find feasible solutions, and in particular, the optimality criterion that is used are similar to those for the indefinite quadratic programming problems discussed in Chapter 4. Modifications on these issues are presented next.

7.3.1 Finding Feasible Solutions

As a heuristic to find good quality feasible solutions early in the branch-and-bound algorithm, we simply apply MINOS to the original polynomial programming problem using the *x*-variable part of the optimal solution $(\bar{x}, \bar{X})$ obtained for the lower bounding problem of the current node as the starting solution. This procedure is used at each non-fathomed node of upto level two in the branch-and-bound tree (taking the level of the root node as 0), and whenever $\bar{x}$ is nearly feasible at an unfathomed higher level node.

Another update of the incumbent solution occurs if $\bar{x}$ turns out to be feasible to the original problem constraints. We use additive feasibility tolerances of 10^{-4} for nonlinear constraints, and 10^{-6} for linear constraints to be compatible with MINOS. If $\bar{x}$ is feasible within these feasibility tolerances, then the incumbent solution is updated using the original objective function value $\phi_0(\bar{x})$ evaluated at the solution $\bar{x}$.

7.3.2 Optimality and Fathoming Criteria

The optimality criterion (4.15), that was used for the indefinite quadratic programming problems in Section 4.4.3, is also used in the present algorithm. This criterion guarantees that the final incumbent solution value at the termination of the branch-and-bound algorithm is within $100\varepsilon\%$ of the actual global minimum.

We also fathom a node if its associated lower bounding problem turns out to be infeasible. Recall that each branch-and-bound node is associated with a partition of the original simple bounds on the variables, and that the RLT lower bounding problem is a relaxation of the original problem over the simple bound restrictions associated with the current node. If the lower bounding problem is infeasible over this partition, then the original problem cannot contain any feasible point over the same partition.

Finally, as a reminder, if $\bar{x}$ is feasible (within the above stated tolerances) to the original problem constraints, and if the definition of the RLT variables appearing in the objective function is satisfied by the solution $(\bar{x}, \bar{X})$, then due to the update of the incumbent solution in Section 7.3.1, this node is automatically fathomed by criterion (4.15).

7.4 Range Reduction Strategy

The tightness of the lower and upper bounds on variables plays an important role in determining the strength of the RLT relaxation representation. To improve simple bounds on variables, Maranas and Floudas (1994) have recently suggested to minimize and maximize each variable iteratively using their exponential transformation based relaxation problem prior to starting the branch-and-bound algorithm. In this procedure, they also use an additional constraint that prevents an underestimating function of the objective function from taking values greater than an available upper bound on the global minimum value of the original problem. In a similar procedure, we suggest to adopt the RLT relaxation to improve the bounds on variables.

For this purpose, we can use any of the RLT bounding problems that were designed in the previous chapter, and minimize and maximize each variable sequentially over the corresponding feasible region. Having an upper bound UB on the global optimum of the original polynomial programming problem, the discarded objective function of the bounding problem is now included in the constraints in the following form,

$$[\phi_0(x)]_\ell \leq \text{UB} \tag{7.7}$$

where $[\phi_0(x)]_\ell$ is the linearized objective function of the problem PP(Ω) as before. The resulting problem is called an *RLT range reduction problem.*

The upper bound UB required in (7.7) may be obtained by using some local optimization algorithm or any available heuristic. If no feasible solution to the original problem is available, then UB is set to $+\infty$. However, we suggest to enumerate a few nodes of the proposed branch-and-bound algorithm in order to use the resulting incumbent solution value as UB in (7.7). This approach also permits us to choose to omit the range restriction strategy altogether based on the strength of the lower bounds produced at the enumerated nodes.

In this procedure, whenever a lower or upper bound on some variable improves by solving an RLT range reduction problem, we re-generate the affected RLT constraints using this improved bound to strengthen the RLT relaxation. Upon completing a *cycle* in which all the variables are minimized and then maximized once by solving $2 \times n$ range reduction problems, a new cycle starts until there is no improvement in the bounds on the variables in a complete cycle. For practical purposes, Maranas and Floudas (1994) use the ratio of the volumes of the before-cycle to after-cycle hyperrectangles as a stopping criterion, where a hyperrectangle is defined by the corresponding bound restrictions on the variables. If the volume of the hyperrectangle is not reduced significantly at the end of a cycle, then the procedure stops. Having $l \leq x \leq u$ and $l' \leq x \leq u'$ as the before- and the after-cycle bounds, respectively, this criterion is formally stated as follows:

$$\text{If } r \equiv \frac{\prod_{j=1}^{n}(u'_j - l'_j)}{\prod_{j=1}^{n}(u_j - l_j)} \geq r_{\min} \approx 0.9, \text{ then stop.} \qquad (7.8)$$

In our application, we set $r_{\min} = 0.9$ in (7.8), and used a predetermined limit $cycle_{\max}$ on the number of cycles covered as an additional stopping criterion.

We applied this range reduction strategy on several problems, using the available constraint-factors in constructing the RLT relaxation as suggested in Section 6.3.2, and then applying the constraint selection strategy of the Section 7.2 on the resulting RLT bounding problem (further details on the bounding problems are presented in Section 7.4). The value for UB for each problem is taken as the incumbent solution found at the root node of the proposed branch-and-bound algorithm, using the original simple bounds on the variables. A comparison of the reduced and the starting bounds, presented below, shows the effectiveness of this range reduction strategy.

For QPP2, having $n = 8$, the range reduction procedure required 41 cpu seconds to complete $cycle_{\max} = 10$ cycles. The reduced and the initial bounds, respectively, were obtained as follows, where the ratio in (7.8) was $r = 0.62$ at the end of cycle 10.

$100.000 \leq x_1 \leq 2408.900$	$100.000 \leq x_1 \leq 10000.000$
$1000.000 \leq x_2 \leq 2792.200$	$1000.000 \leq x_2 \leq 10000.000$
$3003.370 \leq x_3 \leq 5752.000$	$1000.000 \leq x_3 \leq 10000.000$
$111.230 \leq x_4 \leq 279.500$	$10.000 \leq x_4 \leq 390.000$
$270.040 \leq x_5 \leq 377.400$	$10.000 \leq x_5 \leq 780.000$
$46.190 \leq x_6 \leq 288.400$	$10.000 \leq x_6 \leq 390.000$
$215.660 \leq x_7 \leq 382.000$	$10.000 \leq x_7 \leq 780.000$
$370.380 \leq x_8 \leq 475.200$	$10.000 \leq x_8 \leq 880.000$

For the same problem, taking up the reduced bounds where Maranas and Floudas (1994) terminated their scheme, we ran our procedure for 5 cycles. This took 21 cpu seconds to complete, with the final volume reduction ratio being $r = 0.42$. The improved and the starting bounds, respectively, were obtained as follows.

$500.000 \le x_1 \le 720.500$	$500.000 \le x_1 \le 1000.000$
$1166.370 \le x_2 \le 1480.800$	$1000.000 \le x_2 \le 1550.000$
$5000.000 \le x_3 \le 5284.102$	$5000.000 \le x_3 \le 5550.000$
$174.720 \le x_4 \le 192.300$	$126.020 \le x_4 \le 231.600$
$288.660 \le x_5 \le 300.300$	$271.350 \le x_5 \le 328.000$
$208.050 \le x_6 \le 225.200$	$107.310 \le x_6 \le 270.500$
$279.460 \le x_7 \le 295.900$	$238.780 \le x_7 \le 343.900$
$388.930 \le x_8 \le 400.200$	$371.530 \le x_8 \le 426.900$

In the above mentioned paper, due to typographic errors in the original problem bounds, we cannot test our procedure starting from those bounds.

For QPP3, having $n = 8$, the range reduction procedure required 36 cpu seconds to reach the limit $cycle_{max} = 10$ with the final volume reduction ratio being $r = 0.62$. The reduced and the initial bounds, respectively, were obtained as follows.

$0.540 \le x_1 \le 1.000$	$0.000 \le x_1 \le 1.000$
$0.350 \le x_2 \le 0.500$	$0.000 \le x_2 \le 1.000$
$0.000 \le x_3 \le 0.300$	$0.000 \le x_3 \le 1.000$
$0.380 \le x_4 \le 0.400$	$0.000 \le x_4 \le 1.000$
$0.050 \le x_5 \le 8.400$	$0.000 \le x_5 \le 16.000$
$1.480 \le x_6 \le 15.100$	$0.000 \le x_6 \le 16.000$
$0.020 \le x_7 \le 2.800$	$0.000 \le x_7 \le 4.000$
$1.160 \le x_8 \le 3.800$	$0.000 \le x_8 \le 4.000$

For QPP5, having $n = 13$, the range reduction procedure required 39 cpu seconds for 6 cycles, at which the stopping criterion (7.8) was satisfied. The reduced and the initial bounds, respectively, were obtained as follows.

$0.804 \le x_1 \le 0.804$	$0.100 \le x_1 \le 1.000$
$0.900 \le x_2 \le 0.900$	$0.100 \le x_2 \le 1.000$
$0.944 \le x_3 \le 0.985$	$0.900 \le x_3 \le 1.000$
$0.100 \le x_4 \le 0.100$	$0.000 \le x_4 \le 0.100$
$0.191 \le x_5 \le 0.191$	$0.100 \le x_5 \le 0.900$
$0.305 \le x_6 \le 0.618$	$0.100 \le x_6 \le 0.900$
$573.776 \le x_7 \le 574.143$	$0.100 \le x_7 \le 1000.000$
$74.009 \le x_8 \le 74.103$	$0.100 \le x_8 \le 1000.000$
$500.005 \le x_9 \le 500.041$	$500.000 \le x_9 \le 1000.000$

$0.100 \leq x_{10} \leq 0.136$	$0.100 \leq x_{10} \leq 500.000$
$20.228 \leq x_{11} \leq 20.241$	$1.000 \leq x_{11} \leq 150.000$
$77.341 \leq x_{12} \leq 77.354$	$0.000 \leq x_{12} \leq 150.000$
$0.004 \leq x_{13} \leq 0.011$	$0.000 \leq x_{13} \leq 150.000$

For PP4, having $n = 10$ and $\delta = 3$, due to the increased number of RLT constraints generated and subsequently selected, the bound improving procedure required 268 cpu seconds for a single cycle, yielding $r = 7 \times 10^{-6}$. The reduced and the initial bounds, respectively, were obtained as follows.

$1254.249 \leq x_1 \leq 1853.555$	$0.000 \leq x_1 \leq 2000.000$
$9067.840 \leq x_2 \leq 16000.000$	$0.000 \leq x_2 \leq 16000.000$
$26.311 \leq x_3 \leq 71.572$	$0.000 \leq x_3 \leq 120.000$
$2577.103 \leq x_4 \leq 3093.998$	$0.000 \leq x_4 \leq 5000.000$
$1780.076 \leq x_5 \leq 2000.000$	$0.000 \leq x_5 \leq 2000.000$
$87.691 \leq x_6 \leq 91.957$	$85.000 \leq x_6 \leq 93.000$
$94.186 \leq x_7 \leq 95.000$	$90.000 \leq x_7 \leq 95.000$
$8.324 \leq x_8 \leq 10.814$	$3.000 \leq x_8 \leq 12.000$
$1.562 \leq x_9 \leq 1.821$	$1.200 \leq x_9 \leq 4.000$
$152.455 \leq x_{10} \leq 153.535$	$145.000 \leq x_{10} \leq 162.000$

In our computational experiments with the overall branch-and-bound algorithm, we used the above improved bounds on variables for the above problems.

7.5 Computational Results

In this section, we evaluate the proposed algorithm using a set of application oriented problems from the literature. We solved four polynomial programming problems PP1-PP4, and five quadratic polynomial programming (quadratically constrained quadratic programming) problems QPP1-QPP5. These problems are included in Appendix A, and are numbered 21-29. As the lower bounding problem, we applied the strategy of Section 6.3.2, which uses available constraint-factors, and then applied the constraint strategy of Section 7.2 to the resulting (full size) bounding problem. The lower bounding problems, as well as the upper bounding problems in the heuristic of Section 7.3.1, are solved using MINOS 5.1 as a subroutine in a Fortran code. For all the test problems, we used the Branching Variable Selection Method 1 of Section 7.1, and set $\varepsilon = 0.01$ in the optimality criterion (4.15), which guarantees a solution within 1% of global optimality at termination. For the problems QPP3-QPP5 and PP4, we used the improved bound restrictions on the variables as obtained by the range reduction strategy of Section 7.4. For the remaining test problems, we simply used the given original bounds on the variables.

The results presented in Table 10 show that all the problems are solved under 36 cpu seconds on an IBM 3090 mainframe. Problems QPP2 and QPP3 were among the most challenging problems as prognosticated by the lower bounds obtained at the root node. These two problems required an enumeration of 155 and 139 branch-and-bound nodes in 35.42 and 30.96 cpu seconds, respectively. When we applied branching variable selection method 2 instead of the method 1, the number of branch-and-bounds enumerated reduced to 145 for problem QPP2 while this measure increased to 143 for problem QPP3. Although this performance is related to just two test problems, there does not appear to be any significant advantage of one strategy over the other. When we omitted the range reduction strategy, and used the original bound restrictions on the variables for these two problems, the algorithm was prematurely terminated after enumerating the preset limit of 200 nodes. However, for problem QPP2, when we used the improved variable ranges that are reported in Maranas and Floudas (1994) without pursuing further improvement, this problem was solved within 6.45 cpu seconds enumerating only 25 nodes.

For all the problems, except PP3, a good quality incumbent solution was found at the root node itself, whose value did not improve by more than 0.5% at the

Table 10. Performance of the Branch-and-Bound Algorithm for Solving Multi-Dimensional Polynomial Programming Problems.

Problem	(n, δ)	Known $v[(Q)PP]$	v[B&B]	cpu secs.	No. of B&B nodes	Node 0 LB
PP1	(2, 4)	-16.7389	-16.7389	0.97	1	-16.7388
PP2	(4, 4)	17.01417	17.01402	15.05	3	16.75834
PP3	(3, 7)	-5.6848	-5.6852	20.30	3	-5.6847
PP4	(10, 3)	-1768.807	-1768.807	21.24	3	-1795.572
QPP1	(5, 2)	-30665.539	-30665.527	1.05	1	-30765.071
QPP2	(8, 2)	7049.25	7049.24	35.42	155	5000.83
QPP3	(8, 2)	-0.3888	-0.3888	30.96	139	-0.4708
QPP4	(9, 2)	-400.00	-400.00	1.99	3	-500.00
QPP5	(13, 2)	97.588	98.588	3.31	1	97.588

Problem	(n, δ)	M	M_R	reduction%	M_{cx}	aver. cpu secs per RLT prob.
PP1	(2, 4)	35	13	63%	3	0.14
PP2	(4, 4)	344	240	30%	12	4.49
PP3	(3, 7)	981	140	86%	11	2.10
PP4	(10, 3)	2723	487	82%	9	4.51
QPP1	(5, 2)	55	29	47%	1	0.18
QPP2	(8, 2)	190	15	92%	0	0.21
QPP3	(8, 2)	153	27	82%	2	0.19
QPP4	(9, 2)	153	27	82%	2	0.19
QPP5	(13, 2)	435	61	86%	3	0.46

Legend: Problem (n, δ) = Problems from Appendix A having n variables and a maximum polynomial degree of δ, v[(Q)PP] = known optimal (best) solution of problem (Q)PP, v[B&B] = branch-and-bound algorithm incumbent value, cpu secs. = cpu seconds to solve the problem on an IBM 3090 computer, No. of B&B nodes = number of branch-and-bound nodes generated, Node 0 LB = lower bound on (Q)PP at root node, M = number of linear RLT constraints in full sized bounding problem M_R = number of linear RLT constraints in reduced bounding problem, reduction% = percentage reduction in the number of constraints, M_{cx} = number of convex constraints used, aver. cpu secs. per RLT prob. = average cpu seconds consumed per bounding problem (optimality criterion is 1%).

remaining nodes. For problem PP3, it required an enumeration of 3 nodes to find the final incumbent solution.

