Global optimality conditions and optimization methods for polynomial programming problems and their applications

Jing Tian

This thesis is submitted in total fulfilment of the
requirement for the degree of Doctor of Philosophy

School of Science, Information Technology and Engineering
Federation University Australia
PO Box 663
University Drive, Mount Helen
Ballarat, VIC 3353, Australia.

March 2014

Abstract

The polynomial programming problem which has a polynomial objective function, either with no constraints or with polynomial constraints occurs frequently in engineering design, investment science, control theory, network distribution, signal processing and location-allocation contexts. Moreover, the polynomial programming problem is known to be Non-deterministic Polynomial-time hard (NP-hard). The polynomial programming problem has attracted a lot of attention, including quadratic, cubic, homogenous or normal quartic programming problems as special cases.

Existing methods for solving polynomial programming problems include algebraic methods and various convex relaxation methods. Especially, among these methods, semidefinite programming (SDP) and sum of squares (SOS) relaxations are very popular. Theoretically, SDP and SOS relaxation methods are very powerful and successful in solving the general polynomial programming problem with a compact feasible region. However, the solvability in practice depends on the size or the degree of the polynomial programming problem and the required accuracy. Hence, solving large scale SDP problems still remains a computational challenge.

It is well-known that traditional local optimization methods are designed based on necessary local optimality conditions, i.e., Karush-Kuhn-Tucker (KKT) conditions. Motivated by this, some researchers proposed a necessary global optimality condition for a quadratic programming problem and designed a new local optimization method according to the necessary global optimality condition. In this thesis, we try to apply this idea to cubic and quatic

programming problems, and further to general unconstrained and constrained polynomial programming problems. For these polynomial programming problems, we will investigate necessary global optimality conditions and design new local optimization methods according to these conditions. These necessary global optimality conditions are generally stronger than KKT conditions. Hence, the obtained new local minimizers by using the new local optimization methods may improve some KKT points.

Our ultimate aim is to design global optimization methods for these polynomial programming problems. We notice that the filled function method is one of the well-known and practical auxiliary function methods used to achieve a global minimizer. In this thesis, we design global optimization methods by combining the new proposed local optimization methods and some auxiliary functions. The numerical examples illustrate the efficiency and stability of the optimization methods.

Finally, we discuss some applications for solving some sensor network localization problems and systems of polynomial equations. It is worth mentioning that we apply the idea and the results for polynomial programming problems to nonlinear programming problems (NLP). We provide an optimality condition and design new local optimization methods according to the optimality condition and design global optimization methods for the problem (NLP) by combining the new local optimization methods and an auxiliary function. In order to test the performance of the global optimization methods, we compare them with two other heuristic methods. The results demonstrate our methods outperform the two other algorithms.

Statement of Authorship

This thesis contains no work extracted in whole or in part from a thesis, dissertation or research paper previously presented for another degree or diploma except where explicit reference is made. No other person's work has been relied upon or used without due acknowledgment in the main text and bibliography of the thesis.

Signed: Feng Tian

Date: 27/03/2014

Acknowledgement

First of all, I would like to express heartfelt thanks to my parents. Without their rearing and educating, I could not have the ability to do such high level research work, or even started it. I also appreciate the support and help from Dean of SITE, Professor John Yearwood, for offering an opportunity to do my PhD study at Federation University Australia.

Most of all, I would like to dedicate my appreciation to my principal supervisor Associate Professor Zhiyou Wu. Her hard-working, rigorous scholarship infected and inspired me. I feel deeply the honor of being her student. I would not accomplish so much without her guidance, constant support and help. Also, I would like to appreciate the effort of my associate supervisor, Dr. Julien Ugon. I have learned a lot from him, and to him I express many thanks for his great guidance and support. I would also like to say thanks to all SITE staff for their support and all the students in our PhD office for their friendship. I felt enjoyable during my research.

Finally, my deepest gratitude goes to my husband, my son and my little daughter, for their love and support. To my husband: thank you for your understanding and encouragement. To my son and my little daughter: thank you for bringing me so much joy and laughter.

Dedication

To my son Louis Y. Zhao and my daughter Hannah Y. Zhao

List of publication

Journal papers

1. J. Tian, Z.Y. Wu and J. Ugon, Optimization methods for a class of integer polynomial programming problems, *Operations Research Transactions*, 15(4), 2011, 23–35.

2. Z.Y. Wu, J. Quan, G.Q. Li and J. Tian, Necessary optimality conditions and new optimization methods for cubic polynomial optimization problems with mixed variables, *Journal of Optimization Theory and Applications*, 153(2), 2012, 408–435.

3. Z.Y. Wu, Y.J. Yang, F.S. Bai and J. Tian, Global optimality conditions and optimization methods for quadratic assignment problems, *Applied Mathematics and Computation*, 218, 2012, 6214–6231.

4. Z.Y. Wu, J. Tian, J. Quan and J. Ugon, Optimality conditions and optimization methods for quartic polynomial optimization, *Applied Mathematics and Computation*, 232, 2014, 968–982.

5. Z.Y. Wu, J. Tian and J. Ugon, Global optimality conditions and optimization methods for polynomial programming problems, *Journal of Global Optimization*, (submitted)

6. Z.Y. Wu, J. Tian, J. Ugon and L. Zhang, Global optimality conditions and optimization methods for constrained polynomial programming problems, *Applied Mathematics and Computation*, (submitted)

7. Z.Y. Wu, J. Tian and F.S. Bai, Optimality condition and optimization methods for non-linear programming problems, *Computational Optimization and Applications*, (sub-mitted)

Conference presentation

1. Zhiyou Wu, Jing Tian, Optimality conditions and optimization methods for quartic polynomial optimization with mixed variables, *The 2011 Annual Research Conference, University of Ballarat*, November, 2011.

Contents

List of Tables

List of Figures

Introduction

The polynomial programming problem which is a fundamental model in the field of optimization represents a broad range of applications. These include engineering design, investment science, control theory, network distribution, signal processing and location-allocation contexts. Many well-known test functions are polynomial functions, for example, Rosenbrock, Wood, Powell quartic, Six-hump camelback and Goldstein and Price functions. Moreover, some functions, such as sin, log and radicals, can be reformulated into polynomial functions, which extends the applications of polynomial programming problems. The polynomial programming problems are NP-hard. Indeed, even some quadratic programming problems are NP-hard.

For global optimization, a great deal of attention has been focused on two areas: one is global optimality conditions; the other is global optimization methods to solve problems. Over the years, various global optimality conditions for quadratic programming problems and some special classes of polynomial programming problems have been established. The development of checkable global optimality conditions for other polynomial programming problems and general polynomial programming problems remains an important research topic.

When it comes to using global optimization methods to solve polynomial programming problems, perhaps the very first attempt for solving polynomial programming problems is to treat them as nonlinear programming problems. Methods of solving these problems relied on local optimization techniques.

Then, polynomial programming problems attracted more attention. Many researchers focused on methods for solving polynomial programming problems, which include quadratic, cubic, quartic and 0-1 integer programming problems as special cases. There are two mainly methods to solve polynomial programming problems: exact algebraic algorithms and various relaxation methods.

Exact algebraic algorithms, which find all the critical points and then compare the function values of the polynomial at these points, were established. Existing methods include Grobner bases and Stetter-moller method, Resultant method, eigenvalues of companion matrices and Homotopy method. Although algebraic methods usually provide good approximation of the optimal value as well as the global minimizer, the computation cost is huge.