The constraint selection strategy of Section 7.2 reduced the number of constraints in the bounding problem by 30-90% Notice that this procedure is applied only once, and throughout the algorithm, the resulting reduced lower bounding problem is solved. For problem PP3, the full sized RLT problem has 981 linear RLT constraints, and the Stage 2 of the constraint selection procedure consumed 10 cpu seconds, which is around half the total computational effort to solve this problem. For this reason, in the case of problem PP4, for which the full sized bounding problem has 2723 constraints, Stage 2 of the constraint selection strategy was not used.

As the problem size, and especially the degree of the problem, increases, the computational effort required to solve the lower bounding problem at each node becomes the determining factor of the total computational effort required by the proposed branch-and-bound algorithm. For example, on the average, a lower bounding problem is solved in 4.51 cpu seconds for problem PP4. An interesting observation is that to solve almost the same sized problem required around 10 cpu seconds on the average in the range reduction procedure. This effort can be greatly reduced by using Lagrangian relaxation techniques as done for the quadratic programs in Chapter 4.

In this chapter, besides putting the theoretical developments of Chapter 5 into practice, we have also tested various implementation strategies in order to improve the performance of the proposed algorithm. The results presented in Table 10 provide a benchmark for comparing the different constraint generation schemes of Sections 6.3.3-6.3.4. Even at this stage, using our weakest relaxation, the proposed algorithm proved to be very promising. Here, one of our objectives was to evaluate the branch-and-bound ideas that form the basis of the strategies devised in this chapter, so that they might provide guidelines for further improving the algorithm. The computational results also helped to determine the areas where further research is needed. As the results on problem PP4 indicate, a further reduction of the constraints for higher degree problems is necessary. Also, as the results on problem QPP2 indicate, the generation of additional, tighter classes of constraints appear to be necessary. Although these two directions appear to be contradictory, one has to view them in the context of replacing the not so useful constraints with fewer, but stronger, ones. Additional discussion and further suggestions on these issues are presented in Section 8.1 of the closing chapter of this dissertation.

Chapter VIII

Summary, Conclusions, and Future Research

In this dissertation, we have studied the class of polynomial programming problems and its special case, the class of indefinite quadratic programming problems. Polynomial programming problems involve the minimization (or maximization) of a polynomial objective function subject to polynomial constraints. We assume that a bounding hyperrectangle is available for the decision variables. However, there is no convexity assumption either on the objective function or on the feasible region. Polynomial programming problems are an important class of difficult nonconvex problems that arise in a variety of production, location, distribution, and engineering design contexts. Recently, such problems have been attracting increased attention, especially in the chemical engineering design area. These problems have traditionally resisted solution, and this has impeded their use in practice.

To solve polynomial programming problems, we have presented a convergent branch-and-bound algorithm. The crucial element of this algorithm is a technique to find tight lower bounds (considering minimization problems), at each node of the branch-and-bound tree. For this purpose, we have developed a new Reformulation-Linearization-Technique (RLT) for generating tight linear programming relaxations based on an automatic constraint generation scheme. The underlying purpose of this method is to closely approximate the convex envelope of the objective function over the convex hull of the feasible region. In the *reformulation phase* of this method, referring to the simple bound restrictions on the variables as bound-factors, and taking products of these bound-factors with each other, we generate implied nonlinear constraints of polynomial degree equal to the maximum polynomial degree of the given problem. Upon including these constraints in the original problem, in the *linearization phase*, we define a new variable for each *distinct* polynomial term; for example, the term $x_1x_3^2x_4$ is represented by the newly defined RLT variable X_{1334}. The optimal value of the resulting linear programming problem yields a lower bound on the original problem. This follows, because given an optimal solution to the original problem, we can construct a feasible solution for the bounding problem by using the variable definitions in the linearization phase, so that the values of the RLT variables agree with the values of the nonlinear terms they represent. Consequently, all the functions in the linear bounding problem evaluated at this constructed solution give the same value as the corresponding nonlinear functions evaluated at the given optimal solution to the original problem, before the linearization phase. However, the converse is not true; that is, the optimal solution to the linear bounding problem does not necessarily satisfy the RLT variable definitions. The discrepancy between the values of the RLT variables and the nonlinear terms they represent at the optimal solution obtained to the lower bounding problem, forms the basis for designing *suitable* branching decisions that would guarantee the convergence of the algorithm. The objective of the branching variable selection rules is to reduce the maximum of such discrepancies at each node of the branch-and-bound tree. A partitioning of the original problem in the branch-and-bound framework is conducted by splitting the feasible interval of the selected branching variable at the value specified in the solution to the linear programming relaxation.

As a particular class of polynomial programming problems, we have presented a detailed treatment of Indefinite Quadratic Programming Problems (IQP). These problems minimize a nonconvex quadratic objective function subject to linear constraints. Being an NP-complete problem, the main difficulty in solving these problems is that there might exist many local optima, and such local information is inadequate for recognizing global optimality. For these reasons, well-established nonlinear programming methods fail to guarantee a global optimal solution.

Notwithstanding this difficulty, indefinite, and in particular concave quadratic programming problems, have attracted considerable attention in the literature because of the applications in which they arise. Some of these applications include modelling economies of scale in a cost structure, location-allocation problems, some production planning and risk management problems, and various other mathematical models such as the maximum clique problem and the jointly constrained bilinear programming problem.

We have specialized and extended the above branch-and-bound algorithm to solve indefinite quadratic programming problems. The special structure of this problem has been exploited to the extent of not only generating enhanced RLT representations, but also in designing efficient solution procedures for the resulting lower bounding problems, as well as in developing many new features to improve the overall efficiency of the branch-and-bound algorithm.

To obtain a tighter representation for the RLT lower bounding problem, in addition to pairwise bound-factor products, we have proposed the use of products of other constraint-factors with each other, as well as with bound-factors. We call these constraints collectively as RLT constraints. In investigating various properties of the resulting linear lower bounding problem RLT-LP, we have shown that the original linear constraints are implied by the RLT constraints, and that using implied constraints in IQP to generate RLT constraints is not useful. We have also shown that if a linear constraint in IQP is binding at some feasible point, then all of the RLT constraints that are generated by using products of this constraint-factor with other bound/constraint-factors are also binding when the original variables are fixed at this feasible point. Another useful result shows that the optimal objective function value of RLT-LP is invariant with respect to affine transformations applied to the original problem IQP. However, as a particular linear transformation, we considered the eigen-transformation of IQP to obtain a separable objective function. This representation of IQP enabled us to identify RLT constraints that could be deleted without compromising much on the quality of the resulting lower bound.

We have also introduced a special set of sparse, bounding convex constraints in the RLT application to the original problem IQP. This has greatly enhanced the strength of the resulting relaxation. To solve the resulting convex programming lower bounding problem RLT-NLP efficiently, we adopt a suitable Lagrangian dual scheme by exploiting the partial separable structure of some of the RLT constraints. The Lagrangian subproblem solution procedure handles the nonlinear convex constraints without creating any additional computational burden. The Lagrangian dual formulation is further improved by a using layering strategy, which relates some of the RLT constraints that have similar structures by variable duplication. This formulation also allows us to optimize the dual variables corresponding to the variable duplication constraints one at a time, given the remaining dual variable values. The nondifferentiable Lagrangian dual problem LD-RLT-NLP is solved adopting a deflected subgradient method.

Using problems from the literature, we have compared the performance of the foregoing lower bounding schemes. Both RLT-NLP and the RLT application to the eigen-space representation of the original problem along with selected nonlinear convex constraints yield lower bounds very close to the actual minimum of the test problems. In comparison to RLT-NLP, the Lagrangian dual problem LD-RLT-NLP yields somewhat worsened lower bounds, due to the inherent difficulty in solving nondifferentiable problems. However, the computational effort to solve LD-RLT-NLP is significantly reduced compared to that consumed by RLT-NLP.

To further strengthen the bounding problem, we have derived two additional classes of linear constraints based on projecting cubic bound-factor products onto the quadratic space, and on squaring *differences* of bound (or constraint) factors. However, although these constraints serve to somewhat tighten the relaxation, they increase the problem size considerably. Hence, due to the ensuing computational burden, they are not used in the overall branch-and-bound algorithm.

To improve the fathoming power of the algorithm, besides good quality lower bounds, we also need to obtain incumbent solutions (upper bounds) that are close to a global optimal solution at early stages of the algorithm. For this purpose, we have devised several heuristic procedures, including a penalty function based method applied directly on IQP. This method incorporates a projection scheme for near feasible points, and uses MINOS to solve the Lagrangian dual subproblem that corresponds to the incumbent dual solution, after including the original linear functional constraints. These procedures are strongly dependent on the starting point used. For this purpose, we used the solution obtained to the lower bounding problem as the starting point. Additionally, whenever the incumbent value improves, taking this new incumbent point as the initial point, we simply solve the original problem IQP using MINOS.

To further improve the performance of the branch-and-bound algorithm, we developed four different range reduction strategies, which tighten the simple bounds on the variables based on feasibility and various optimality conditions. The first procedure is based on knapsack problems defined by individual linear functional constraints along with the simple bound restrictions on the variables. The second procedure is motivated by the number of constraints that have to be binding at optimality. The third and the fourth procedures are applied to cutting planes based on, respectively, the Lagrangian dual objective function, and the eigen transformed separable objective function.

Other features of the algorithm include the branch-and-bound search strategy, for which we adopt a hybrid depth-first and best-first search strategy as developed by Sherali and Myers (1985). Also, to avoid the enumeration of alternative and close to global optimal solutions, we set an optimality criterion such that when the algorithm terminates, the final incumbent solution is guaranteed to be within a predetermined percentage of the global minimum.

Empirical experiments are conducted using various test problems from the literature. Using an optimality criterion of 5% for six of the more difficult problems that have 20 variables, and 1% for the remaining five smaller sized problems, all the problems are solved under 3.30 cpu seconds, except one, which required 16 cpu seconds. However, for this problem a better solution than the one reported in the literature has been found. Seven larger sized randomly generated problems having upto 50 variables are solved at the root node with a reasonable amount of computational effort using an optimality criterion of 5%. The best incumbent solution for all the problems is found at the root node, except for one problem, where the best solution was found at the second node. This encourages the use of the proposed algorithm as a heuristic procedure for larger sized problems by limiting the number of branch-and-bound nodes enumerated. We recommend exploiting any inherent special structures of a given problem, whenever possible, based on the material presented in Chapter 3, to construct an enhanced RLT bounding problem.

Upon completing our discussion on quadratic programming problems, we have, both theoretically and computationally, investigated various implementation issues regarding the construction of RLT-based bounding problems for polynomial programming problems. In particular, it is shown that RLT applied to a given problem yields tighter bounds than RLT applied to the equivalent "quadrified" problem, where the original problem is transformed to a quadratically constrained quadratic programming problem by successive variable definitions. This scheme was first suggested by Shor (1990a).

Considering one-dimensional polynomial programming problems, we have developed various strategies to improve the RLT lower bounding problem. Compared to a recently proposed exponential transformation based bounding scheme, all the RLT based methods performed substantially better in our computational experiments. Using one of these methods, the resulting lower bounding problem produced the actual global minimum for 4 out of 7 test problems from the literature, and for the remaining problems, the maximum deviation of the lower bounds from the actual global minimum turned out to be only 3%. In contrast, the alternate exponential transformation method in the literature produces bounds as far as 788,526% from the optimum!

For multi-dimensional polynomial programming problems, we also developed additional constraint generation schemes, such as using available constraint-factors, mean value theorem based cuts, and degree reduction of nonlinear constraints. Test problems from the literature are solved to compare the performance of these methods. For 5 out of 6 test problems, lower bounds obtained using the above constraint generation strategies were within 1.2% of the actual global minimum. Using the available constraint-factors to generate additional RLT constraints, we implemented the proposed algorithm. In this implementation, we addressed the issues of alternative branching variable selection rules, constraint reduction strategies, and techniques for finding good quality feasible solutions. We also showed that feasible ranges of variables can be improved using the RLT relaxation. Using test problems from the literature, we conducted computational experiments, and the results confirm the effectiveness of the proposed RLT relaxation based branch-and-bound algorithm.

8.1 Future Research

In this final section, we suggest strategies that have the potential to improve the performance of the proposed algorithm, as well as present directions for future research. We first address the implementation issues that are alluded to at the end of Section 7.5.

In Section 6.3.4, we have proposed a degree reduction method to obtain constraint factors from constraints of degree δ. A possible practical way of using these degree-reduced constraint-factors for generating RLT constraints could be achieved as follows. Consider a nonlinear term $\prod_{j \in J} x_j$ of degree δ that appears in some original problem constraint having the coefficient $\alpha_J \neq 0$. Guided by the solution $(\bar{x}, \bar{X})$ of the lower bounding at some node of the branch-and-bound tree, among candidate bound-factor product constraints that can replace $\prod_{j \in J} x_j$ by a function of degree $\delta - 1$, we can select the one that is closest to being satisfied in linearized form at the solution $(\bar{x}, \bar{X})$. Formally, if $\alpha_J > 0$ (< 0) then among all the bound-factor product constraints of degree δ in which $\prod_{j \in J} x_j$ has a coefficient of -1 $(+1)$, we select the one that, in linearized form, yields the smallest slack value at the solution $(\bar{x}, \bar{X})$. Notice that even when we use the constraint selection strategy of Section 7.2, all such linearized bound-factor products are present in the reduced lower bounding problem by virtue of the selection criteria used in Stage 1 of this strategy. Proceeding as described above, we can eliminate all the terms of degree δ from the given constraint to obtain a $\delta - 1$ degree constraint-factor, and then repeat this procedure for each constraint of degree δ. The resulting constraint-factors can now be used to generate new RLT constraints at the immediate descendant nodes of the current node.

In Sections 6.3.1 and 7.5, we observed that as the degree of the problem increases, the number of generated new RLT constraints increases rapidly. However, for higher degree problems, distinct nonlinear terms appearing in the problem are typically fewer than all possible nonlinear terms of degree upto and including δ. This sparsity of the matrix of nonlinear terms can be incorporated into the RLT constraint generation scheme to reduce the number of RLT constraints. Otherwise, even if a problem of degree δ contains only a single term of degree δ, we still generate bound-factors of degree δ, which bring along irrelevant terms of degree δ. Instead of letting the highest degree term of the problem dictate the degree of the bound-factor products, we can generate customized bound-factor product constraints for each individual nonlinear term appearing in the problem.

Considering two RLT variables X_J and $X_{J'}$ whose corresponding nonlinear terms are of respective degrees δ and δ', where $\delta > \delta'$, we may generate linearized bound-factor products of degree δ that contain X_J, and linearized bound-factor products of degree δ' that contain $X_{J'}$. Moreover, if $J' \subset J$, then we do not need to generate bound-factor product based RLT constraints for $X_{J'}$, since they are implied by those generated for X_J, as per Proposition 8 of Chapter 5. However, when a constraint selection strategy is used, then we need to generate some of the bound-factor product based RLT constraints for $X_{J'}$, if they are not implied by the reduced set of those generated for X_J. An extension of this idea to constraint-factor based RLT constraints might be complicated, as well as computationally burdensome. Notwithstanding this general concern, it might be still practical to extend this method for the RLT constraints generated via products of constraint-factors with bound-factors.