Over the past two decades, various relaxation methods have been studied extensively and intensively. Among them, semidefinite programming (SDP) and sum of squares (SOS) relaxations are very popular. Theoretically, SDP relaxation method is very powerful and successful in solving the general polynomial programming problem with a compact feasible region. However, the size of SDP relaxations to be solved increases rapidly as the size or the degree of the polynomial programming problem increases or higher accuracy is required. Indeed, SDP relaxations for the polynomial optimization can only be solved for small or moderately large problems, which severely affects their practical applications. Bigger problems would be solved if sparsity is exploited. To solve SOS relaxations of a polynomial programming problem, we need to convert them into conventional SDP relaxations. This is equivalent to solving some SDP problems, so efficient numerical methods to solve large scale SDP problems still remain a computational challenge.

In this thesis, we focus on both global optimality conditions and global optimization method to solve some classes of polynomial programming problems. It is well-known that traditional local optimization methods are designed according to Karush-Kuhn-Tucker (KKT) local optimality conditions. Motivated by this, some researchers proposed a necessary global optimality condition for a quadratic programming problem and designed a new local optimization method according to the necessary global optimality condition. Now we try to derive necessary global optimality conditions to cubic and quartic programming problems, and further to general unconstrained and constrained polynomial programming problems and then establish new local optimization methods according to these necessary conditions. The necessary global optimality conditions are generally stronger than KKT conditions. Hence, the obtained new local minimizers may improve some KKT points. However, the difficulty is still there - how to escape from a new local minimizer to a global one. The filled function method is one of the well-known and practical auxiliary function methods to settle this difficulty. So, we design global optimization methods to solve these polynomial programming problems by combining the new local optimization methods and some auxiliary functions. The numerical examples illustrate the efficiency and stability of the optimization methods.

Finally, we discuss some applications for solving some sensor network localization problems and systems of polynomial equations. The results illustrate our optimization methods are efficient and stable. It is worth mentioning that we apply the idea and the results for polynomial programming problems to nonlinear program problems (NLP). We provide an optimality condition and design local and global optimization methods for the problem (NLP). In order to test the performance of the global optimization methods, we compare them with two other heuristic methods. The results demonstrate our methods outperform the two other algorithms.

Outline of the thesis

The remainder of the thesis is organized as follows.

In Chapter 1, a literature review is given, including global optimization methods and local and global optimality conditions for nonlinear programming problems and polynomial programming problems.

In Chapter 2, we focus on cubic programming problems with mixed variables which are denoted by (MCP). For (MCP), we investigate some necessary local optimality conditions and some necessary global optimality conditions, which are very easy to check. We propose some new local optimization methods by using the proposed necessary local optimality conditions and the necessary global optimality conditions. A novel global optimization method is then proposed to solve problems (MCP) by combining these local optimization methods together with an auxiliary function. Some numerical examples are also presented to indicate the significance of our optimality conditions and show the efficiency of our optimization methods.

In Chapter 3, we consider quartic programming problems with box constraints which are denoted by (QPOP). We do not consider mixed variables because discrete variables are treated using the same procedure as we did for cubic problems with mixed variables. For (QPOP), we discuss a necessary global optimality condition by using some linear transformations. We then present a new local optimization method based on this necessary global optimality condition, which may improve some KKT points. Finally, we design a global optimization method to solve (QPOP) by combining the new local optimization method and an auxiliary function. Numerical examples illustrate the efficiency of the optimization methods.

After building up knowledge from cubic and quartic programming problems, in Chapter 4, we concentrate on general polynomial programming problems which are denoted by (GP). We try to provide a necessary global optimality condition for the problem (GP) by using some properties of univariate polynomial functions. A new local optimization method is designed for the problem (GP) according to the necessary global optimality condition, which may improve some KKT points. Finally, we design a global optimization method to solve the problem (GP) by combining the new local optimization method and an auxiliary function.

In Chapter 5, we are concerned with general constrained polynomial programming problems which are denoted by (GPP). A global necessary optimality condition for the problem (GPP) is considered. We design a new local optimization method based on the necessary global optimality condition and design a global optimization method by combining the new local optimization method and an auxiliary function. We investigate the efficiency and stability of our optimization methods.

In Chapter 6, we discuss some applications for solving some sensor network localization problems and systems of polynomial equations. In particular, we apply the idea and the results for polynomial programming problems to nonlinear programming problems (NLP). We provide an optimality condition for (NLP). We design two new local optimization methods and two global optimization methods (GOMs). The performance of GOMs is tested by comparing them with two other heuristic methods: simulated annealing heuristic pattern search (SAHPS) and quasi-filled function method (QFFM). The results demonstrate GOMs outperform two other algorithms and the proposed new local optimization methods are significant improvement of the traditional local optimization methods.

Chapter 1.

Literature review

The polynomial programming problem is the following generic optimization model

$$
\begin{aligned}
\min \quad & f(x) \\
s.t. \quad & g_i(x) \leq 0, i = 1, \cdots, m, \\
& h_j(x) = 0, j = 1, \cdots, l, \\
& x \in X \subset R^n
\end{aligned}
$$

where $f(x)$, $g_i(x)$ $(i = 1, \cdots, m)$ and $h_j(x)$ $(j = 1, \cdots, l)$ are some multivariate polynomial functions. X is a feasible set. Specifically, X is a box in this thesis.

Because of the inherent simplicity of the problem structure and rich modeling capabilities, the polynomial programming problem is a fundamental model in the field of optimization. The history of the polynomial programming problem might date back to the eighteenth century, when Monge formulated a continuous mass transportation problem as a huge assignment problem (a special polynomial programming problem) that minimizes the cost for transporting all the molecules [119]. Since the 19th century, researches have studied the relationship between nonnegative polynomial function and the sum of squares of polynomials.

In this chapter, we give an overview of global optimization methods and local and global optimality conditions for nonlinear programming problems and polynomial programming problems.

1.1. Global optimization methods

1.1.1. Global optimization methods for nonlinear programming problems

Traditionally, polynomial programming problems have been treated as a subclass of the general nonlinear programming problems, for which many methods have been put forward and many algorithms have been designed, including exact methods and heuristic methods. The exact methods have a rigorous guarantee for finding at least one global solution. However, it is difficult for the exact methods to handle larger dimensional models and more complicated models. For problems with higher dimensions or without special model structure, heuristics methods behave well in practice although they do not have strict convergence guarantees [105]. We will give a brief list of these methods below. For more details in the idea and applications, see [105].

1. Exact methods

   a) **Adaptive stochastic search methods** These methods are based on random sampling in a feasible set, see [2, 138].

   b) **Bayesian search algorithms** These methods are based on Bayesian networks to model promising solutions and bias the sampling of new candidate solutions, see [75, 98].

   c) **Branch and bound algorithms** These methods are based on a systematic enumeration of all candidate solutions. The fruitless candidates are discarded using upper and lower bounds, see [53, 85].

d) **Enumerative strategies** These methods are based on a complete enumeration of all possible solutions, see [113].

e) **Homotopy and trajectory methods** These methods are based on listing all stationary points of the objective function within the feasible set, see [42, 55].

f) **Integral methods** These methods are based on determination of the essential supremum of the objective function over the feasible set by approximating the level sets of the objective function, see [74, 109].

g) **'Naive' (passive) approaches** These methods are based on a simultaneous grid search and a pure random search, see [2, 71].

h) **Relaxation (out approximation) strategies** These methods are based on a sequence of relaxed sub-problems which are easier to solve, see [52, 113].