At Stage 2 of the constraint selection strategy proposed in Section 7.2, we require a comparison of each constraint selected at Stage 1 with each eliminated constraint. If there exists a large number of RLT constraints, this strategy may lose its practicality. At Stage 2, we may try to use other criteria instead of the number of covered selected constraints. For example, the total number of *covered terms* within the set of selected constraints can be computed more efficiently for each eliminated constraint. To compute this criterion for an eliminated constraint, we do not have to access each individual selected constraint. The required information on the set of selected constraints can be condensed for each term of interest. If this criterion alone fails to select a satisfactory collection of constraints at Stage 2, then we may still use this criterion to construct a smaller candidate set of eliminated constraints to which the Stage 2 procedure as described in Section 7.2 can be practically applied.

The proposed algorithm can be specialized for solving one-dimensional polynomial programming problems using various bounding problem generation methods devised in Section 6.2. The promising methods of Sections 6.2.4 and 6.2.5 require the selection of a set of grid points. Strategies to generate such grid points need to be investigated. For example, using the solution to the lower bounding problem and/or using points where the derivative of the objective function becomes zero for determining grid points are among possible strategies that can be explored.

Another specialization of the proposed algorithm is for solving quadratic polynomial programming problems. We have proposed a specialized RLT constraint generation scheme that uses mean value theorem cuts in Section 6.3.3. However, considering a nonlinear constraint, a (possibly) distinct MVT-cut can be generated for each corner point of the hyperrectangle formed by the bounds on the variables appearing nonlinearly in this constraint. Therefore, practical applications of this strategy need to be investigated. For example, the corner points where the given constraint is feasible might be discarded. Other strategies may be devised along the lines suggested above for the degree reduction method.

The range reduction strategies devised for quadratic programming in Chapter 4 have considerably improved the performance of the specialized algorithm for solving this class of problems. Similar strategies can also be adopted for polynomial programming problems. For related references, see Ryoo and Sahinidis (1993), Hansen et al. (1991), and Hamed and McCormick (1993).

To supplement the heuristic of Section 7.3 for finding good quality feasible solutions, other fast heuristics can be devised. For example, the penalty function based procedure of Section 4.7 could be generalized for polynomial programming problems, or condensation based generalized geometric programming approaches could be adopted.

The Reformulation-Linearization-Technique has proven to be a powerful tool in generating tight relaxation representations for nonconvex quadratic and polynomial programming problems. Due to its automatic constraint generation property, there are many opportunities to tighten the relaxation representations. Often, new strategies have to be tested computationally to determine their relative merit, and in particular, to devise practical ways of incorporating them in the overall algorithm. Further developments and refinements of some of the strategies alluded to in this section will be the subject of continuing research on this subject.

Finally, the proposed algorithm has the flexibility to be easily extended to handle 0-1 polynomial programming problems, as well as to exploit special structures inherent in particular problems of interest based on the theoretical results we have presented. An immediate specialization of the indefinite quadratic programming implementation is to solve maximum clique problems over large-scale, dense networks. Such problems can be formulated as that of maximizing a bilinear objective function subject to a single equality constraint, along with simple bounds on the variables. This topic will also be pursued in future research.

Appendix A

Test Problems

In this section we provide a complete description of the problems that have been referred to in this research, along with their original sources. For each problem, a global (or the best known) solution is denoted by x^*, and the corresponding objective function value is denoted $f(x^*)$. (Whenever an optimal solution to a given test problem is not heretofore known, this is so indicated.)

PROBLEM 1: BLP1

(Al-Khayyal and Falk (1983): pp. 274)

$$\begin{array}{ll} \text{Minimize} & -x + xy - y \\ \text{subject to} & -6x + 8y \leq 3 \\ & 3x - y \leq 3 \\ & 0 \leq x,\ y \leq 5 \end{array}$$

Global Solution: $(x^*, y^*) = (7/6,\ 1/2)$, $f(x^*, y^*) = -1.0833$.

PROBLEM 2: BLP2

(Al-Khayyal and Falk (1983): Ex. 1)

Minimize $(2,-4,8,4,9)x + x^t y + (3,-1,-2,-4,5)y$
subject to $B^1 x \le b^1, \quad B^2 y \le b^2$
$0 \le x_i \le 20, \quad 0 \le y_i \le 20$

where

$b^1 = (1,3,5,4,0)^t, \quad b^2 = (0,7,3,6,2)^t$

$$B^1 = \begin{bmatrix} -8 & 0 & -6 & 7 & -7 \\ -6 & 2 & -3 & 9 & -3 \\ 6 & 0 & -7 & -8 & 2 \\ -1 & 1 & -8 & -7 & -5 \\ 4 & -7 & 4 & 5 & 1 \end{bmatrix}, \quad B^2 = \begin{bmatrix} 0 & 5 & -4 & 9 & -7 \\ 7 & 4 & 3 & 7 & 5 \\ 6 & 1 & -8 & 8 & 0 \\ -3 & 2 & 7 & 0 & 1 \\ -2 & -3 & 8 & 5 & -2 \end{bmatrix}$$

Global Solution: $x^* = (4.5667, 20, 3.2, 0, 0)$, $y^* = (0, 0, 0.19565, 0.086957, 0)$, $f(x^*, y^*) = -45.38$.

PROBLEM 3: BLP3

(Al-Khayyal and Falk (1983): Ex. 2)

Minimize $(-1,-2,-3,-4,-5)x + x^t y + (-1,-2,-3,-4,-5)y$
subject to $B\begin{pmatrix} x \\ y \end{pmatrix} \le b$
$x_1 + x_2 + x_3 - 2x_4 + x_5 + y_1 + y_2 + 4y_3 + y_4 + 3y_5 \le 200$
$\sum_{i=1}^{5} x_i \ge 1, \quad \sum_{i=1}^{5} y_i \ge 2$
$0 \le x_i \le 100, \quad 0 \le y_i \le 100$

where

$b = (80, 57, 92, 55, 76, 14, 47, 51, 36, 92)^t$

$$B = \begin{bmatrix} 1 & 7 & 5 & 5 & 0 & -6 & -3 & -3 & 5 & -7 \\ -3 & 3 & 8 & 7 & -9 & -7 & -9 & 0 & 8 & -7 \\ 1 & 0 & 1 & 3 & 8 & 9 & 0 & 9 & -7 & -8 \\ -1 & -2 & 2 & 0 & 9 & 5 & -3 & 1 & -1 & -5 \\ -5 & 8 & -8 & 0 & 3 & 0 & 4 & -5 & -2 & 9 \\ 4 & -1 & 6 & -4 & -7 & -8 & -7 & 6 & -2 & -9 \\ 0 & 7 & 4 & 0 & 9 & 0 & 0 & -6 & -5 & -5 \\ -5 & -1 & 0 & 7 & -1 & 2 & 5 & -8 & -5 & 2 \\ -4 & -7 & 0 & -9 & 2 & 6 & -9 & 1 & -5 & 0 \\ -2 & 6 & 0 & 8 & -6 & 8 & 8 & 5 & 2 & -7 \end{bmatrix}$$

Global Solution: $x^* = (100, 0, 0, 80.94, 0)$, $y^* = (0, 0, 17.828, 0, 63.523)$, $f(x^*, y^*) = -794.86$.

PROBLEM 4: CQP1

(Floudas and Pardalos (1990) no: 2.5)

Minimize $c^t x - 0.5x^t Qx + d^t y$
subject to $AX \leq b$

$X = (x, y)^t$

$0 \leq X \leq 1$

$x \in R^7, \qquad y \in R^3$

where

$b = (-4, 22, -6, -23, -12, -3, 1, 12, 15, 9, -1)^t$
$Q = 10I$
$d = (10, 10, 10)$
$c = (-20, -80, -20, -50, -60, -90, 0)$

$$A = \begin{bmatrix} -2 & -6 & -1 & 0 & -3 & -3 & -2 & -6 & -2 & -2 \\ 6 & -5 & 8 & -3 & 0 & 1 & 3 & 8 & 9 & -3 \\ -5 & 6 & 5 & 3 & 8 & -8 & 9 & 2 & 0 & -9 \\ 9 & 5 & 0 & -9 & 1 & -8 & 3 & -9 & -9 & -3 \\ -8 & 7 & -4 & -5 & -9 & 1 & -7 & -1 & 3 & -2 \\ -7 & -5 & -2 & 0 & -6 & -6 & -7 & -6 & 7 & 7 \\ 1 & -3 & -3 & -4 & -1 & 0 & -4 & 1 & 6 & 0 \\ 1 & -2 & 6 & 9 & 0 & -7 & 9 & -9 & -6 & 4 \\ -4 & 6 & 7 & 2 & 2 & 0 & 6 & 6 & -7 & 4 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

Global Solution: $(x^*, y^*) = (1, 0.907, 0, 1, 0.715, 1, 0, 0.917, 1, 1)$, $f(x^*, y^*) = -267.959$.

PROBLEM 5: CQP2

(Floudas and Pardalos (1990) no: 2.6)

Minimize $c^t x - 0.5 x^t Q x$
subject to $Ax \le b$
$0 \le x_i \le 1 \quad i = 1,\ldots,10$
$x \in R^{10}$

where

$b = (-4, 22, -6, -23, -12)^t$
$Q = 100I$
$d = (10, 10, 10)$
$c = (48, 42, 48, 45, 44, 41, 47, 42, 45, 46)$

$$A = \begin{bmatrix} -2 & -6 & -1 & 0 & -3 & -3 & -2 & -6 & -2 & -2 \\ 6 & -5 & 8 & -3 & 0 & 1 & 3 & 8 & 9 & -3 \\ -5 & 6 & 5 & 3 & 8 & -8 & 9 & 2 & 0 & -9 \\ 9 & 5 & 0 & -9 & 1 & -8 & 3 & -9 & -9 & -3 \\ -8 & 7 & -4 & -5 & -9 & 1 & -7 & -1 & 3 & -2 \end{bmatrix}$$

Global Solution: $x^* = (1, 0, 0, 1, 1, 1, 0, 1, 1, 1)$, $f(x^*) = -39$.

PROBLEM 6: CQP3

(Floudas and Pardalos (1990) no: 2.7)

Minimize $f(x) = -0.5\sum_{i=1}^{20}(x_i - 2)^2$

subject to $Ax \leq b$

$x_i \geq 0 \quad i = 1, \ldots, 20$

$x \in R^{20}$

where

$b = (-5, 2, -1, -3, 5, 4, -1, 0, 9, 40)^t$

$$A^t = \begin{bmatrix}
-3 & 7 & 0 & -5 & 1 & 1 & 0 & 2 & -1 & 1 \\
7 & 0 & -5 & 1 & 1 & 0 & 2 & -1 & -1 & 1 \\
0 & -5 & 1 & 1 & 0 & 2 & -1 & -1 & -9 & 1 \\
-5 & 1 & 1 & 0 & 2 & -1 & -1 & -9 & 3 & 1 \\
1 & 1 & 0 & 2 & -1 & -1 & -9 & 3 & 5 & 1 \\
1 & 0 & 2 & -1 & -1 & -9 & 3 & 5 & 0 & 1 \\
0 & 2 & -1 & -1 & -9 & 3 & 5 & 0 & 0 & 1 \\
2 & -1 & -1 & -9 & 3 & 5 & 0 & 0 & 1 & 1 \\
-1 & -1 & -9 & 3 & 5 & 0 & 0 & 1 & 7 & 1 \\
-1 & -9 & 3 & 5 & 0 & 0 & 1 & 7 & -7 & 1 \\
-9 & 3 & 5 & 0 & 0 & 1 & 7 & -7 & -4 & 1 \\
3 & 5 & 0 & 0 & 1 & 7 & -7 & -4 & -6 & 1 \\
5 & 0 & 0 & 1 & 7 & -7 & -4 & -6 & -3 & 1 \\
0 & 0 & 1 & 7 & -7 & -4 & -6 & -3 & 7 & 1 \\
0 & 1 & 7 & -7 & -4 & -6 & -3 & 7 & 0 & 1 \\
1 & 7 & -7 & -4 & -6 & -3 & 7 & 0 & -5 & 1 \\
7 & -7 & -4 & -6 & -3 & 7 & 0 & -5 & 1 & 1 \\
-7 & -4 & -6 & -3 & 7 & 0 & -5 & 1 & 1 & 1 \\
-4 & -6 & -3 & 7 & 0 & -5 & 1 & 1 & 0 & 1 \\
-6 & -3 & 7 & 0 & -5 & 1 & 1 & 0 & 2 & 1
\end{bmatrix}$$

Global Solution: $x^* = (0, 0, 28.8024, 0, 0, 4.1792, 0, 0, 0, 0, 0, 0, 0, 0, 0.6188, 4.0933, 0, 2.3064, 0, 0)$, $f(x^*) = -394.7506$.

PROBLEM 7: CQP4

(The same as Problem 6 except the objective function)

$$f(x) = -0.5 \sum_{i=1}^{20} (x_i + 2)^2$$

Global Solution: $x^* =$ (0, 0, 28.8024, 0, 0, 4.1792, 0, 0, 0, 0, 0, 0, 0, 0, 0.6188, 4.0933, 0, 2.3064, 0, 0), $f(x^*) = -884.75058$.

PROBLEM 8: CQP5

(The same as Problem 6 except the objective function)

$$f(x) = -10 \sum_{i=1}^{20} x_i^2$$

Global Solution: $x^* =$ (0, 0, 28.8024, 0, 0, 4.1792, 0, 0, 0, 0, 0, 0, 0, 0, 0.6188, 4.0933, 0, 2.3064, 0, 0), $f(x^*) = -8695.01193$.

PROBLEM 9: CQP6

(The same as Problem 6 except the objective function)

$$f(x) = -0.5 \sum_{i=1}^{20} (x_i - 8)^2$$

Global Solution: $x^* = (0, 0, 28.8024, 0, 0, 4.1792, 0, 0, 0, 0, 0, 0, 0, 0, 0.6188, 4.0933, 0, 2.3064, 0, 0)$, $f(x^*) = -754.75062$.

PROBLEM 10: CQP7

(The same as Problem 6 except the objective function)

$$f(x) = -0.5\sum_{i=1}^{20} i(x_i - 2)^2$$

Global Solution: $x^* = (0, 0, 28.8024, 0, 0, 4.1792, 0, 0, 0, 0, 0, 0, 0, 0, 0.6188, 4.0933, 0, 2.3064, 0, 0)$, $f(x^*) = -4105.2279$.