2. Heuristic methods

a) **Approximate convex underestimation** These methods are based on directed sampling in the feasible set to estimate the convexity characteristics of the objective function, see [84].

b) **Continuation methods** These methods are based on transforming the objective function into some more simpler function and then using a local minimization procedure to trace all minimizers back to the original function, see [73].

c) **Genetic algorithms, evolution strategies** These methods are based on four phases: evaluation, selection, recombination and mutation, see [56, 72].

d) **'Globalized' extensions of local search methods** These methods are based on a preliminary glboal search phase, followed by local scope search. [2, 71].

e) **Sequential improvement of local optima** These methods are based on searching for gradually better optima by constructed auxiliary functions, which include tunneling, deflation and filled function methods, see [13, 149].

f) **Simulated annealing** These methods are based on the physical analogy of cooling crystal structures that spontaneously arrive at a stable configuration, characterized by - globally or locally- minimal potential energy, see [12, 56].

g) **Tabu search (TS)** These methods are based on memory structures to forbid search moves to points already visited, see [41, 56].

Among these methods, we are interested in the filled function methods which belong to sequential improvement of local optima methods. We will introduce filled function methods later.

1.1.2. Global optimization methods for polynomial programming problems

Over the years, there have been attempts at developing global optimization methods to solve polynomial programming problems, which include quadratic, cubic, quartic and 0-1 integer programming problems as special cases. Existing methods for solving polynomial programming problems include algebraic methods and various convex relaxation methods.

Algebraic algorithms were established early as a means of solving polynomial programming problems. They are used to find all the critical points and then compare the function values of the polynomial at these points. Existing methods include Grobner bases and Stetter-moller method [51, 135], Resultant method [59], eigenvalues of companion matrices [27] and Homotopy method [83, 123]. Although the algebraic methods usually provide good approximation of the optimal value as well as the global minimizer, the computation cost is huge [30].

Over the past two decades, various relaxation methods have been developed, which include a lift-and-project linear programming (LP) procedure for 0-1 integer linear programs

[36], the reformulation-linearization technique (RLT) [47, 48], a semidefinite programming (SDP) relaxation method [67, 81], the successive convex relaxation method (SCRM) for quadratic optimization problems [94, 95], second order cone programming (SOCP) relaxations for quadratic optimization problems [118] and sums of squares (SOS) relaxations for polynomial optimization problems [68, 77–80]. These methods share the following basic idea [93]:

1. Add (redundant) valid inequality constraints to a target optimization problem in the n-dimensional Euclidean space R^n.

2. Lift the problem with the additional inequality constraints in R^n to an equivalent optimization problem in a symmetric matrix space; the resulting problem is an LP with additional rank-1 and positive semidefinite constraints on its matrix variables.

3. Relax the rank-1 constraint (and positive semidefinite constraint in cases of the RLT and the lift-and-project LP procedure) so that the resulting feasible region is convex.

4. Project the relaxed lifted problem in the matrix space back to the original Euclidean space R^n.

Among these methods, SDP and SOS relaxation methods have been widely used.

In theory, the SDP method is very powerful and successful in solving the general polynomial programming problem with a compact feasible region. Its optimal value can be approximated within any accuracy by the sequence of SDP relaxations. However, the size of SDP relaxations to be solved increases very rapidly as the size or the degree of the polynomial programming problem increases or higher accuracy is required. Indeed, SDP relaxations themselves can only solve small or moderately large polynomial programming problems, which severely limits their practical applications.

The SOS method theoretically can solve any general polynomial programming problems to any given accuracy. However, to solve SOS relaxations of a polynomial programming problem, we need to convert them into conventional SDP relaxations. This is equivalent to solving some SDP problems.

It is known that practical solvability of SDP methods depends on their sizes. This motivated a number of researchers to propose new methods for solving large scale SDP relaxations, such as accelerated first order methods and second order methods [81,110]. However, as the authors in [81] mentioned, so far there are few efficient numerical methods for solving large scale polynomial programming problems. In [81], regularization methods (RM) instead of interior point methods were applied to solve large scale SDP problems arising from general polynomial optimization. RM changed the linear semidefinite program into the equivalent convex semidefinite program by adding quadratic terms and then used the Newton-CG (conjugate gradient) Augmented Lagrangian regularization method to solve the original and dual problems. RM requires much less memory storage. Even though, this method may not extract the corresponding global minimizer from the global optimal function value. So, solving large scale SDP problems still remains a computational challenge.

As special cases of polynomial programming problems, quadratic, cubic and quatic programming problems have also been studied by many researchers. For quadratic programming problems, besides SDP and SOS relaxation methods, there are two other methods which are widely used: active-set methods [39, 70, 104] and interior-point methods [29, 37, 39]. Recent developed methods, which are closely related to this thesis, are that authors in [45, 153] present necessary global optimality conditions and design some new local optimization methods according to these conditions and design some global optimization methods by combining the new local optimization methods and some auxiliary functions. For cubic programming problems, [21] presented a specialization of the convex simplex method, the main idea of which is selecting a direction of improvement by observing the partial derivative and choosing an optimal step by minimizing the objective function in that direction. [25] converted indefinite cubic polynomial programming problems into convex optimization problems by some linear and homeomorphisms transformations. For quatic programming problems, [89] designed a global descent algorithm for normal quartic polynomials to find a global minimizer $(n = 2)$ or an ϵ-global minimizer $(n \geq 3)$. [139] presented a general semidefinite relaxation scheme for quartic homogeneous polynomial optimization under quadratic constraints by using a matrix listing transformation $X = xx^T$ to relax the quartic programming problem with quadratic constraints to a quadratic programming problem with linear constraints.

1.1.3. Filled function methods

The local optimization methods have been well developed and shown to be robust and reliable in finding a local optimal solution. However, the difficulty is how to leave a local minimizer to another lower one. The filled function method which belongs to the auxiliary function methods is one of the well-known and practical methods used to settle this difficulty. The filled function method includes two phases – local minimization and filling. These two phases are used alternately. In the first phase, starting from a given point, any local minimization method can be employed, such as the Quasi-Newton method and the Conjugate Gradient method. Using one of these methods, a local minimizer x_1 is found. After entering the second phase, an auxiliary function called a filled function is constructed based on the current local minimizer. The second phase ends when a point $x_1^* \neq x_1$ is found which satisfies $f(x_1^*) < f(x_1)$. Then the point x_1^* is regarded as a new starting point and the first phase is reentered and so on. The above process repeats until the time when minimizing a filled function does not yield a better solution. The current local minimum will be then taken as a global minimizer.

Filled function method for unconstrained programming problem

The filled function method was initially introduced by Ge in [111]. In [111], an unconstrained programming problem is considered. There are three assumptions:

1. The objective function is a twice continuously differentiable function $F(x)$ on R^n.

2. $F(x)$ satisfies the condition $F(x) \to +\infty$ as $\|x\| \to +\infty$.

3. $F(x)$ has only a finite number of minimizers.

By assumption 2, there exists a closed bounded domain $\Omega \subset R^n$ whose interior contains all global minimizers of $F(x)$. By assumption 3, every minimizer is therefore isolated.

Definition 1. *[111] The basin of $F(x)$ at an isolated minimizer x_1^* is a connected domain B_1^* which contains x_1^* and in which starting from any point the steepest descent trajectory of $F(x)$ converges to x_1^*, but outside which the steepest descent trajectory of $F(x)$ does not converge to x_1^*. Suppose $\hat{x}_1^*$ is a maximizer of $F(x)$. The hill of $F(x)$ at $\hat{x}_1^*$ is the basin of $-F(x)$ at its minimizer $\hat{x}_1^*$.*

Definition 2. *[111] A minimizer x_2^* of $F(x)$ is lower (or higher) than x_1^* iff*

$$F(x_2^*) \leq (or >) F(x_1^*) \tag{1.1}$$

and that the basin of $F(x)$ at x_2^*, B_2^* say, is lower (or higher) than B_1^* iff inequality (1.1) holds.

Definition 3. *[111] A function $P(x)$ is called a* filled function *of $F(x)$ at x_1^* if $P(x)$ has the following properties:*
(1) x_1^* is a maximizer of $P(x)$ and the whole basin B_1^* of $F(x)$ at x_1^* becomes a part of a hill of $P(x)$;
(2) $P(x)$ has no minimizers or saddle points in any higher basin of $F(x)$ than B_1^*;

(3) if $F(x)$ has a lower basin (at x) than B_1^*, then there is a point x' in such a basin that minimizes $P(x)$ on the line through x and x_1^*.

The filled function proposed in [111] is as follows:

$$P(x, r, \rho) = \frac{1}{r + F(x)} exp\left(-\frac{\|x - x_1^*\|^2}{\rho^2}\right)$$

where the parameters r and ρ need to be chosen appropriately. This filled function has some drawbacks, then many researchers devoted to this subject and proposed some other filled functions in references [22, 90, 106, 112, 132–134, 141, 149].

In [149], Wu et al. proposed two new kinds of modified functions: a new filled function and a quasi-filled function. There are also three assumptions:

1. The objective function is a continuously differentiable function $f(x)$ on R^n.
2. $f(x)$ satisfies the condition $f(x) \to +\infty$ as $\|x\| \to +\infty$.
3. Let Y be the set of all local minimizers. The set F defined by $F = \{f(x)|x \in Y\}$ is a finite set.

Assumption 3 means only the number of local minimal values is finite instead of the number of local minimizers.

Let x^* be a local minimizer and let L be the set which consists of all the local minimizers lower than x^*. A new definition of filled function is proposed.

Definition 4. *[149] A differentiable function $p(x)$ is a filled function corresponding to a local minimizer x^* if it satisfies the following properties:*
(1) x^* is a strictly local maximizer of $p(x)$;
(2) For any $x \neq x^*$ satisfying $f(x) \geq f(x^*)$, x is not a stationary point $p(x)$, i.e., $\nabla p(x) \neq 0$;
(3) if x^* is not a global minimizer, i.e., $L \neq \emptyset$, then for any $\bar{x} \in L$, $\bar{x}$ is a local minimizer of

$p(x)$ *and furthermore satisfies*

$$p(\bar{x}) < p(x^*)$$
$$p(\bar{x}) < p(x),\ for\ any\ x \in \partial\Omega.$$

where $\partial\Omega$ *denotes the boundary of* Ω.

(4) For any $x_1, x_2 \in \Omega$ *satisfying* $f(x_1) \geq f(x^*)$ *and* $f(x_2) \geq f(x^*)$, $\|x_2 - x^*\| > (\geq)\|x_1 - x^*\|$ *if and only if* $p(x_2) < (\leq)p(x_1)$.

Based on the new definition, a new filled function is proposed as:

$$H_{q,r,x^*}(x) = q(\exp(-\frac{||x - x^*||^2}{q})g_r(f(x) - f(x^*)) + f_r(f(x) - f(x^*))$$

where $r > 0$, $q > 0$ are parameters, x^* is the current local minimum, and for any $r > 0$, g_r and f_r are defined as:

$$g_r(t) = \begin{cases} 1, & t > 0 \\ -\frac{2}{r^3}t^3 - \frac{3}{r^2}t^2 + 1, & -r < t \leq 0 \\ 0, & t \leq -r \end{cases}$$

and

$$f_r(t) = \begin{cases} t + r, & t \leq -r \\ \frac{r-2}{r^3}t^3 + \frac{r-3}{r^2}t^2 + 1, & -r < t \leq 0 \\ 1, & t > 0 \end{cases}.$$

However, the local minimizer of the filled function will very easily go to the boundary of Ω. Another filled function called quasi-filled function was proposed, which local minimizer on

Ω must be in the interior of Ω. The quasi-filled function is

$$F_{q,r,c,x_0^*}(x) = q\left(exp(-\frac{\|x - x_0^*\|^2}{q})g_{r,c}\Big(f(x) - f(x_0^*)\Big) + h_{r,c}\Big(f(x) - f(x_0^*)\Big)\right). \quad (1.2)$$

where for any $r > 0$ and given $c > 0$,

$$g_{r,c}(t) = \begin{cases} c, & t \geq 0 \\ -\frac{2c}{r^3}t^3 - \frac{3c}{r^2}t^2 + c, & -r < t \leq 0 \\ 0, & t \leq -r \end{cases} \quad (1.3)$$

and

$$h_{r,c}(t) = \begin{cases} t + r, & t \leq -r \\ \frac{r-2}{r^3}t^3 + \frac{r-3}{r^2}t^2 + 1, & -r < t \leq 0 \\ 1, & 0 < t \leq 1 \\ -\frac{4c-2}{r^3}t^3 + \frac{(6c-3)(r+2)}{r^3}t^2 & \\ -\frac{(6c-3)(2+2r)}{r^3}t + \frac{4c-2+(6c-3)r}{r^3} + 1, & 1 \leq t \leq 1 + r \\ 2c & t > 1 + r \end{cases}. \quad (1.4)$$

In reference [149], the properties of function $F_{q,r,c,x^*}(x)$ are discussed as follows.

1. If x^* is a local minimizer of original problem, then for any $r > 0$, $q > 0$, $c > 0$, x^* is a strictly local maximizer of $F_{q,r,c,x^*}(x)$ on S.

2. For any $r > 0$, $q > 0$ and $c > 0$, if $x \in S$ and $x \neq x^*$ satisfies $0 \leq f(x) - f(x^*) \leq 1$ or $f(x) - f(x^*) \geq 1 + r$, then x is not a stationary point of $F_{q,r,c,x^*}(x)$. Otherwise, if x is a stationary point of $f(x)$, then x is not a stationary point of $F_{q,r,c,x^*}(x)$. And $\nabla F_{q,r,c,x^*}(x)(x - x^*) < 0$ for any x satisfying the above conditions.

3. If x^* is not a global minimizer of original problem. Let

$$L = \{\bar{x} \mid \bar{x} \text{ is the local minimizer of original problem satisfying } f(\bar{x}) < f(x^*)\}.$$

Then $L \neq \emptyset$. For any $\bar{x} \in L$, when $r \leq \frac{\beta_0}{2}$, $\bar{x}$ is a local minimizer of $F_{q,r,c,x^*}(x)$ and satisfies

$$F_{q,r,c,x^*}(\bar{x}) < F_{q,r,c,x^*}(x^*), F_{q,r,c,x^*}(\bar{x}) < F_{q,r,c,x^*}(x) \ for \ any \ x \in \partial S,$$

where $\beta_0 = \underset{y_1, y_2 \in F, y_1 \neq y_2}{min} |y_1 - y_2|$ (F is the set of value functions of all local minimizers of original problem) and ∂S is the boundary of S. Obviously, $\bar{x}$ is a stationary point of $F_{q,r,c,x^*}(x)$.