PROBLEM 11: IQP

(Floudas and Pardalos (1990) no: 2.10)

$$\begin{aligned} \text{Minimize} \quad & f(x, y) = -0.5\sum_{i=1}^{10} \lambda_i(x_i - \alpha_i)^2 + 0.5\sum_{i=1}^{10} \mu_i(y_i - \beta_i)^2 \\ \text{subject to} \quad & A_1x + A_2y \leq b \\ & x, y \geq 0 \\ & x \in R^{10},\ y \in R^{10} \end{aligned}$$

where

$\lambda = (63, 15, 44, 91, 45, 50, 89, 58, 86, 82)$
$\mu = (42, 98, 48, 91, 11, 63, 61, 61, 38, 26)$
$\alpha = (-19, -27, -23, -53, -42, 26, -33, -23, 41, 19)$
$\beta = (-52, -3, 81, 30, -85, 68, 27, -81, 97, -73)$

$$A_1 = \begin{bmatrix} 3&5&5&6&4&4&5&6&4&4 \\ 5&4&5&4&1&4&4&2&5&2 \\ 1&5&2&4&7&3&1&5&7&6 \\ 3&2&6&3&2&1&6&1&7&3 \\ 6&6&6&4&5&2&2&4&3&2 \\ 5&5&2&1&3&5&5&7&4&3 \\ 3&6&6&3&1&6&1&6&7&1 \\ 1&2&1&7&8&7&6&5&8&7 \\ 8&5&2&5&3&8&1&3&3&5 \\ 1&1&1&1&1&1&1&1&1&1 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 8&2&4&1&1&1&2&1&7&3 \\ 3&6&1&7&7&5&8&7&2&1 \\ 1&7&2&4&7&5&3&4&1&2 \\ 7&7&8&2&3&4&5&8&1&2 \\ 7&5&3&6&7&5&8&4&6&3 \\ 4&1&7&3&8&3&1&6&2&8 \\ 4&3&1&4&3&6&4&6&5&4 \\ 2&3&5&5&4&5&4&2&2&8 \\ 4&5&5&6&1&7&1&2&2&4 \\ 1&1&1&1&1&1&1&1&1&1 \end{bmatrix}$$

$b^t = (380, 415, 385, 405, 470, 415, 400, 460, 400, 200)$

Global Solution: $x^* = (0, 0, 0, 0, 0, 4.348, 0, 0, 0, 0)$, $y^* = (0, 0, 0, 62.609, 0, 0, 0, 0, 0, 0, 0)$, $f(x^*, y^*) = 49318$.

PROBLEM 12:

(Visweswaran and Floudas (1993), Hansen et al. (1993))

Minimize $\{-y^3 + 4.5y^2 - 6y : 0 \le y \le 3\}$

Global Solution: $y^* = 3$, $f(y^*) = -4.5$.

PROBLEM 13:

(Dixon and Szego (1975), Visweswaran and Floudas (1993), Hansen et al. (1993))

Minimize $\{y^4 - 4y^3 + 4y^2 : -5 \le y \le 5\}$

Global Solution: y^* {0 or 2}, $f(y^*) = 0$.

PROBLEM 14:

(Dixon (1990), Visweswaran and Floudas (1993), Hansen et al. (1993))

Minimize $\{y^4 - 3y^3 - 1.5y^2 + 10y : -5 \leq y \leq 5\}$

Global Solution: $y^* = -1$, $f(y^*) = -7.5$.

PROBLEM 15:

(Wilkinson (1963), Visweswaran and Floudas (1993), Hansen et al. (1993))

Minimize $\{0.2y^5 - 1.6995y^4 + 0.998266y^3 - 0.0218343y^2 + 0.000089248y : 1 \leq y \leq 10\}$

Global Solution: $y^* = 6.325$, $f(y^*) = -443.67$.

PROBLEM 16:

(Goldstein and Price (1971), Visweswaran and Floudas (1993), Hansen et al. (1993))

Minimize $\{y^6 - 15y^4 + 27y^2 + 250 : -5 \leq y \leq 5\}$

Global Solution: $y^* \in$ {-3 or 3}, $f(y^*) = 7$.

PROBLEM 17:

(Dixon and Szego (1975), Visweswaran and Floudas (1993), Hansen et al. (1993))

Minimize $\left\{\frac{1}{6}y^6 - 1.05y^4 + 1.75y^2 : -5 \leq y \leq 5\right\}$

Global Solution: $y^* = 0$, $f(y^*) = 0$.

PROBLEM 18:

(Wingo (1985), Visweswaran and Floudas (1993), Hansen et al. (1993))

$$\text{Minimize } \left\{ \frac{1}{6}y^6 - \frac{52}{25}y^5 + \frac{39}{80}y^4 + \frac{71}{10}y^3 - \frac{79}{20}y^2 - y + \frac{1}{10} \; : \; -2 \le y \le 11 \right\}$$

Global Solution: $y^* = 10$, $f(y^*) = -29763.233$.

PROBLEM 19:

(Floudas and Pardalos (1990) no 4.6, Visweswaran and Floudas (1993), Hansen et al. (1993))

$$\begin{aligned} &\text{Minimize} && -x-y \\ &\text{subject to} && y \le 2x^4 - 8x^3 + 8x^2 + 2 \\ &&& y \le 4x^4 - 32x^3 + 88x^2 - 96x + 36 \\ &&& 0 \le x \le 3, \quad 0 \le y \le 4 \end{aligned}$$

Solution: $(x^*, y^*) = (2.3295, 3.1783)$, $f(x^*, y^*) = -5.5079$.

PROBLEM 20: ✓

(Manousiouthakis and Sourlas (1992), Ryoo and Sahinidis (1993))

$$\begin{aligned} &\text{Minimize} && x_1^4 - 14x_1^2 + 24x_1 - x_2^2 \\ &\text{subject to} && -x_1 + x_2 - 8 \le 0 \\ &&& x_2 - x_1^2 - 2x_1 + 2 \le 0 \\ &&& -8 \le x_1 \le 10, \quad 0 \le x_2 \le 10 \end{aligned}$$

Global Solution: $f(x^*) = -118.705$.

PROBLEM 21: PP1

(Soland (1971), Stephanopoulos and Westerberg (1975), Floudas et al. (1989), Swaney (1990), Ryoo and Sahinidis (1993))

$$\begin{aligned} &\text{Minimize} && \cancel{\surd}-12x_1 - 7x_2 + x_2^2 \\ &\text{subject to} && -2x_1^4 - x_2 + 2 = 0 \\ &&& 0 \le x_1 \le 2, \quad 0 \le x_2 \le 3 \end{aligned}$$

Global Solution: $f(x^*) = -16.7389.$

PROBLEM 22: QPP1

(Colville (1968) no: 3, Dembo (1976), Himmelblau (1972), Hock and Schittkowski (1981) no: 83, Floudas and Pardalos (1990) no: 3.2, Hansen et al. (1991))

$$\begin{aligned} &\text{Minimize} && c_1x_3^2 + c_2x_1x_5 + c_3x_1 - c_4 \\ &\text{subject to} && 0 \le a_1 + a_2x_2x_5 + a_3x_1x_4 - a_4x_3x_5 \le 92 \\ &&& 90 \le a_5 + a_6x_2x_5 + a_7x_1x_2 + a_8x_3^2 \le 110 \\ &&& 20 \le a_9 + a_{10}x_3x_5 + a_{11}x_1x_3 + a_{12}x_3x_4 \le 25 \\ &&& 78 \le x_1 \le 102 \\ &&& 33 \le x_2 \le 45 \\ &&& 27 \le x_j \le 45 \qquad j = 3, 4, 5 \end{aligned}$$

where $(a_i,\ i = 1,\dots,12) =$ (85.334407, 0.0056858, 0.0006262, 0.0022053, 80.51249, 0.0071317, 0.0029955, 0.0021813, 9.300961, 0.0047026, 0.0012547, 0.0019085),

$(c_i,\ i = 1,\dots,4) =$ (5.3578547, 0.8356891, 37.293239, 40792.141).

Global Solution: $x^* =$ (78, 33, 29.99526, 45, 36.77581), $f(x^*) = -30665.53867.$

PROBLEM 23: PP2

(Bartholomew-Biggs (1976), Hock and Schittkowski (1981) no: 71)

Minimize $x_1x_4(x_1 + x_2 + x_3) + x_3$

subject to ✓ $x_1x_2x_3x_4 - 25 \geq 0$

$x_1^2 + x_2^2 + x_3^2 + x_4^2 - 40 = 0$

$1 \leq x_j \leq 5 \qquad j = 1, \ldots, 4$

Best Known Solution: $x^* = (1, 4.7429994, 3.8211503, 1.3794082)$, $f(x^*) = 17.0140173$.

PROBLEM 24: PP3 Flywheel Design

(Siddall (1972), Eason and Fenton (1974), Schittkowski (1987) no: 343)

Minimize $-0.0201x_1^4x_2x_3^2/10^7$

subject to $x_1^2x_2 \leq 675$

$x_1^2x_3^2 \leq 0.419 \times 10^7$

$0 \leq x_1 \leq 36, \quad 0 \leq x_2 \leq 5, \quad 0 \leq x_3 \leq 125$

Solution: $x^* = (16.51, 2.477, 124)$ $f(x^*) = -5.684802$.

PROBLEM 25: QPP2 Heat Exchanger Design

(Avriel and Williams (1971), Dembo (1976), Hock and Schittkowski (1981) no: 106, Floudas and Pardalos (1990) no: 3.1, Hansen et al. (1991))

$$\begin{aligned}
\text{Minimize} \quad & x_1 + x_2 + x_3 \\
\text{subject to} \quad & .0025(x_4 + x_6) \leq 1 \\
& .0025(-x_4 + x_5 + x_7) \leq 1 \\
& .01(-x_5 + x_8) \leq 1 \\
& 100x_1 - x_1x_6 + 833.33252x_4 \leq 83333.333 \\
& x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \leq 0 \\
& x_3x_5 - x_3x_8 - 2500x_5 \leq -1250000 \\
& 100 \leq x_1 \leq 10000 \\
& 1000 \leq x_j \leq 10000, \qquad j = 2, 3 \\
& 10 \leq x_j \leq 1000, \qquad j = 4, \ldots, 8
\end{aligned}$$

Global Solution: $x^* = (579.31, 1359.97, 5109.97, 182.02, 295.6, 217.98, 286.42, 395.60)$, $f(x^*) = 7049.25$.

By a simple observation of the three linear constraints, upper bounds on some of the variables are improved as follows:

$x_4 \leq 390, \quad x_2 \leq 780, \quad x_6 \leq 390, \quad x_7 \leq 780, \quad x_8 \leq 880$

PROBLEM 26: QPP3 Reactor Network Design

(Manousiouthakis and Sourlas (1992), Ryoo and Sahinidis (1993))

$$\begin{aligned}
\text{Minimize} \quad & -x_4 \\
\text{subject to} \quad & x_1 - 1 + k_1x_1x_5 = 0 \\
& x_2 - x_1 + k_2x_2x_6 = 0 \\
& x_3 + x_1 - 1 + k_3x_3x_5 = 0 \\
& x_4 - x_3 + x_2 - x_1 + k_4x_4x_6 = 0 \\
& x_5^{0.5} + x_6^{0.5} \leq 4 \\
& 0 \leq x_j \leq 1 \quad j = 1, \ldots, 4 \\
& 0 \leq x_5 \leq 16 \quad 0 \leq x_6 \leq 16
\end{aligned}$$

where $k_1 = 0.09755988$, $k_2 = 0.99k_1$, $k_3 = 0.0391908$, and $k_4 = 0.9k_3$.

Global Solution: $f(x^*) = -0.3888$.

The last constraint can be equivalently represented by using variable definitions as follows:

$$x_7^2 = x_5, \quad x_8^2 = x_6, \quad x_7 + x_8 \le 4$$
$$0 \le x_7 \le 4 \quad 0 \le x_8 \le 4$$

PROBLEM 27: QPP4 Pooling Problem

(Haverly (1978), Lasdon el al. (1979), Liebman et al. (1986), Swaney (1990), Visweswaran and Floudas (1990), Floudas and Pardalos (1990) no: 6.2, Ryoo and Sahinidis (1993))

$$\begin{array}{ll}
\text{Minimize} & -9x_5 - 15x_9 + 6x_1 + 16x_2 + 10x_7 + 10x_8 \\
\text{subject to} & x_1 + x_2 = x_3 + x_4 \\
& x_3 + x_7 = x_5 \\
& x_4 + x_8 = x_9 \\
& x_3x_6 + 2x_7 \le 2.5x_5 \\
& x_4x_6 + 2x_8 \le 1.5x_9 \\
& 3x_1 + x_2 - x_3x_6 - x_4x_6 = 0 \\
& 0 \le x_1 \le 300, \quad 0 \le x_2 \le 300, \quad 0 \le x_3 \le 100, \quad 0 \le x_4 \le 200 \\
& 0 \le x_5 \le 100, \quad 1 \le x_6 \le 3, \quad 0 \le x_7 \le 100, \quad 0 \le x_8 \le 200 \\
& 0 \le x_9 \le 200
\end{array}$$

✓

Best Known Solution: $x^* = (0, 100, 0, 100, 0, 1, 0, 100, 200)$, $f(x^*) = -400$.

PROBLEM 28: QPP5 3-Stage Membrane Separation

(Dembo (1976), Hock and Schittkowski (1981) no: 116)

Minimize $x_{11} + x_{12} + x_{13}$

subject to
$$x_3 - x_2 \geq 0$$
$$x_2 - x_1 \geq 0$$
$$1 - .002x_7 + .002x_8 \geq 0$$
$$50 \leq x_{11} + x_{12} + x_{13} \leq 250$$
$$x_{11} - 1.262626x_8 + 1.231059x_1x_8 \geq 0$$
$$x_{12} - 1.262626x_9 + 1.231059x_2x_9 \geq 0$$
$$x_{13} - 1.262626x_{10} + 1.231059x_3x_{10} \geq 0$$
$$x_4 - .03475x_1 - .975x_1x_4 + .00975x_1^2 \geq 0$$
$$x_5 - .03475x_2 - .975x_2x_5 + .00975x_2^2 \geq 0$$
$$x_6 - .03475x_3 - .975x_3x_6 + .00975x_3^2 \geq 0$$
$$x_5x_7 - x_1x_8 - x_4x_7 + x_4x_8 \geq 0$$
$$1 - .002(x_2x_9 + x_5x_8 - x_1x_8 - x_6x_9) - x_5 - x_6 \geq 0$$
$$x_2x_9 - x_3x_{10} - x_6x_9 - 500x_2 + 500x_6 + x_2x_{10} \geq 0$$
$$x_2 - .002(x_2x_{10} - x_3x_{10}) \geq .9$$
$$.1 \leq x_1 \leq 1, \quad .1 \leq x_2 \leq 1, \quad .9 \leq x_3 \leq 1, \quad .0001 \leq x_4 \leq .1$$
$$.1 \leq x_5 \leq .9, \quad .1 \leq x_6 \leq .9, \quad .1 \leq x_7 \leq 1000, \quad .1 \leq x_8 \leq 1000$$
$$500 \leq x_9 \leq 1000, \quad .1 \leq x_{10} \leq 500, \quad 1 \leq x_{11} \leq 150$$
$$.0001 \leq x_{12} \leq 150, \quad .0001 \leq x_{13} \leq 150$$

Best Known Solution: $x^* =$ (.8037703, .8999860, .9709724, .09999952, .1908154, .4605717, 574.0803, 74.08043, 500.0162, .1, 20.23413, 77.34755, .00673039)

$f(x^*) = 97.588409$.