4. For any x_0 satisfying $f(x_0) - f(x^*) \leq 1$, the local minimizer $\bar{x}$ of function $F_{q,r,c,x^*}(x)$ over S starting from x_0 is in the interior of S when r and c satisfy the following conditions, respectively. $r \leq f_0 - 1$ and $c \geq 1$, where f_0 satisfies that there exist a point $x_1^0 \in S$ and a constant $f_0 > 1$ such that $f(x) \geq f(x_1^0) + f_0$ for any $x \in \partial S$.

In [45], authors proposed another filled function which is designed for solving mixed integer programming problems:

$$F_{r,x^*}(x) = \frac{1}{\|x - x^*\|^2 + 1} g_r\Big(f(x) - f(x^*)\Big) + f_r\Big(f(x) - f(x^*)\Big), \tag{1.5}$$

where $r > 0$ is a parameter, x^* is the current local minimizer and for any $r > 0$,

$$g_r(t) \ = \ \begin{cases} 1, & t > 0 \\ -\dfrac{2}{r^3}t^3 - \dfrac{3}{r^2}t^2 + 1, & -r < t \leq 0 \\ 0, & t \leq -r \end{cases} , \tag{1.6}$$

$$f_r(t) = \begin{cases} t+r & t \le -r \\ \dfrac{r-2}{r^3}t^3 + \dfrac{r-3}{r^2}t^2 + 1, & -r < t \le 0 \\ 1 & t > 0 \end{cases} . \tag{1.7}$$

In reference [45], the properties of this auxiliary function are discussed as follows.

1. Suppose that x^* is a local minimizer of original problem, then x^* is a strictly local maximizer of $F_{r,x^*}(x)$ on S for any $r > 0$.

2. Let $\bar{x}$ be the global minimizer of the original problem and let

$$\beta = f(x^*) - f(\bar{x}).$$

   If x^* is not a global minimizer of the original problem, i.e., $\beta > 0$, then $\bar{x}$ is a local minimizer of $F_{r,x^*}(x)$ on S when $r \le \beta$.

3. Any K-K-T point $\widehat{x}$ (see Definition 3.3 in [45] for the definition of K-K-T point) of $F_{r,x^*}(x)$ on S satisfies one of the following conditions:

   1°. $f(\widehat{x}) < f(x^*)$;

   2°. $\widehat{x} := (\widehat{x}_1, \ldots, \widehat{x}_n)^T$ satisfies that $\widehat{x}_i = \begin{cases} u_i \text{ or } v_i, & i \in M_{\bar{x}} \\ u_i + v_i - \bar{x}_i, & \text{otherwise} \end{cases}.$

In particular, [148] and [151] proposed two filled function methods to solve the following systems of nonlinear equations.

$$(SNE) \qquad h_i(x) = 0,\ i = 1, 2, \cdots, m$$
$$x \in X$$

$h_i(x)$, $i = 1, 2, \cdots, m$ are continuously differentiable nonlinear equations and X is a box.

We know that solving (SNE) is equivalent to solving the following optimization problem:

$$(OP) \qquad \begin{aligned} & min\ f(x) := \frac{1}{2}\sum_{i=1}^{m} h_i^2(x), \\ & x \in X. \end{aligned}$$

Next, we will introduce the filled function method provided in [148] which is under the following assumption.

Assumption 1. *[148] $f(x)$ satisfies the coercivity condition, i.e.* $\lim\limits_{\|x\|\to+\infty} f(x) = +\infty$.

Let $x_0 \in R^n$. By Assumption 1, there exists a box X such that

$$x_0 \in X \text{ and } f(x) \geq 2f(x_0) \text{ for any } x \in R^n \setminus \mathrm{int}X, \tag{1.8}$$

where $\mathrm{int}X$ denotes the interior of X.

To solve problem (OP), [148] present a new auxiliary function which can be a filled function, a quasi-filled function or a strict filled function with appropriately chosen parameters.

We give the definitions of these three functions as follows.

Definition 5. *[151] Let $\bar{x}_0 \in X$ satisfy $\bar{x}_0 \neq x^*$ and $f(\bar{x}_0) \leq \frac{5f(x^*)}{4}$. A continuously differentiable function $P_{x^*}(x)$ is said to be a filled function of problem (1.8) at x^* with $f(x^*) > 0$, if:*

1° x^* *is a strict local maximizer of $P_{x^*}(x)$ on X;*

2° *Any local minimizer $\bar{x}$ of $P_{x^*}(x)$ on X starting from $\bar{x}_0$ satisfies*

$$f(\bar{x}) < \frac{f(x^*)}{2} \quad \textit{or} \quad \bar{x} \textit{ is a vertex of } X;$$

3° *Any $\widetilde{x} \in X$ with $\nabla P_{x^*}(\widetilde{x}) = 0$ satisfies $f(\widetilde{x}) < \frac{f(x^*)}{2}$;*

4° *Any local minimizer $\widehat{x}$ of $f(x)$ on X with $f(\widehat{x}) \leq \frac{f(x^*)}{4}$ is a local minimizer of $P_{x^*}(x)$ on X.*

Definition 6. *[148] Let $\bar{x}_0 \in X$ satisfy $\bar{x}_0 \neq x^*$ and $f(\bar{x}_0) \leq \frac{5f(x^*)}{4}$, let $f(x)$ be differentiable on X. A continuous function $P_{x^*}(x)$ is said to be a quasi-filled function of problem (1.8) at x^* with $f(x^*) > 0$, if:*

1° x^* is a strict local maximizer of $P_{x^*}(x)$ on X;

2° Any local minimizer $\bar{x}$ of $P_{x^*}(x)$ on X starting from $\bar{x}_0$ satisfies $\bar{x} \in \mathrm{int}X$ and one of the following results holds:

$$
\begin{aligned}
&(1) \quad f(\bar{x}) \leq \frac{f(x^*)}{2}, \\
&(2) \quad \frac{3f(x^*)}{2} \leq f(\bar{x}) \leq \frac{7f(x^*)}{4} \text{ and } \nabla f(\bar{x}) \neq 0;
\end{aligned}
$$

3° Any local minimizer $\widehat{x}$ of problem $f(x)$ on X with $f(\widehat{x}) \leq \frac{f(x^*)}{4}$ is a local minimizer of $P_{x^*}(x)$ on X.

Definition 7. *[148] Let $\bar{x}_0 \in X$ satisfy $\bar{x}_0 \neq x^*$ and $f(\bar{x}_0) \leq \frac{5f(x^*)}{4}$. A continuous function $P_{x^*}(x)$ is said to be a strict filled function of problem (1.8) at x^* with $f(x^*) > 0$, if:*

1° x^* is a strict local maximizer of $P_{x^*}(x)$ on X;

2° Any local minimizer $\bar{x}$ of $P_{x^*}(x)$ on X starting from $\bar{x}_0$ satisfies

$$f(\bar{x}) < \frac{f(x^*)}{2}.$$

3° Any local minimizer $\widehat{x}$ of function $f(x)$ on X with $f(\widehat{x}) \leq \frac{f(x^*)}{4}$ is a local minimizer of $P_{x^*}(x)$ on X.