PROBLEM 29: PP4 Alkylation Process

(Bracken and McCormick (1968), Hock and Schittkowski (1981) no: 114)

Minimize $5.04x_1 + 0.035x_2 + 10x_3 + 3.36x_5 - 0.063x_4x_7$

subject to $g_1(x) = 35.82 - .222x_{10} - bx_9 \geq 0$

$g_2(x) = -133 + 3x_7 - ax_{10} \geq 0$

$g_3(x) = -g_1(x) + (1/b - b)x_9 \geq 0$

$g_4(x) = -g_2(x) + (1/a - a)x_{10} \geq 0$

$g_5(x) = 1.12x_1 + .13167x_1x_8 - .00667x_1x_8^2 - ax_4 \geq 0$

$g_6(x) = 57.425 + 1.098x_8 - .038x_8^2 + .325x_6 - ax_7 \geq 0$

$g_7(x) = -g_5(x) + (1/a - a)x_4 \geq 0$

$g_8(x) = -g_6(x) + (1/a - a)x_7 \geq 0$

$g_9(x) = 1.22x_4 - x_1 - x_5 = 0$

$g_{10}(x) = 98000x_3/(x_4x_9 + 1000x_3) - x_6 = 0$

$g_{11}(x) = (x_2 + x_5)/x_1 - x_8 = 0$

$.00001 \leq x_1 \leq 2000, \quad .00001 \leq x_2 \leq 16000, \quad .00001 \leq x_3 \leq 120$

$.00001 \leq x_4 \leq 5000, \quad .00001 \leq x_5 \leq 2000, \quad 85 \leq x_6 \leq 93$

$90 \leq x_7 \leq 95, \quad 3 \leq x_8 \leq 12, \quad 1.2 \leq x_9 \leq 4, \quad 145 \leq x_{10} \leq 162$

where $a = .99$ and $b = .9$.

Best Known Solution: $x^* = (1698.096, 15818.73, 54.10228, 3031.226, 2000, 90.11537, 95, 10.49336, 1.5616136, 153.53535)$, $f(x^*) = -1768.80696$.

To obtain a polynomial programming formulation for this problem, substitute $y_1 = 1/x_1$ and $y_2 = 1/(x_4x_9 + 1000x_3)$ in $g_{10}(x)$ and $g_{11}(x)$ above. Then, include the following constraints:

$y_1x_1 = 1$

$y_2x_4x_9 + 1000y_2x_3 = 1$

$1/2000 \leq y_1 \leq 1/0.00001$

$1/(5000 \times 4 + 1000 \times 120) \leq y_2 \leq 1/(0.00001 \times 1.2 + 1000 \times 0.00001)$

An easier way to construct a polynomial programming formulation for this problem is to multiply $g_{10}(x) = 0$ and $g_{11}(x) = 0$ by $(x_4x_9 + 1000x_3)$ and x_1, respectively:

$$g_{10}(x) = 98000x_3 - x_4x_6x_9 - 1000x_3x_6 = 0$$
$$g_{11}(x) = x_2 + x_5 - x_1x_8 = 0$$

PROBLEM 30: Transformer Design

(Bartholomew-Biggs (1976), Hock and Schittkowski (1981) no: 93)

$$\begin{array}{ll} \text{Minimize} & .0204x_1x_4(x_1 + x_2 + x_3) + .0187x_2x_3(x_1 + 1.57x_2 + x_4) \\ & + .0607x_1x_4x_5^2(x_1 + x_2 + x_3) + .0437x_2x_3x_6^2(x_1 + 1.57x_2 + x_4) \\ \text{subject to} & .001x_1x_2x_3x_4x_5x_6 - 2.07 \geq 0 \\ & .00062x_1x_4x_5^2(x_1 + x_2 + x_3) + .00058x_2x_3x_6^2(x_1 + 1.57x_2 + x_4) \leq 1 \\ & x_j \geq 0, \quad j = 1,\ldots, 6 \end{array}$$

Best Known Solution: $x^* =$ (5.332666, 4.656744, 10.43299, 12.08230, .7526074, .87865084), $f(x^*) = 135.075961$.

PROBLEM 31:

(Stephanopoulos and Westerberg (1975), Floudas, Aggarwal and Ciric (1989), Floudas and Pardalos (1990) no: 4.3)

$$\begin{array}{ll} \text{Minimize} & x_1^{0.6} + x_2^{0.6} - 6x_1 - 4x_3 - 3x_4 \\ \text{subject to} & -3x_1 + x_2 - 3x_3 = 0 \\ & x_1 + 2x_3 \leq 4 \\ & x_2 + 2x_4 \leq 4 \\ & x_1 \leq 3, \quad x_3 \leq 1 \\ & x_1,\ x_2,\ x_3,\ x_4 \geq 0 \end{array}$$

Best Known Solution: $x^* =$ (4/3, 4, 0, 0), $f(x^*) = -4.5142$.

After using implied upper bounds on x_2 and x_4, and introducing new variables y_1, y_2, we can cast the above problem in our desired form:

$$
\begin{array}{ll}
\text{Minimize} & y_1 + y_2 - 6x_1 - 4x_3 - 3x_4 \\
\text{subject to} & -3x_1 + x_2 - 3x_3 = 0 \\
& x_1 + 2x_3 \leq 4 \\
& x_2 + 2x_4 \leq 4 \\
& x_1^3 - y_1^5 = 0 \\
& x_2^3 - y_2^5 = 0 \\
& 0 \leq x_1 \leq 3, \quad 0 \leq x_2 \leq 12 \\
& 0 \leq x_3 \leq 1, \quad 0 \leq x_4 \leq 2 \\
& 0 \leq y_1 \leq 3^{3/5}, \quad 0 \leq y_2 \leq 12^{3/5}
\end{array}
$$

PROBLEM 32: Maximum Clique Problem

(Pardalos and Phillips (1990), Floudas and Pardalos (1990) no: 2.9): Given an undirected graph $G(V, E)$, where V is the set of vertices and E is the set of edges,

$$
\begin{array}{ll}
\text{Minimize} & -\sum_{(i,j)\in E} x_i x_j \\
\text{subject to} & \sum_{v_i \in V} x_i = 1 \\
& x_i \geq 0 \quad i = 1, \dots, |V|
\end{array}
$$

If k is the size of the maximum clique of G, then $f(x^*) = -0.5(1 - (1/k))$. The global minimum is defined by $x_i^* = (1/k)$, if the vertex v_i belongs to the maximum clique, and zero otherwise.

PROBLEM 33:

(Wolfe (1972), Pierre and Lowe (1975) pp.255 ex.6.7, Schittkowski (1987) no: 368)

Minimize $\left(\sum_{i=1}^{8} x_i^3\right)^2 - \left(\sum_{i=1}^{8} x_i^2\right)\left(\sum_{i=1}^{8} x_i^4\right)$

subject to $0 \leq x_i \leq 1 \quad i = 1,\ldots, 8$

Global Solution:

Any four of the total eight variables are equal to 1, and the remaining four variables are equal to 1/2, $f(x^*) = -1$. Global minima occur at 70 distinct points. Also, there are 185 other local minima.

Appendix B

Optimal Dual Completion of γ-variables One At A Time

In this section, we present an exact method for optimizing a single γ_{kl} dual variable at a time while other dual variables are fixed at their current values, as proposed in Section 4.3. Recall that the function $\phi(\gamma_{kl})$ in (4.14) is a piecewise concave function. The exact optimization method we propose is based on finding a point $\gamma_{kl} = \gamma^*_{kl}$, where there exists a subgradient of $\phi(\gamma_{kl})$ having zero value; hence, satisfying a sufficient condition for optimality.

In the branch-and-bound algorithm, we re-scale each node subproblem using the scaling scheme in Section 4.4.1. Therefore, we assume that,

$$0 \le x_i \le 1 \quad \forall\ i = 1,\dots,n \tag{B.1.1}$$

in the following procedure. Later, we show that by employing a suitable scaling scheme, the procedure given below can be easily adopted for any set of valid simple bounds on the x-variables.

Case 1 : First, let us consider the case $s_{kl} \le 0$. Under the assumption (B.1), let us re-write (4.14) as follows:

If $\gamma_{kl} < s_{kl}$,

$$\phi(\gamma_{kl}) = \phi_0 + \underset{(B.1)}{\text{minimum}}\{p_{kk}x_k^2 + (p_k + \gamma_{kl})x_k + p_{ll}x_l^2 + p_l x_l\}. \tag{B.2.1}$$

If $s_{kl} \le \gamma_{kl} < 0$,

$$\phi(\gamma_{kl}) = \phi_0 + \underset{(B.1)}{\text{minimum}}\{p_{kk}x_k^2 + (p_k + \gamma_{kl})x_k + p_{ll}x_l^2 + (p_l + s_{kl} - \gamma_{kl})x_l\}. \tag{B.2.2}$$

If $\gamma_{kl} \ge 0$,

$$\phi(\gamma_{kl}) = \phi_0 + \underset{(B.1)}{\text{minimum}}\{p_{kk}x_k^2 + (p_k + \gamma_{kl})x_k + p_{ll}x_l^2 + (p_l + s_{kl})x_l - \gamma_{kl}\}. \tag{B.2.3}$$

Let $\bar{x}$ be the minimal solution in (B.2). Then, we have a subgradient of $\phi(\cdot)$ as follows.

$$\phi'(\gamma_{kl}) = \begin{cases} \bar{x}_k & \text{if } \gamma_{kl} < s_{kl} \\ \bar{x}_k - \bar{x}_l & \text{if } s_{kl} \le \gamma_{kl} < 0 \\ \bar{x}_k - 1 & \text{if } \gamma_{kl} \ge 0 \end{cases} \tag{B.3}$$

Let γ_{kl}^* be a maximizing point of $\phi(\gamma_{kl})$. Since $\bar{x}_k \ge 0$ and $(\bar{x}_k - 1) \le 0$, there exists $\gamma_{kl}^* \in [s_{kl}, 0]$. Next, we search for a point γ_{kl}^* where $\phi'(\cdot)$ changes sign, or $\phi'(\gamma_{kl}^*) = 0$.

Case 1.1 : $p_{kk} \le 0$ and $p_{ll} \le 0$.

Due to the condition of Case 1.1, for $s_{kl} \le \gamma_{kl} \le 0$, we determine $\bar{x}_k$ and $\bar{x}_l$ as follows.

If $p_{kk} + p_k + \gamma_{kl} \geq 0$ then $\bar{x}_k = 0$, otherwise $\bar{x}_k = 1$. $(B.4.1)$

If $p_{ll} + p_l + (s_{kl} - \gamma_{kl}) \geq 0$ then $\bar{x}_l = 0$, otherwise $\bar{x}_l = 1$. $(B.4.2)$

Notice that at $\gamma_{kl} = -(p_{kk} + p_k)$, the value of $\bar{x}_k$ changes from 1 to 0. From (B.3), $\phi'(-(p_{kk} + p_k)) \leq 0$, and $\phi'(-(p_{kk} + p_k) - \varepsilon) \geq 0$, for any $\varepsilon \geq 0$. Hence, we can set $\gamma^*_{kl} = -(p_{kk} + p_k)$.

Case 1.2 : $p_{kk} \leq 0$ and $p_{ll} > 0$.

The only difference from the previous case is that for any $\gamma_{kl} \in [s_{kl}, 0)$, to determine $\bar{x}_k$ and $\bar{x}_l$, we use (B.4) unless $0 \leq -(p_l + s_{kl} - \gamma_{kl})/2p_{ll} \leq 1$, in which case $\bar{x}_l = -(p_l + s_{kl} - \gamma_{kl})/2p_{kk}$. However, using the same argument in Case 1.1, we can find γ^*_{kl} by manipulating $\bar{x}_k$, and we set $\gamma^*_{kl} = -(p_{kk} + p_k)$.

Case 1.3 : $p_{kk} > 0$ and $p_{ll} \leq 0$.

For $-(2p_{kk} + p_k) \leq \gamma_{kl} \leq -p_k$, $\bar{x}_k = -(p_k + \gamma_{kl})/2p_{kk}$. Otherwise, $\bar{x}_k$ is determined by (B.4.1).

1. At $\gamma_{kl} = s_{kl}$, if $\bar{x}_l = 1$, that is $p_{ll} + p_l < 0$, then $\phi'(s_{kl}) \leq 0$. Also, we know that $\phi'(s_{kl} - \varepsilon) \geq 0$, for any $\varepsilon > 0$. Hence, set $\gamma^*_{kl} = s_{kl}$.

2. At $\gamma_{kl} = 0$, if $\bar{x}_l = 0$, that is $p_{ll} + p_l + s_{kl} \geq 0$, then since $\bar{x}_l = 0$, for all $\gamma_{kl} \in R^1$, we have $\phi'(\gamma_{kl}) \geq 0$, for all $\gamma_{kl} < 0$. Also, we know that $\phi'(0) \leq 0$. Hence, set $\gamma^*_{kl} = 0$.

3. For $s_{kl} < \gamma_{kl} < 0$, since γ^*_{kl} has not been fixed yet, we have $s_{kl} \leq p_{ll} + p_l + s_{kl} < 0$. From (B.4), $\bar{x}_l$ changes its value from 0 to 1 at $\gamma_{kl} = p_{ll} + p_l + s_{kl}$, then $\phi'(p_{ll} + p_l + s_{kl}) \geq 0$, while $\phi'(p_{ll} + p_l + s_{kl} + \varepsilon) \leq 0$, for any $\varepsilon > 0$. Hence, set $\gamma^*_{kl} = p_{ll} + p_l + s_{kl}$.

Case 1.4 : $p_{kk} > 0$ and $p_{ll} > 0$.

Define $C_x \equiv \{y_{kl} : p_l + s_{kl} \leq y_{kl} \leq 2p_{ll} + p_l + s_{kl}\}$. For $y_{kl} \in [s_{kl}, 0)$, if $y_{kl} \in C_x$ then $\bar{x}_l = -(p_l + s_{kl} - y_{kl})/2p_{ll}$. Otherwise, $\bar{x}_l$ is determined by (B.4.2). For $y_{kl} < s_{kl}$, $\bar{x}_l$ is constant with the same value at $y_{kl} = s_{kl}$. For $y_{kl} \geq 0$, $\bar{x}_l$ is constant with the same value at $y_{kl} = 0$. Now, we update y_{kl} as follows.

1) If $s_{kl} \geq 2p_{ll} + p_l + s_{kl}$, then $\bar{x}_l = 1$, for $y_{kl} \geq s_{kl}$, so $\phi'(s_{kl} + \varepsilon) \leq 0$, $\forall \varepsilon > 0$. We also know that $\phi'(s_{kl} - \varepsilon) \geq 0$, $\forall \varepsilon > 0$. Hence, set $y^*_{kl} = s_{kl}$.

2) If $0 \leq p_l + s_{kl}$, then $\bar{x}_l = 0$, for $y_{kl} \leq 0$, so $\phi'(0 - \varepsilon) \geq 0$, $\forall \varepsilon > 0$. We also know that $\phi'(0 + \varepsilon) \leq 0$, $\forall \varepsilon > 0$. Hence, set $y^*_{kl} = 0$.

At this point, suppose y^*_{kl} is not fixed yet. Since $\bar{x}_l \neq 1$ at $y_{kl} = s_{kl}$, and $\bar{x}_l \neq 0$ at $y_{kl} = 0$, we know that $C_x \bigcap [s_{kl}, 0] \neq \emptyset$. For a moment, suppose that we remove the bound restrictions on x_k and x_l. Then $\bar{x}_k = \bar{x}_l$ at

$$y_{kl} = \frac{-p_k p_{ll} + (p_l + s_{kl}) p_{kk}}{p_{ll} + p_{kk}} \equiv y_{kl}{}^c. \qquad (B.5)$$

3) If $p_l + s_{kl} \leq s_{kl}$ and $y_{kl}{}^c \leq s_{kl}$, then at $y_{kl} = s_{kl}$, $\bar{x}_l \geq \bar{x}_k$, that is $\phi(s_{kl}) \leq 0$ and $y^*_{kl} = s_{kl}$. If $p_l + s_{kl} \geq s_{kl}$ and $y_{kl}{}^c \leq p_l + s_{kl}$ then at $y_{kl} = p_l + s_{kl}$, $\bar{x}_k = 0$ and $\bar{x}_l = 0$, so set $y^*_{kl} = p_l + s_{kl}$. We can combine the above two IF statements as : If $y_{kl}{}^c \leq \max\{p_l + s_{kl},\ s_{kl}\}$, then set $y^*_{kl} = \max\{p_l + s_{kl},\ s_{kl}\}$.

4) If $2p_{ll} + p_l + s_{kl} \leq 0$ and $y_{kl}{}^c \geq 2p_{ll} + p_l + s_{kl}$, then at $y_{kl} = 2p_{ll} + p_l + s_{kl}$, $\bar{x}_l = 1$ and $\bar{x}_k = 1$. So set $y^*_{kl} = 2p_{ll} + p_l + s_{kl}$. If $2p_{ll} + p_l + s_{kl} \geq 0$ and $y_{kl}{}^c \geq 0$, then at $y_{kl} = 0$, $\bar{x}_k \geq \bar{x}_l$, i.e. $\phi'(0 - \varepsilon) \geq 0$, $\forall \varepsilon > 0$. Since we know that $\phi'(0) \leq 0$, we set $y^*_{kl} = 0$. We can combine the above two IF statements as : If $y_{kl}{}^c \geq \min\{2p_{ll} + p_l + s_{kl},\ 0\}$, then set $y^*_{kl} = \min\{2p_{ll} + p_l + s_{kl},\ 0\}$.

5) At this point, we know that $y_{kl}{}^c \in C_x \bigcap [s_{kl}, 0)$; hence, set $y^*_{kl} = y_{kl}{}^c$.

Case 2 : Now, we consider the case $s_{kl} > 0$. For this case, $\phi(\gamma_{kl})$ reduces to the following form.

If $\gamma_{kl} < 0$,

$$\phi(\gamma_{kl}) = \phi_0 + \underset{(4.11.3)}{\text{minimum}}\{p_{kk}x_k^2 + p_{ll}x_l^2 + p_kx_k + p_lx_l + (s_{kl} - \gamma_{kl})(l_lx_k + l_kx_l - l_kl_l) \quad (B.6.1)$$
$$+ \gamma_{kl}(u_lx_k + l_kx_l - l_ku_l)\}.$$

If $0 \le \gamma_{kl} < s_{kl}$,

$$\phi(\gamma_{kl}) = \phi_0 + \underset{(4.11.3)}{\text{minimum}}\{p_{kk}x_k^2 + p_{ll}x_l^2 + p_kx_k + p_lx_l + (s_{kl} - \gamma_{kl})(l_lx_k + l_kx_l - l_kl_l) \quad (B.6.2)$$
$$+ \gamma_{kl}(u_lx_k + u_kx_l - u_ku_l)\}.$$

If $\gamma_{kl} \ge s_{kl}$,

$$\phi(\gamma_{kl}) = \phi_0 + \underset{(4.11.3)}{\text{minimum}}\{p_{kk}x_k^2 + p_{ll}x_l^2 + p_kx_k + p_lx_l + (s_{kl} - \gamma_{kl})(l_lx_k + u_kx_l - u_kl_l) \quad (B.6.3)$$
$$+ \gamma_{kl}(u_lx_k + u_kx_l - u_ku_l)\}.$$

Under the condition (B.1), we have a subgradient of $\phi(\gamma_{kl})$ as follows.

$$\phi'(\gamma_{kl}) = \begin{cases} \bar{x}_k & \text{if } \gamma_{kl} < 0 \\ \bar{x}_k - (1 - \bar{x}_l) & \text{if } 0 \le \gamma_{kl} < s_{kl} \\ \bar{x}_k - 1 & \text{if } \gamma_{kl} \ge s_{kl} \end{cases} \quad (B.7)$$

Then, we find an optimal γ_{kl}^* in a similar fashion as in Case 1 as summarized below.

Case 2.1 : $p_{kk} \le 0$. Set $\gamma_{kl}^* = -(p_{kk} + p_k)$.

Case 2.2 : $p_{kk} > 0$ and $p_{ll} \le 0$. Set

$\gamma_{kl}^* = 0$, if $p_{ll} + p_l \ge 0$

$\gamma_{kl}^* = -(p_{ll} + p_l)$, if $-s_{kl} \le p_{ll} + p_l < 0$

$\gamma_{kl}^* = s_{kl}$, if $p_{ll} + p_l < -s_{kl}$

Case 2.3 : $p_{kk} > 0$ and $p_{ll} > 0$. Set

$\gamma^*_{kl} = 0$, if $-p_l \leq 0$

$\gamma^*_{kl} = s_{kl}$, if $-(p_l + 2p_{ll}) \geq s_{kl}$

Or else, given that the above two conditions do not occur within Case 2.3, we define $\gamma_{kl}{}^c \equiv (-p_k(p_{ll} + p_l) - p_l(p_{kk} + p_k))/(p_{ll} + p_{kk})$, and set

$\gamma^*_{kl} = \max\{0, -(p_l + p_{ll})\}$, if $\gamma_{kl}{}^c \leq \max\{0, -(p_l + p_{ll})\}$

$\gamma^*_{kl} = \min\{s_{kl}, -p_l\}$, if $\gamma_{kl}{}^c \geq \min\{s_{kl}, -p_l\}$

$\gamma^*_{kl} = \gamma_{kl}{}^c$, otherwise.

As mentioned earlier, we can adopt the above procedure for any set of bound restrictions $l_i \leq x_i \leq u_i$, $i = 1, \ldots, n$. Given the coefficients p_k, p_{kk}, p_l, p_{ll}, and s_{kl} in problem (4.12) corresponding to the bound restrictions $l \leq x \leq u$, the required coefficients in the above procedure, which are valid for the scaled bounds $0 \leq x \leq 1$, and which are now denoted by superscript "s" below, can be obtained using the variable redefinitions (4.30), as follows.

$$p_i^s = (u_i - l_i)(2l_i p_{ii} + p_i + s_{kl} l_j) \quad i \in \{k, l\}$$

$$p_{ii}^s = (u_i - l_i)^2 p_{ii} \quad i \in \{k, l\}$$

$$s_{kl}^s = s_{kl}(u_k - l_k)(u_l - l_l)$$

Let $\gamma_{kl}{}^s = \gamma^{*s}_{kl}$ be the resulting optimal value for the scaled problem, then we set the optimal value of the original "unscaled" γ_{kl} in problem (4.12) as $\gamma^*_{kl} = \gamma^{*s}_{kl}/(u_k - l_k)(u_l - l_l)$.

Appendix C

Recovering Primal Optimal Solutions

In general, conjugate subgradient algorithms applied to a Lagrangian dual problem do not produce a primal optimal/feasible solutions corresponding to the optimal dual solution. We use a two stage procedure to recover a good quality primal feasible solution. The first stage consists of averaging the primal optimal solutions obtained while solving the Lagrangian relaxation subproblems over the conjugate subgradient iterations at which the Lagrangian dual solution improves in value (see Sherali and Choi, 1994). When the algorithm terminates, if the resulting average solution is not feasible, then using this solution as the starting point for a penalty function based routine, a feasible solution is recovered.

More specifically, the first stage keeps the running average of the x-variable part of the primal optimal solution to the Lagrangian relaxation subproblem for each conjugate subgradient iteration at which the Lagrangian dual incumbent improves. Let x^0 be the final average value, then we define $w^0_{kl} = x^0_k x^0_l$ $\forall 1 \leq k \leq l \leq n$.

In the second stage, before setting up a penalty function for RLT-NLP, let us re-group the constraints for convenience as the set

$$\left[H_i^t(x, w) \le h_i, \quad i \in I_p \equiv \{1, \ldots, \frac{m(m+1)}{2} + 2mn + 2n(n-1)\} \right]$$

which represents the constraints (3.24.1)-(3.24.7), and let us denote the remaining set of constraints as $X_p \equiv \{(x, w) : (3.24.8), (3.24.9)\}$ Finally, let us denote the objective function (3.24.0) of RLT-NLP in vector form as $C^t(x, w)$.

The penalty function we employ is a hybrid l_1-augmented Lagrangian penalty function (see Sherali and Ulular (1989)) given as follows.

$$P(x, w) = C^t(x, w) + \frac{1}{2} \sum_{i \in I_p} \mu_i \max^2\left\{(H_i^t(x, w) - h_i + \frac{\pi_i}{\mu_i}),\ 0\right\} - \frac{1}{2} \sum_{i \in I_p} \frac{\pi_i^2}{\mu_i} + \sum_{i \in I_p} \mu_i \max\{(H_i^t(x, w) - h_i),\ 0\} \tag{C.1}$$

Here the penalty parameters are defined as follows:

$$\mu_i = \min\left\{\theta_1 + \pi_i + \max\{(H_i^t(x^0, w^0) - h_i),\ 0\}\phi_1,\ \mu_{\max}\right\} \tag{C.2}$$

where π_i, $i = 1, \ldots, |I_p|$, are the incumbent dual multipliers corresponding to the constraints (3.24.1)-(3.24.7), and where $\theta_1 = 4$ and $\phi_1 = 5$, as recommended by Sherali and Ulular (1989). The parameter $\mu_{\max}$ is set at a large value of 90000. Note that the dual multiplier values corresponding to the constraints (3.24.1)-(3.24.3) are directly available from the incumbent dual solution to the Lagrangian dual problem. The values of the dual multipliers corresponding to the constraints (3.24.4)-(3.24.7) can be easily obtained from the Lagrangian subproblem corresponding to the incumbent dual solution as shown below.

Consider the Lagrangian subproblem obtained by dualizing (3.24.1)-(3.24.3) in RLT-NLP using the incumbent dual solution. Let this problem be stated as follows.

$$\hat{c}_0 + \text{Minimize} \quad \sum_{k=1}^{n} \hat{c}_k x_k + \sum_{k=1}^{n} \hat{q}_{kk} w_{kk} + \sum_{k=1}^{n-1} \sum_{l=k+1}^{n} (\hat{q}_{kl} w_{kl} + \hat{\gamma}_{kl} w'_{kl})$$

subject to $(3.24.4) - (3.24.9)$.

For some (k, l), $1 \leq k < l \leq n$, let β^1_{kl}, β^2_{kl}, β^3_{kl}, and β^4_{kl} be the dual multipliers associated with the constraints (3.24.4), (3.24.5), (3.24.6), and (3.24.7), respectively. Then, we can derive the values of these dual multipliers as follows:

$$\begin{array}{lll}
\beta^1_{kl} = \hat{q}_{kl}, & \beta^2_{kl} = 0, & \text{if } \hat{q}_{kl} \geq 0, \\
\beta^1_{kl} = 0, & \beta^2_{kl} = |\hat{q}_{kl}|, & \text{if } \hat{q}_{kl} < 0, \\
\beta^3_{kl} = \hat{\gamma}_{kl}, & \beta^4_{kl} = 0, & \text{if } \hat{\gamma}_{kl} \geq 0, \\
\beta^3_{kl} = 0, & \beta^4_{kl} = |\hat{\gamma}_{kl}|, & \text{if } \hat{\gamma}_{kl} < 0.
\end{array}$$

The corresponding penalty problem we solve is stated as **PRP**: Minimize $\{P(x, w)\text{: } (x, w) \in X_p\}$. To solve PRP, we adopt the previously described conjugate subgradient procedure. While the conjugate subgradient procedure keeps track of the incumbent solution for the problem PRP, we also keep track of a second solution (x^M, w^M) based on its merit with respect to near feasibility.

To measure the infeasibility of a point (x, w), let us define the function $F_H(x, w) = \sum_{i \in I_p} \max\{H_i^t(x, w) - h_i, 0\}/\|H_i\|$. At the k^{th} iteration, we update (x^M, w^M) by setting it to the current solution (x^k, w^k), if at least one of the following conditions are satisfied.

$$\begin{aligned} &(i) \quad P(x^k, w^k) < P(x^M, w^M) - 0.01|P(x^M, w^M)| \\ &(ii) \quad P(x^k, w^k) \in [P(x^M, w^M) \pm 0.01|P(x^M, w^M)|] \quad \text{and} \quad F_H(x^k, w^k) < F_H(x^M, w^M) \end{aligned} \tag{C.3}$$

Let us now state some other implementation details for solving Problem PRP.

• We used the above defined point (x^0, w^0) as the starting solution.

• In the step-length computations, as a lower bound on the optimal value to PRP, we used the Lagrangian dual incumbent value.

• A subgradient of $P(x, w)$ at the iterate (x^k, w^k) is given as follows:

$$C + \sum_{i \in I_p} \mu_i \max\{H_i^t(x^k, w^k) - h_i + \frac{\pi_i}{\mu_i}, 0\} H_i + \sum_{i \in I_p} \mu_i \begin{bmatrix} 1, & \text{if } H_i^t(x^k, w^k) - h_i \geq 0 \\ 0, & \text{otherwise} \end{bmatrix} H_i .$$

• At each iteration, moving along the deflected subgradient using the prescribed step-length may lead to a point that is infeasible to X_p. In this case, we need to project this point onto X_p. Suppose that $(\hat{x}, \hat{w}) \notin X_p$. Then, the projected point on X_p, say $(\tilde{x}, \tilde{w})$, is given via the solution to the minimum distance problem Minimize $\{ \|(x, w) - (\hat{x}, \hat{w})\|_2^2 : (x, w) \in X_p\}$. Notice that the set X_p does not put any restriction on the cross product term variables. Therefore, we can set $\tilde{w}_{kl} = \hat{w}_{kl}$ $\forall 1 \leq k < l \leq n$. Consequently, the foregoing minimum distance problem reduces to the following separable form:

$$\sum_{k=1}^{n} \underset{l_k \leq x_k \leq u_k}{\text{Minimize}} \left\{ (x_k - \hat{x}_k)^2 + (w_{kk} - \hat{w}_{kk})^2 : \ x_k^2 \leq w_{kk} \leq (u_k + l_k)x_k - u_k l_k \right\}, \qquad \text{(C.4)}$$

which can be solved explicitly as follows.

Exact Solution of Problem (C.4): The following procedure can be easily justified by observing the 2-dimensional feasible region for any $k \in \{1, \ldots, n\}$.

Step 0: If $\hat{x}_k^2 \leq \hat{w}_{kk} \leq (u_k + l_k)\hat{x}_k - u_k l_k$, then we have $(\tilde{x}_k, \tilde{w}_{kk}) = (\hat{x}_k, \hat{w}_k)$.

Step 1: If $-(u_k + l_k)\hat{x}_k + \hat{w}_{kk} + u_k l_k \geq 0$ then we first project $(\hat{x}_k, \hat{w}_{kk})$ onto the hyperplane $-(u_k + l_k)x_k + w_{kk} + u_k l_k = 0$ as follows:

$$\begin{pmatrix} x_k^{(1)} \\ w_{kk}^{(1)} \end{pmatrix} = \begin{pmatrix} \hat{x}_k \\ \hat{w}_{kk} \end{pmatrix} - \lambda \begin{pmatrix} -(u_k + l_k) \\ 1 \end{pmatrix}, \quad \text{where } \lambda = \frac{-(u_k + l_k)\hat{x}_k + \hat{w}_{kk} + u_k l_k}{(u_k + l_k)^2 + 1}.$$

If $x_k^{(1)} \in [l_k, u_k]$, then the projection is complete, and we have $(\tilde{x}_k, \tilde{w}_{kk}) = (x_k^{(1)}, w_{kk}^{(1)})$. Otherwise, we set $\tilde{x}_k$ to its closest bound to $x_k^{(1)}$, and subsequently, let $\tilde{w}_{kk} = \tilde{x}_k^2$.