In the following, we will introduce an auxiliary function. Let

$$
\begin{aligned}
& G_{q,x^*}(x) \\
= \;& \exp(-\|x - x^*\|^2) g_{\frac{f(x^*)}{4}}\Big(f(x) - \frac{f(x^*)}{2}\Big) + q h_{\frac{f(x^*)}{4}, f(x^*)}\Big(f(x) - \frac{f(x^*)}{2}\Big),
\end{aligned}
\tag{1.9}
$$

where $q > 0$ is a parameter and

$$g_r(t) = \begin{cases} 1 & t \geq 0 \\ -\frac{2}{r^3}t^3 - \frac{3}{r^2}t^2 + 1 & -r < t \leq 0 \\ 0 & t \leq -r \end{cases} \tag{1.10}$$

and

$$h_{r,c}(t) = \begin{cases} t + r & t \leq -r \\ \frac{r-2}{r^3}t^3 + \frac{r-3}{r^2}t^2 + 1, & -r < t < 0 \\ 1 & 0 \leq t \leq c \\ -\frac{2}{r^3}t^3 + \frac{(6c+3r)}{r^3}t^2 - \\ \frac{(6cr+6c^2)}{r^3}t + \frac{3c^2r+2c^3}{r^3} + 1 & c < t < c + r \\ 2 & t \geq c + r \end{cases} . \tag{1.11}$$

Consider the following box-constrained optimization problem:

$$\min_{x \in X} \; G_{q,x^*}(x). \tag{1.12}$$

We have the following properties.

1. Let $f(x^*) > 0$. Then for any $q > 0$, x^* is a strict local maximizer of problem (1.12).

2. Assume that f is continuously differentiable on X and Assumption 1 holds. Let x^* satisfy $0 < f(x^*) \leq f(x_0)$ (x_0 satisfies (1.8)) and $\bar{x}_0 \neq x^*$ be a point such that $f(\bar{x}_0) - f(x^*) \leq \frac{f(x^*)}{4}$. Then,

   1°. there exists $q^1_{x^*} \geq 0$ such that when $q > q^1_{x^*}$, any local minimizer $\bar{x}$ of problem (1.12) obtained by search starting from $\bar{x}_0$ satisfies $\bar{x} \in \text{int}X$;

$2°$. there exists $q^2_{x^*} > 0$ such that when $0 < q < q^2_{x^*}$, any stationary point $\widetilde{x} \in X$ with $\widetilde{x} \neq x^*$ of function $G_{q,x^*}(x)$ satisfies $f(\widetilde{x}) < \frac{f(x^*)}{2}$.

3. Let x^* satisfy $0 < f(x^*) \leq f(x_0)$ (x_0 satisfies (1.8)). Any local minimizer $\bar{x}$ of problem $f(x)$ on X with $f(\bar{x}) < \frac{f(x^*)}{4}$ is a local minimizer of problem (1.12). Specially, any solution of (NSE) must be a local minimizer of problem (1.12).

Filled function method for constrained programming problems

Wenxing Zhu presented a class of filled functions and a class of globally concavized filled functions for box constrained continuous global optimization in the references [130] and [131], respectively. (P) is the original problem with box constraints and (AP) is the auxiliary problem, in which, the objective function is the filled function defined as follows. In [130], the definition of a filled function is presented as follows:

Definition 8. *The function $p(x)$ is called a filled function of problem (P) at its minimizer x_1^* if $p(x)$ is a continuously differentiable function and has the following properties:*

1. *Problem (AP) has no Kuhn-Tucker point in the region $S_1 = \{x \in X : f(x) \geq f(x_1^*)\}$ except a prefixed point $x_0 \in S_1$ that is a minimizer of $p(x)$.*

2. *Problem (AP) does have a minimizer in the region $S_2 = \{x \in X : f(x) < f(x_1^*)\}$ if $S_2 \neq \Phi$.*

where a Kuhn-Tucker point of problem (AP) is a point $y \in X$ which satisfies the following necessary conditions:

$$\begin{aligned} \frac{\partial p(y)}{\partial x_i} &\geq 0, \quad\quad y_i = l_i; \\ \frac{\partial p(y)}{\partial x_i} &\leq 0, \quad\quad y_i = u_i; \\ \frac{\partial p(y)}{\partial x_i} &= 0, \quad l_i < y_i < u_i. \end{aligned}$$

Under three assumptions of $u(x)$ and $v(x)$, five simple filled functions are presented. For the details of these assumptions, see [130].

$$
\begin{aligned}
p(x) &= u(x) - Av(x); \\
p(x) &= u(x) - ln(1 + Av(x)); \\
p(x) &= u(x) - p \cdot sin(Av(x)), where\ p\ is\ a\ constant\ and\ p > max_{x \in X} u(x); \\
p(x) &= u(x) - p \cdot arctg(Av(x)), where\ p\ is\ a\ constant\ and\ p > \frac{max_{x \in X} u(x)}{\pi/2}; \\
p(x) &= u(x) - p \cdot (1 - e^{-Av(x)}), where\ p\ is\ a\ constant\ and\ p > max_{x \in X} u(x).
\end{aligned}
$$

In [131], the definition of a globally concavized filled function is presented as follows:

Definition 9. *The function $p(x)$ is called a globally concavized filled function of problem (P) at its minimizer x_1^* if $p(x)$ is a continuously differentiable function and has the following properties:*

1. *x_1^* is a maximizer of problem (AP).*
2. *All minimizers or stationary points of Problem (AP) in set $S_1 = \{x \in X : f(x) \geq f(x_1^*)\}$, except x_1^*, are on the boundary of the bounded closed box X.*
3. *Problem (AP) does have a minimizer in the set $S_2 = \{x \in X : f(x) < f(x_1^*)\}$ if $S_2 \neq \Phi$.*

where a stationary of problem (AP) is defined as the same as the Kuhn-Tucker point of problem (AP) in [130].

Two globally concavized filled functions are presented

$$
p(x, A, h) = \frac{1}{\|x - x_1^*\| + c} arctan(A[f(x) - f(x_1^*) + h]);
$$
$$
p(x, A, h) = \frac{1}{\|x - x_1^*\| + c} tanh(A[f(x) - f(x_1^*) + h]).
$$

where the two parameters A is large enough and h is small enough.

Furthermore, Wu et al. proposed a filled function method for inequality constrained global optimization problems in [146].

$$
\begin{aligned}
(P) \quad min \quad & f(x) \\
s.t. \quad & g_i(x) \leq 0, i = 1, \cdots, m, \\
& x \in X
\end{aligned}
$$

where $f : X \to R$, $g_i : X \to R$, $i = 1, \cdots, m$ and X is a box. The filled function is presented as:

$$
p_{r,c,q,x^*}(x) = \frac{1}{\|x - x^*\|^2 + 1} f_{r,c}\left(g_r(f(x) - f(x^*)) + \sum_{i=1}^{m} g_{\frac{r}{q}}(g_i(x)) - 2r \right),
$$

where $c > 0$, $r > 0$ and $q > 0$ are parameters, x^* is the current local minimum, and :

$$
f_{r,c}(t) = \begin{cases} c, & t \geq 0 \\ -\frac{2c}{r^3}t^3 - \frac{3c}{r^2}t^2 + c, & -r < t \leq 0 \\ 0, & t \leq -r \end{cases}
$$

and

$$
g_r(t) = \begin{cases} t + 2, & t \geq 0 \\ \frac{r-4}{r^3}t^3 + \frac{2r-6}{r^2}t^2 + t + 2, & -r < t < 0 \\ 0, & t \leq -r \end{cases}.
$$

Recently, Wu et al. proposed a new filled function method for general constrained global optimization problems in [147].

$$
\begin{aligned}
(P) \quad min \quad & f(x) \\
s.t. \quad & g_i(x) \leq 0, i = 1, \cdots, m, \\
& h_j(x) = 0, j = 1, \cdots, l, \\
& x \in X
\end{aligned}
$$

where $f : X \to R$, $g_i, h_j : X \to R$, $i = 1, \cdots, m$, $j = 1, \cdots, l$ are continuously differentiable on X, and X is an open box.