Step 2: Now, $-(u_k + l_k)\hat{x}_k + \hat{w}_{kk} + u_k l_k < 0$, but $\hat{x}_k^2 > \hat{w}_{kk}$. We first find the closest point to $(\hat{x}_k, \hat{w}_{kk})$ on the surface $x_k^2 = w_{kk}$ (or equivalently on the region $\{(x_k, w_{kk}): x_k^2 \leq w_{kk}\}$) by solving the problem

$$\text{Minimize } \left\{(x_k - \hat{x}_k)^2 + (w_{kk} - \hat{w}_{kk})^2 : \; x_k^2 \leq w_{kk}\right\}. \tag{C.5}$$

Noting that at optimality, the constraint in (C.5) holds as equality, the dual feasibility KKT condition, aside from the nonnegativity restriction on the dual variable, requires the solution of the following third-order polynomial equation.

$$2x_k^3 + (1 - 2\hat{w}_{kk})x_k - \hat{x}_k = 0 \tag{C.6}$$

Exact methods to find the roots of a cubic equation date back to the early 1500's. Equation (C.6) can have from one to three distinct real roots. We pick the one that gives the minimum value for the problem (C.5), or we can use the nonnegativity condition on the dual variable given by $2(x_k^2 - \hat{w}_{kk})$ for the constraint in (C.5) to select the optimal solution. Let $x_k^{(2)}$ be the optimal solution thus obtained. If $x_k^{(2)} \in [l_k, u_k]$ then the projection is complete, and we set $(\tilde{x}_k, \tilde{w}_{kk}) = (x_k^{(2)}, (x_k^{(2)})^2)$. Otherwise, we set $\tilde{x}_k$ to its upper or lower bound that is closest to $x_k^{(2)}$, and subsequently, let $\tilde{w}_{kk} = \tilde{x}_k^2$.

Bibliography

Adler, I., and R. Monteiro (1991), "An Interior Point Algorithm Applied to a Class of Convex Separable Programming Problems," *TIMS/ORSA National Meeting*, Nashville, Tenn., May 12-15.

Al-Khayyal, F. A. (1992), "Generalized Bilinear Programming: Part I Models, Applications and Linear Programming Relaxation," *European J. of Oper. Res.*, **60**: 306-314.

Al-Khayyal, F. A. (1990), "Jointly Constrained Bilinear Programs and Related Problems: An Overview," *Computers Math. Applic.*, **19**: 53-62.

Al-Khayyal, F. A., and J. E. Falk (1983), "Jointly Constrained Biconvex Programming," *Math. of Oper. Res.*, **8**: 273-286.

Al-Khayyal, F. A., and C. Larson (1990), "Global Minimization of A Quadratic Function Subject to A Bounded Mixed Integer Constraint Set," *(manuscript)*, Dept. of ISyE, Ga Tech., Atlanta, GA 30332.

Al-Khayyal, F. A., C. Larson, and T. Van Voorhis (1994), "A Relaxation Method for Nonconvex Quadratically Constrained Quadratic Programs," Working Paper, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332.

Allufi-Pentini, F., F. Parisi, and F. Zirilli (1984), "Global Optimization and Stochastic Differential Equations," *JOTA*, **47**: 1-16.

Avriel, M., R. Dembo, and U. Passy (1975), "Solution of Generalized Geometric Programming," *International Journal of Numerical Methods in Engineering*, **9**: 149-169.

Avriel, M., A. C. Williams (1971), "An Extension of Geometric Programming with Applications in Engineering Optimization," *Journal of Engineering Mathematics*, **5(3)**: 187-194.

Avriel, M., A. C. Williams (1970), "Complementary Geometric Programming," *SIAM Journal of Appl. Math.*, **19**: 125-141.

Balas, E. (1975), "Nonconvex Quadratic Programming via Generalized Polars," *SIAM Journal on Applied Math.*, **28**: 335-349.

Balas, E., and J. B. Mazzola (1984a), "Nonlinear 0-1 Programming: I. Linearization Techniques," *Math. Progr.*, **30**: 1-21.

Balas, E., and J. B. Mazzola (1984b), "Nonlinear 0-1 Programming: II. Dominance Relations and Algorithms," *Math. Progr.*, **30**: 22-45.

Barr, A. S., S. C. Sarin, and A. G. Bishara (1989), "Procedure for Structural Optimization," *ACI Structural Journal*, **Sept-Oct**: 524-531.

Bartholomew-Biggs, M. C. (1976), "A Numerical Comparison Between Two Approaches to Nonlinear Programming Problems," Technical Report No. 77, Numerical Optimisation Centre, Hatfield, England.

Becker, R. W., and G. V. Lago (1970), "A Global Optimization Algorithm," in *Proceedings of the 8th Allerton Conference on Circuits and Systems Theory*.

Beightler, C. S., and D. T. Phillips (1976), *Applied Geometric Programming*, John Wiley & Sons, Inc., New York.

Benacer, R., and Pham Dinh Tao (1986), "Global Maximization of A Nondefinite Quadratic Function Over A Convex Polyhedron," pp. 65-76, in *Fermat Days 85: Mathematics for Optimization*, J. B. Hiriart-Urruty (ed.), North-Holland, Amsterdam.

Betro, B. (1991), "Bayesian Methods in Global Optimization," *Journal of Global Optimization*, **1**: 1-14.

Betro, B. (1984), "Bayesian Testing of Nonparametric Hypotheses and Its Applications to Global Optimization," *JOTA*, **42**: 31-50.

Boender, C. G. E., and A. H. G. Rinnooy Kan (1983), "A Bayesian Analysis of the Number of Cells in a Multinomial Distribution," *The Statistician*, **32**: 240-248.

Boender, C. G. E., A. H. G. Rinnooy Kan, L. Stougie, and G. T. Timmer (1982), "A Stochastic Method for Global Optimization," *Mathematical Programming*, **22**: 125-140.

Bomze, I. M. (1992), "Copositivity Conditions for Global Optimality Indefinite Quadratic Programming Problems," *Czechoslovak Journal for Operations Research*, **1**: 1-19.

Bomze, I. M., and G. Danninger (1993), "A Global Optimization Algorithm for Concave Quadratic Programming Problems," *SIAM J. Optimization*, **3(4)**: 826-842.

Bracken, J., and G. P. McCormick (1968), *"Selected Applications of Nonlinear Programming"*, John Wiley & Sons, Inc., New York.

Bromberg, M., and T. Chang (1992), "One Dimensional Global Optimization Using Linear Lower Bounds," pp. 200-220, in *Recent Advances in Global Optimization*, P. M. Pardalos, and C. A. Floudas (eds.), Princeton University Press, New Jersey.

Burns, S. A., and A. Locascio (1991), "A Monomial-Based Method for Solving Systems of Non-Linear Algebraic Equations," *International Journal For Numerical Methods In Engineering*, **31**: 1295-1318.

Burns, S. A. (1987), "Generalized Geometric Programming With Many Equality Constraints," *International Journal For Numerical Methods In Engineering*, **24**: 725-741.

Carraghan, R., and P. M. Pardalos (1990), "An Exact Algorithm for the Maximum Clique Problem," *O.R. Letters*, **9**: 375-382.

Chvatal, V. (1983), *Linear Programming*, W. H. Freeman and Company, New York.

Colville, A. R. (1968), "A Comparative Study on Nonlinear Programming Codes," IBM New York Scientific Center Report No. 320-2949.

Danninger, G., and I. M. Bomze (1993), "Using Copositivity for Global Optimality Criteria in Concave Quadratic Programming Problems," *Math. Prog.*, **62**: 575-580.

Dekkers, A., and E. Aarts (1991), "Global Optimization and Simulated Annealing," *Mathematical Programming.*, **50**: 367-393.

Dembo, R. S. (1976), "A Set of Geometric Programming Test Problems and Their Solutions," *Mathematical Programming*, **10**: 192-213.

Dixon, L. C. W. (1990), "On Finding the Global Minimum of a Function in One Variable," Presented at the SIAM National Meeting, Chicago.

Dixon, L. C. W., and G. P. Szego (1975), *Towards Global Optimization*, North Holland, Amsterdam.

Duffin, R. J., E. L. Peterson, and C. Zener (1967), *Geometric Programming*, John Wiley & Sons, Inc., New York.

Eason, E. D., and R. G. Fenton (1974), "A Comparison of Numerical Optimization Methods for Engineering Design," *Transactions of the ASME*, **96B(1)**: 196-200.

Falk, J. E., and K. L. Hoffman (1976), "A Successive Underestimation Method for Concave Minimization Problems," *Math. of Oper. Res.*, **1**: 251-259.

Fisher, M. L. (1981), "The Lagrangian Relaxation Methods for Solving Integer Programming Problems," *Management Science*, **27**: 1-18.

Floudas, C. A., A. Aggarwal, and A. R. Ciric (1989), "Global Optimum Search for Nonconvex NLP and MINLP Problems," *Computers Chem. Engng*, **13(10)**: 1117-1132.

Floudas, C. A., B. Jaumard, and V. Visweswaran (1994), Private Communication.

Floudas, C. A., and P. M. Pardalos (1990), *"A Collection of Test Problems for Constrained Global Optimization Algorithms"*, Springer-Verlag, Berlin.

Floudas, C. A., and V. Visweswaran (1993), "A Primal-Relaxed Dual Global Optimization Approach: Theory," *JOTA*, **78**:187-225.

Floudas, C. A., and V. Visweswaran (1990), "A Global Optimization Algorithm (GOP) for Certain Classes of Nonconvex NLP's I. Theory," *Computers and Chemical Engineering*, **14**: 1397-1417.

Gochet, W., and Y. Smeers (1979), "A Branch-and-Bound Method for Reversed Geometric Programming," *Operations Research*, **27(5)**: 982-996.

Golstein, A. A., and J. F. Price (1971), "On Descent from Local Minima," *Mathematics of Computation*, **25**: 569-574.

Golub, G. H., and C. F. Van Loan (1989), *Matrix Computations*, Second Edition, The Johns Hopkins University Press, Baltimore.

Guignard, M., and S. Kim (1987), "Lagrangian Decomposition: A Model Yielding Stronger Lagrangean Bounds," *Mathematical Programming*, **39**: 215-228.

Hamed A. S. E., and G. P. McCormick (1993) "Calculation of Bounds on Variables Satisfying Nonlinear Inequality Constraints," *Journal of Global Optimization*, **3**: 25-47.

Hansen, E. (1992), *Global Optimization Using Interval Analysis*, Monographs and textbooks in pure and applied mathematics, no: 165, Marcel Dekker, Inc., New York.

Hansen, E. (1988), "An Overview of Global Optimization Using Interval Analysis," pp. 289-308, in *Reliability in Computing*, R. E. Moore (ed.), Academic Press, Inc., Boston.

Hansen, P., and B. Jaumard (1992), "Reduction of Indefinite Quadratic Programs to Bilinear Programs," *Journal of Global Optimization*, **2(1)**: 41-60.

Hansen, P., B. Jaumard, and S. Lu (1991), "An Analytical Approach to Global Optimization," *Mathematical Programming*, **52**: 227-254.

Hansen, P., B. Jaumard, and V. Mathon (1989), "Constrained Nonlinear 0-1 Programming," Working paper, RUTCOR, Rutgers University, Rutgers, New Jersey.

Hansen, P., B. Jaumard, and J. Xiong (1993), "Decomposition and Interval Arithmetic Applied to Global Minimization of Polynomial and Rational Functions," *Journal of Global Optimization*, **3**: 421-437.

Haverly, C. A. (1978), "Studies of the Behavior of Recursion for the Pooling Problem," *SIGMAP Bull.*, **25**: 19.

Hesse, R. (1973), "A Heuristic Search Procedure for Estimating a Global Solution of Nonconvex Programming Problems," *Operations Research*, **21**: 1267-1280.

Himmelblau, D. M. (1972), *Applied Nonlinear Programming*, McGraw-Hill Company, New York.

Hiriart-Urruty, J. B., and C. Lemarechal (1990), " Testing Necessary and Sufficient Conditions for Global Optimality in The Problem of Maximizing A Convex Quadratic Function Over A Convex Polyhedron," Preliminary Report, Seminar of Numerical Analysis, Univ. Paul Sabatier, Toulouse.

Hiriart-Urruty, J. B. (1989), "From Convex Optimization to Nonconvex Optimization, Part I: Necessary and Sufficient Conditions for Global Optimality," pp. 219-239, in *Nonsmooth Optimization and Related Topics*, F. H. Clarke et al. (eds.), Plenum Press, New York.

Hock, W., and K. Schittowski (1981), *Test Examples for Nonlinear Programming Codes*, Springer-Verlag, Berlin.

Horst, R. (1986), "A General Class of Branch-and-Bound Methods in Global Optimization with Some New Approaches for Concave Minimization," *JOTA*, **51**: 271-291.

Horst, R. (1990), "Deterministic Methods in Constrained Global Optimization: Some Recent Advances and New Fields of Application," *Naval Research Logistics Quarterly*, **37**: 433-471.

Horst, R., and H. Tuy (1993), *Global Optimization: Deterministic Approaches*, 2nd ed., Springer-Verlag, Berlin, Germany.

Kalantari, B., and J. B. Rosen (1987), "An Algorithm for Global Minimization of Linearly Constrained Concave Quadratic Functions," *Math. of Oper. Res.*, **12**: 544-561.

Kamath, A., and N. Karmarkar (1992), "A Continuous Approach To Compute Upper Bounds In Quadratic Maximization Problems with Integer Constraints," pp. 147-159, in *Recent Advances in Global Optimization*, P. M. Pardalos, and C. A. Floudas (eds.), Princeton University Press, New Jersey.

Kapoor, S., and P. M. Vaidya (1986), "Fast Algorithms for Convex Quadratic Programming and Multicommodity Flows," in *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, ACM Press, New York.

Kirkpatrick, S., C. D. Crelatt, Jr., and M. P. Vecchi (1983), "Optimization by Simulated Annealing," *Science*, **220**: 671-680.

Kostreva, M. M., and L. A. Kinard (1991), "A Differentiable Homotopy Approach for Solving Polynomial Optimization Problems and Noncooperative Games," *Computers Math. Applic.*, **21(6/7)**: 135-143.

Kough, P. F., "The Indefinite Quadratic Programming Problem," *Operations Research*, **27(3)**: 516-533.

Kozlov, M. K., S. P. Tarasov, and L. G. Hacijan (1979), "Polynomial Solvability of Convex Quadratic Programming," *Soviet Mathematics Doklady*, **20**: 1108-1111.

Larsson, T., and Z. Lin (1989), "A Primal Convergence Result for Dual Subgradient Optimization with Applications to Multicommodity Network Flows," Research Report, Department of Mathematics, Linkoping Institute of Technology, S-581 83, Linkoping, Sweden.

Lasdon, L. S., A. D. Waren, S. Sarkar, and F. Palacios (1979), "Solving the Pooling Problem Using Generalized Reduced Gradient and Successive Linear Programming Algorithms," *SIGMAP Bull.*, **77**: 9-15.

Liebman, J., L. Lasdon, and A. Waren (1986), *Modelling and Optimization with GINO*, The Scientific Press, Palo Alto, CA.

Lucidi, S., and M. Piccioni (1989), "Random Tunneling by Means of Acceptance-Rejection Sampling for Global Optimization," *JOTA*, **62**: 255-277.

Manas, M. (1968), "An Algorithm for A Nonconvex Programming Problem," *Econ Math Obzor Acad. Nacl. Ceskoslov*, **4(2)**: 202-212.

Manousiouthakis, M., and D. Sourlas (1992), "A Global Optimization Approach to Rationally Constrained Rational Programming," *Chem. Eng. Comm.*, **115**: 127-147.

Maranas, C. D., and C. A. Floudas (1994), "Global Optimization in Generalized Geometric Programming," Working Paper, Department of Chemical Engineering, Princeton University, Princeton, N.J. 08544-5263.

McKeown, P. G. (1978), "Extreme Point Ranking Algorithms: A computational survey," pp. 216-222, in *Computers and Mathematical Programming*, W. W. White (ed.), National Bureau of Standards Special Publication 502, U. S. Government Printing Office, Washington, DC.

Meyer, G. G. (1988), "Convergence of Relaxation Algorithms by Averaging," *Mathematical Programming*, **40**: 205-212.

Mockus, J. (1994), "Application of Bayesian Approach to Numerical Methods of Global and Stochastic Optimization", *Journal of Global Optimization*, **4**: 347-365.

Mockus, J. (1989), *Bayesian Approach to Global Optimization*, Kluwer Academic Publishers, Boston.

Mockus, J., V. Tiesis, and A. Zilinskas (1978), "The Application of Bayesian Methods for Seeking the Extremum", in Dixon, L. C. W., and G. P. Szego (eds.), *Towards Global Optimization II*, North Holland, Amsterdam.

Moore, R. E. (1966), *Interval Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey.

Mueller R. K. (1970), "A Method for Solving The Indefinite Quadratic Programming Problem," *Management Science*, **16(5)**: 333-339.

Murtagh, B. A., and M. A. Saunders (1987), "MINOS 5.1 User's Guide," Technical Report Sol 83-20R, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, California.

Murty, K. G., and S. N. Kabadi (1987), "Some NP-Complete Problems in Quadratic and Nonlinear Programming," *Mathematical Programming*, **39**: 117-129.

Muu L. D., and W. Oettli (1991), "An Algorithm for Indefinite Quadratic Programming with Convex Constraints," *Operations Research Letters*, **10**: 323-327.

Neumaier, A. (1992), "An Optimality Criterion for Global Quadratic Optimization," *Journal of Global Optimization*, **2**: 201-208.

Pardalos, P. M. (1991), "Global Optimization Algorithms for Linearly Constrained Indefinite Quadratic Programs," *Computers Math. Applic.*, **21**: 87-97.

Pardalos, P. M. (1988), "Enumerative Techniques for Solving Some Nonconvex Global Optimization Problems," *OR Spektrum*, **10**: 29-35.

Pardalos, P. M., J. H. Glick, and J. B. Rosen (1987), "Global Minimization of Indefinite Quadratic Problems," *Computing*, **39**: 281-291.

Pardalos, P. M., and A. Phillips (1990), "Global Optimization Approach to The Maximum Clique Problem," *Int. J. Computer Math.*, **33**: 209-216.

Pardalos, P. M., and J. B. Rosen (1987), *Constrained Global Optimization: Algorithms and Applications*, Springer-Verlag, Berlin.

Pardalos, P. M., and G. Schnitger (1988), "Checking Local Optimality in Constrained Quadratic Programming Is NP-Hard," *Operations Research Letters*, **7(1)**: 33-35.

Pardalos, P. M., and S. A. Vavasis (1992), "Open Questions in Complexity Theory for Numerical Optimization," *Mathematical Programming*, **57**: 337-339.

Pardalos, P. M., and S. A. Vavasis (1991), "Quadratic Programming with One Negative Eigenvalue Is NP-Hard," *Journal of Global Optimization*, **1**: 15-22.

Patel, N. R., R. L. Smith, and Z. B. Zabinsky (1988), "Pure Adaptive Search in Monte Carlo Optimization," *Mathematical Programming*, **43**: 317-328.

Phillips, A. T., and J. B. Rosen (1990), "Guaranteed ε-Approximate Solution for Indefinite Quadratic Global Minimization," *Naval Research Logistics*, **37**: 499-514.

Phillips, A. T., J. B. Rosen, and M. Van Vliet (1992), "A Parallel Stochastic Method for Solving Linearly Constrained Concave Global Minimization Problem," *J. of Global Optimization*, **2**: 243-258.

Piccioni, M. (1987), "A Combined Multistart-Annealing Algorithm for Continuous Global Optimization," Technical Research Report 87-45, Systems and Research Center, The University of Maryland, College Park, MD.

Pierre, D. A., and M. J. Lowe (1975) *Mathematical Programming Via Augmented Lagrangians*, Addison-Wesley Publishing Co., London.

Quesada, I., and I. E. Grossmann (1992), "A Global Optimization Algorithm for Linear Fractional and Bilinear Programs," Working Paper, Department of Chemical Engineering, Carnegie Mellon U., Pittsburg, PA 15213.

Ratschek, H., and J. Rokne (1988), *New Computer Methods for Global Optimization*, John Wiley & Sons, New York.

Rijckaert, M. J., and X. M. Martens (1980), "Comparison of Generalized Geometric Programming," pp. 288-320, in *Advances in Geometric Programming*, M. Avriel (ed.), Plenum Press, New York.

Rinnooy Kan, A. H. G., and G. T. Timmer (1987a), "Stochastic Global Optimization Methods. Part I: Clustering Methods," *Mathematical Programming*, **39**: 27-56.

Rinnooy Kan, A. H. G., and G. T. Timmer (1987b), "Stochastic Global Optimization Methods. Part II: Multi-Level Methods," *Mathematical Programming*, **39**: 57-78.

Ritter K. (1966), "A Method for Solving Maximum Problems with A Nonconcave Quadratic Objective Function," *Z. Wahrscheinlihkeitstheoric*, **4**: 340-351.

Rosen, J. B. (1983), "Global Minimization of a Linearly Constrained Concave Function by Partition of Feasible Domain," *Math. of Oper. Res.*, **8**: 215-230.

Rosenbrock, H. H. (1960), "An Automatic Method for Finding the Greatest or Least Value of a Function," *Computer Journal*, **3**: 175-184.

Ryoo, H. S., and N. V. Sahinidis (1993), "Global Optimization of Nonconvex NLPs and MINLPs with Applications in Process Design," Technical Report, Department of Mechanical and Industrial Engineering, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801.

Sahni, S (1974), "Computationally Related Problems," *SIAM Journal on Computing*, **3**: 262-279.

Schittowski, K. (1987), *More Test Examples for Nonlinear Programming Codes*, Springer-Verlag, Berlin.

Schoen, F. (1991), "Stochastic Techniques for Global Optimization: A Survey of Recent Advances," *Journal of Global Optimization*, **1**: 207-228.

Sherali, H. D., and W. P. Adams (1994), "A Hierarchy of Relaxations and Convex Hull Characterizations for Mixed-Integer Zero-One Programming Problems," *Discrete Applied Mathematics*, **52**: 83-106.

Sherali, H. D., and W. P. Adams (1990), "A Hierarchy of Relaxations Between the Continuous and Convex Hull Representations for Zero-One Programming Problems," *SIAM Journal on Discrete Mathematics*, **3**: 411-430.

Sherali, H. D., and A. R. Alameddine (1992), "A New Reformulation-Linearization Technique for Bilinear Programming Problems," *Journal of Global Optimization*, **2(3)**: 379-410.

Sherali, H. D., and A. R. Alameddine (1990), "An Explicit Characterization of the Convex Envelope of a Bivariate Bilinear Function Over Special Polytopes," *Annals of Operations Research*, **25**: 197-210.

Sherali, H. D., and G. Choi (1994), "Recovery of Primal Solutions When Using Subgradient Optimization Methods to Solve Lagrangian Duals of Linear Programs," Working Paper, Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061.

Sherali, H. D., and D. C. Myers (1985/6), "The Design of Branch and Bound Algorithms for a Class of Nonlinear Integer Programs," *Annals of Oper. Res.*, **5**: 463-484.

Sherali, H. D., S. Ramachandran, and S. Kim (1993), "A Localization and Reformulation Discrete Programming Approach for the Rectilinear Distance Location-Allocation Problem," *Discrete Appl. Math.*, to appear.

Sherali, H. D., and C. H. Tuncbilek (1994), "A Reformulation-Convexification Approach for Solving Nonconvex Quadratic Programming Problems," under revision for *The Journal of Global Optimization*.

Sherali, H. D., and C. H. Tuncbilek (1992b), " A Global Optimization Algorithm for Polynomial Programming Problems Using a Reformulation-Linearization Technique," *The Journal of Global Optimization*, **2**: 101-112.

Sherali, H. D., and C. H. Tuncbilek (1992a), "A Squared-Euclidean Distance Location-Allocation Problem," *Naval Research Logistics*, **39**: 447-469.

Sherali, H. D., and O. Ulular (1989), "A Primal-Dual Conjugate Subgradient Algorithm for Specially Structured Linear and Convex Programming Problems," *Appl. Math. Optim.*, **20**: 193-221.

Shor, N. Z. (1990a), "Dual Estimates In Multiextremal Problems," Working Paper, Institute of Cybernetics, Academy of Sciences of the Ukrainian SSR, Kiev 252207, USSR. (Presented at the IIASA Workshop on Global Optimization, Sopron, Hungary (1990)).

Shor, N. Z. (1990b), "Dual Quadratic Estimates In Polynomial and Boolean Programming," *Annals of Operations Research*, **25**: 163-168.

Siddall, J. N. (1972), *"Analytical Decision Making in Engineering Design"*, Section 5.3, Prentice-Hall, Englewood Cliffs, N.J..

Soland, R. M. (1971), "An Algorithm for Separable Nonconvex Programming Problems II: Nonconvex Constraints," *Management Science*, **17**: 759-773.

Stephanopoulos, G., and A. W. Westerberg (1975), "The Use of Hestenes' Method of Multipliers to Resolve Dual Gaps in Engineering System Optimization," *JOTA*, **15**: 285-309.

Swaney, R. E. (1990), "Global Solution of Algebraic Nonlinear Programs," AIChE Annual meeting, Chicago.

Thoai, N. V., and H. Tuy (1980), "Convergent Algorithms for Minimizing a Concave Function," *Math. of Oper. Res.*, **5**: 556-566.

Torn, A. A. (1978), "A Search Clustering Approach to Global Optimization," in Dixon, L. C. W., and G. P. Szego (eds.), *Towards Global Optimization*, North Holland, Amsterdam.

Torn, A., and A. Zilinskas (1987), *Global Optimization*, Springer-Verlag, New York.

Tuy, H. (1987), "Global Minimization of a Difference of Two Convex Functions," *Mathematical Programming Study*, **30**: 150-182.

Tuy, H. (1964), "Concave Programming Under Linear Constraints," *Soviet Math.*, **5**: 1437-1440.

Vasudevan, G., L. T. Watson, and F. H. Lutze (1989), "A Homotopy Approach for Solving Global Optimization Problems," *Proc. Amer. Cont. Conf.*, Pittsburg, 780-785.

Vavasis, S. A. (1992), "Approximation Algorithms for Indefinite Quadratic Programming," *Mathematical Programming*, **57**: 279-311.

Vavasis, S. A. (1991), *Nonlinear Optimization: Complexity Issues*, Oxford University Press, New York.

Vavasis, S. A. (1990), "Quadratic Programming Is in NP," *Information Processing Letters*, **36**: 73-77.

Visweswaran, V, and C. A. Floudas (1993), "New Properties and Computational Improvement of the GOP Algorithm for Problems with Quadratic Objective Function and Constraints," *Journal of Global Optimization*, **3**: 439-462.

Visweswaran, V, and C. A. Floudas (1992), "Unconstrained and Constrained Global Optimization of Polynomial Functions In One Variable," *Journal of Global Optimization*, **2**: 73-99.

Visweswaran, V, and C. A. Floudas (1990), "A Global Optimization Algorithm (GOP) for Certain Classes of Nonconvex NLP's-II. Application of Theory and Test Problems," *Computers chem. Engng*, **14**: 1419-1434.

Wang, X., and T. Chang (1994), "An Improved Linear Lower Bound and A Fast One Dimensional Global Optimization Algorithm," Working Paper, Dept. of Mathematics, University of California, Davis, CA 95616.

Watson, L. T. (1986), "Numerical Linear Algebra Aspects of Globally Convergent Homotopy Methods," *SIAM Review*, **28(4)**: 529-545.

Watson, L. T., S. C. Billups, and A. P. Morgan (1987), "Algorithm 652 HOMPACK: A Suite of Codes for Globally Convergent Homotopy Algorithms," *ACM Transactions on Mathematical Software*, **13(3)**: 281-310.

Wilkinson, J. H. (1963), *Rounding Errors in Algebraic Processes*, Prentice-Hall, Engelwood Cliffs, N.J.

Wingo, D. R. (1985), "Globally Minimizing Polynomials without evaluating Derivatives," *International Journal of Computer Mathematics*, **17**: 287-294.

Wolfe, P. (1972), "Explicit Solution of an Optimization Problem" *Mathematical Programming*, **2**: 258-260.

Ye, Y. (1992), "On Affine Scaling Algorithms for Nonconvex Quadratic Programming," *Mathematical Programming*, **56**: 285-300.

Ye, Y., and E. Tse (1989), "An Extension of Karmarkar's Projective Algorithm for Convex Quadratic Programming," *Mathematical Programming*, **44**: 157-179.

Zabinsky, Z. B., R. L. Smith, J. F. McDonald, H. E. Romeijn, and D. E. Kaufman (1993), "Improving Hit-and-Run for Global Optimization," *J. of Global Optimization*, **3**: 171-192.

Zabinsky, Z. B., and R. L. Smith (1992), "Pure Adaptive Search in Global Optimization," *Mathematical Programming*, **53**: 323-338.

Zhigljavsky, A. A. (1991), *Theory of Global Random Search*, Kluwer Academic Publishers, Boston.

Zikan, K. (1990), "Track Initialization and Multiple Object Tracking Problem," Working Paper, Department of Operations Research, Stanford University, Stanford, California 94305.

Zilinskas, A. (1985), "Axiomatic Characterization of a Global Optimization Algorithm and Investigation of Its Search Strategy," *Operations Research Letters*, **4**: 35-39.

Zilinskas, A. (1982), "Axiomatic Approach to Statistical Models and Their Use in Multimodal Optimization Theory," *Mathematical Programming*, **22**: 104-116.

Zwart, P. B. (1973), "Nonlinear Programming: Counterexamples To Two Global Optimization Algorithms," *Operations Research*, **21(6)**: 1260-1266.

Vita

Cihan Halit Tuncbilek was born in Istanbul, Turkey on December 12, 1963. He graduated from Bogazici University, Istanbul where he received a Bachelor of Science degree in Industrial Engineering in 1987. He attended Virginia Polytechnic Institute and State University where he received a Master of Science degree in Industrial Engineering and Operations Research in June 1990, and a Doctor of Philosophy degree in Industrial and Systems Engineering in December 1994.

[illegible]