In [147], first, an auxiliary function is employed to find an $\epsilon-$approximate feasible solution via locally solving a smooth unconstrained optimization problem, where ϵ is any preset positive number. Then a filled function is constructed to search for an approximate global minimizer of problem P.

The filled function is presented as

$$
\begin{aligned}
& F_{r,x_r^*}(x) = \\
& \frac{1}{\|x - x_r^*\|^2 + 1}\phi\left(\psi_{\frac{r}{2}}(f(x) - f(x_r^*) + \frac{r}{2}) + \sum_{i=1}^{m}\psi_{\frac{r}{2}}(g_i(x) - \frac{r}{2}) + \sum_{j=1}^{l}\psi_{\frac{3r^2}{4}}(h_j^2(x) - \frac{r^2}{4})\right)
\end{aligned}
$$

where $r > 0$ is a parameter and

$$
\psi_r(t) = \begin{cases} \frac{2}{r}t - 1, & t \geq r \\ \frac{(t-r)^2}{r^2} + \frac{2}{r}t - 1, & 0 < t < r \\ 0, & t \leq 0 \end{cases}
$$

and

$$\phi(t) = \begin{cases} 1, & t \geq 1 \\ -2t^3 + 3t^2, & 0 < t < 1 \\ 0, & t \leq 0 \end{cases}.$$

Since the filled function methods only employ extensively improved local optimization algorithms, these methods have been attracting much attention by more and more researchers. However, when it comes to the behavior of a filled function, it depends directly on the construction of the filled function. Hence, many researchers still devote to revise or present new filled functions.

1.2. Optimality conditions

Necessary global optimality conditions are efficient tools to prove that a given point is not an optimal solution and sufficient global optimality conditions are strong tools to check that a given point is an optimal solution. Without these global optimality conditions, most algorithms cannot stop properly. Much attention has been devoted to the development of global optimality conditions.

1.2.1. Optimality conditions for nonlinear programming problems

For optimality conditions of nonlinear programming problems, most literature focuses on special models, such as generalized convex programming problems [60, 115, 121] and nonconvex problems involving directionally differentiable functions [114]. Since Karush-Kuhn-Tucker (KKT) optimality conditions are also sufficient for optimality if the functions involved in the mathematical programming problems are convex, generalized convex functions received more attention later [60]. Researchers tried to solve this question: under what assumptions, are the KKT conditions also sufficient for the various generalizations of convex problems? [115] defined semilocally quasiconvex and semilocally pseudoconvex functions and obtained sufficient optimality conditions for a class of nonlinear programming problems involving such functions. [60] considered a nonlinear programming problem where the functions involved are $\eta-$semidifferentiable and presented KKT necessary optimality conditions and sufficient optimality conditions. [121] introduced a new class nonconvex functions called G-invex functions and provided some necessary conditions and sufficient conditions. [114] studied optimality conditions for nonconvex problems involving a class of directionally differentiable functions and generalized the necessary and sufficient optimality conditions by using the weak subgradient notion. More generally, although [126] developed necessary global optimality conditions for nonlinear programming problems with polynomial constraints, as it mentioned, the conditions are difficult to check for general large dimensional problems since the conditions involve in solving a sequence of semidefinite programs.

1.2.2. Optimality conditions for polynomial programming problems

The polynomial programming problem as a special case of nonlinear and nonconvex programming problems attracts a lot of attention. Besides development of various global optimization methods to solve it, a number of global optimality conditions appear in literature. At the early stage, the global optimality conditions focus on quadratic programming problems. References [3, 43, 45, 57, 58, 64–66, 76, 91, 124, 125, 152] present various global optimality conditions for the problems with quadratic objective function subject to different constraints, such as box constraints, binary constraints, quadratic constraints, linear constraints and mixed variables. In particular, we mention that the global optimality conditions introduced in [9, 82, 142, 143] and [145] are based on abstract convexity. They are expressed in terms of abstract subdifferential (L-subdifferential) and abstract normal cone (L-normal cone).

L**-Subdifferential** [8]. Let $f : R^n \to R$ and $x_0 \in \operatorname{dom} f$. An element $l \in L$ is called an L-subgradient of f at a point $x_0 \in R^n$ if $f(x) \geq f(x_0) + l(x) - l(x_0),\ \forall x \in R^n$. The set $\partial_L f(x)$ of all L-subgradients of f at x_0 is referred to as $L-$subdifferential of f at x_0.

L**-normal Cone** [8]. For a set $D \subset R^n$ and $x_0 \in D$, the *normal cone* of D at x_0 with respect to L, called as L-normal cone, is given by $N_{L,D}(x_0) := \{l \in L : l(x) - l(x_0) \leq 0 \text{ for each } x \in D\}$.

Furthermore, [136] discussed some global optimality conditions for a special kind of cubic polynomial optimization problems where the cubic objective function contains no third order cross terms. [82] presented sufficient global optimality conditions and necessary global optimality conditions for some classes of polynomial integer programming problems where the objective function contains no cross terms for more than the second order.

For the general polynomial programming problem, [127] presented global optimality conditions for polynomial optimization over box or bivalent constraints by using separable polynomial relaxations. However, We notice that it is not easy to decompose a polynomial function to the sum of a separable polynomial function and an SOS-convex polynomial function. Based on the so-called Positivstellensatz (a polynomial analogue of the transposition theorem for linear systems), it is possible to formulate global necessary and sufficient conditions for general polynomial programming problems with polynomial constraints (GPP) [54]. [67] proved in Theorem 4.2 a sufficient condition for global optimality in (GPP), which is a special case of the global necessary and sufficient condition presented in [54]. [126] provided another necessary and sufficient global optimality condition for (GPP). However, all these conditions are complex and difficult to check in practice since the conditions involve solving a sequence of semidefinite programs. Only under the idealized assumptions that all semidefinite programs can be solved exactly, it is possible for these conditions to be checked [54].

It is well-known that traditional local optimization methods are designed based on KKT conditions. Motivated by this, [45] focused on both global optimality conditions and global optimization methods for mixed integer quadratic programming problems (MIQP). A necessary global optimality condition and a sufficient global optimality condition were proposed. A local optimization method was designed by using the necessary global optimality condition and a global optimization method was designed by combining the sufficient global optimality condition, an auxiliary function and the obtained local optimization method. In next section, let us review the global optimality conditions and local and global optimization methods provided in [45].

1.2.3. Local and global optimality conditions for a mixed integer quadratic programming problem

[45] considered the following mixed integer quadratic model programming problem:

$$
\begin{aligned}
(MIQP) \quad min \quad & \frac{1}{2}x^T A x + a^T x \\
s.t. \quad & x \in U = \left\{ (x_1, \cdots, x_n)^T \;\middle|\; \begin{array}{ll} x_i \in \{u_i, u_{i+1}, \cdots, v_i\}, & i \in I \\ x_i \in [u_i, v_i], & i \in J \end{array} \right\}
\end{aligned}
$$

where $a \in R^n$, $A \in S^n$ and S^n is the set of all symmetric $n \times n$ matrices, $u_i < v_i$, $\forall i = 1, \cdots, n$ and u_i, v_i, $\forall i \in I$ are integers in R, $I, J \subseteq \{1, \cdots, n\}$, $I \bigcap J = \emptyset$ and $I \bigcup J = \{1, \cdots, n\}$. For $\bar{x} \in U$, let

$$
\tilde{\bar{x}}_i := \begin{cases} -1, & \text{if } \bar{x}_i = u_i \\ 1, & \text{if } \bar{x}_i = v_i \\ sign(a + A\bar{x})_i, & \text{if } \bar{x}_i \in (u_i, v_i) \end{cases}
$$

$$b_{\bar{x}_i} := \begin{cases} \tilde{\bar{x}}_i \frac{(a+A\bar{x})_i}{v_i - u_i}, & i \in J \\ max\left\{ \tilde{\bar{x}}_i (a + A\bar{x})_i, \tilde{\bar{x}}_i \frac{(a+A\bar{x})_i}{v_i - u_i} \right\}, & i \in I \end{cases}$$

$$b_{\bar{x}} = (b_{\bar{x}_1}, \cdots, b_{\bar{x}_n})^T$$

where

$$sign(a + A\bar{x})_i := \begin{cases} -1, & (a + A\bar{x})_i < 0 \\ 0, & (a + A\bar{x})_i = 0 \\ 1, & (a + A\bar{x})_i > 0 \end{cases}$$

For $Q = diag(q_1, \cdots, q_n)$ and $q_i \in R$, $i = 1, \cdots, n$, let

$$\tilde{q}_i = \begin{cases} min\{0, q_i\}, & i \in J \\ q_i, & i \in I \end{cases}$$

$$\tilde{Q} = diag(\tilde{q}_1, \cdots, \tilde{q}_n)$$

For $A = (a_{ij})_{n \times n}$, let

$$\tilde{a}_{ii} = \begin{cases} min\{0, a_{ii}\}, & i \in J \\ a_{ii}, & i \in I \end{cases}$$

$$diag(\tilde{A}) = diag(\tilde{a}_{11}, \cdots, \tilde{a}_{nn})$$

Theorem 1. *[45] (Sufficient global optimality condition for $(MIQP)$) Let $\bar{x} \in U$. If*

$$[SC] \begin{cases} b_{\bar{x}_i} \leq 0, \forall i \in J \\ diag(b_{\bar{x}}) \preceq \frac{1}{2}A \end{cases}$$

then $\bar{x}$ is a global minimizer of problem $(MIQP)$.

Theorem 2. *[45] (Necessary global optimality condition for $(MIQP)$) Let $\bar{x} \in U$. If $\bar{x}$ is a global minimizer of problem $(MIQP)$, then the following condition holds:*

$$[NC]\ diag(b_{\bar{x}}) \preceq \frac{1}{2}diag(\tilde{A})$$

The significance of this paper is to design a new local optimization method according to the necessary global optimality condition.

Let

$$N_i(\bar{x}) = \begin{cases} \{\bar{x} + (w_i - \bar{x}_i)e_i | w_i = u_i, u_{i+1}, \cdots, v_i\}, \forall i \in I \\ \{\bar{x} + (w_i - \bar{x}_i)e_i | w_i = u_i, v_i\}, \forall i \in J \end{cases}$$

where e_i is the ith unit vector (the n dimensional vector with the ith component equals to one and the other component equal to zero). The following algorithm was designed to solve (MIQP):

Algorithm 1. Local optimization method for (MIQP) (LOM_{MIQP})

Step 1. Take an initial point $x_0 \in U$. Let $\bar{x} = x_0$, $k := 1$.

Step 2. Check whether the following condition $[NC]_1$ holds:

$$[NC]_1\ \ b_{\bar{x}_i} \leq \frac{1}{2}a_{ii},\ \ \forall i = 1, \cdots, n.$$

If $[NC]_1$ does not hold, go to Step 3; otherwise, check whether the following condition

$[NC]_2$ holds:

$$[NC]_2 \;\; b_{\bar{x}_i} \leq 0, \;\; \forall i \in J.$$

If $[NC]_2$ holds, go to Step 5, else go to Step 4.

Step 3. Let $x^* = (x_1^*, \cdots, x_n^*)^T := argmin\{f(x)|x \in \bigcup_{i=1}^{n} N_i(\bar{x})\}$ and let $\bar{x} = x^*$, go to Step 2.

Step 4. Let $h(y) := f(\bar{x}_1, \cdots, \bar{x}_k, y_1, \cdots, y_{n-k})$, and let $y^* := (y_1^*, \cdots, y_{n-k}^*)^T$ be a local minimizer or a KKT point of $h(y)$ on $U_J = \prod_{i \in J} [u_i, v_i]$ starting from $(\bar{x}_{k+1}, \cdots, \bar{x}_n)^T$. Let $\bar{y} := (\bar{x}_1, \cdots, \bar{x}_k, y_1^*, \cdots, y_{n-k}^*)$ and let $\bar{x} = \bar{y}$, go to Step 3.

Step 5. Stop. $\bar{x}$ is a local minimizer of problem (MIQP).

[45] also designed a local optimization method (LOM_{MP}) which was used to solve the auxiliary function problem. For the details of the local optimization method (LOM_{MP}), see [45].

Next, [45] designed a global optimization method by combining the sufficient global optimality condition, the proposed local optimization method and an auxiliary function which is defined by (1.5).

Algorithm 2. Global optimization method for (MIQP) (GOM)

Step 0. Take an initial point $x_1 \in U$. a sufficiently small positive number μ, and an initial $r_1 > 0$. Set $k := 1$.

Step 1. Use the local minimization method (LOM_{MIQP}) to solve problem (MIQP) starting from x_k. Let x_k^* be the obtained local minimizer.

Step 2. Verify x_k^* whether satisfies the following global optimality sufficient conditions:

$$[SC]_k \begin{cases} (b_{x_k^*})_i \leq 0, \forall i \in J \\ diag(bx_k^*) \preceq \frac{A}{2} \end{cases}$$

If $[SC]_k$ holds, then go to Step 6; otherwise, let $r := r_1$ go to Step 3.

Step 3. Construct the following auxiliary function

$$F_{r,\bar{x}}(x) = \frac{1}{\|x - \bar{x}\|^2 + 1} g_r\Big(f(x) - f(\bar{x})\Big) + f_r\Big(f(x) - f(\bar{x})\Big),$$

Consider the following problem:

$$\begin{aligned} \min \quad & F_{r,x_k^*}(x) \\ s.t. \quad & x \in U. \end{aligned} \tag{1.13}$$

Let $\bar{x}_k := x_k^*$, go to Step 4.

Step 4. Use the local minimization method (LOM_{MP}) to solve problem (1.13) starting from $\bar{x}_k$. Let $\bar{x}_k^*$ be the local minimizer of problem (1.13). If $f(\bar{x}_k^*) < f(x_k^*)$, let $x_{k+1} := \bar{x}_k^*$, $k := k + 1$, go to Step 1; otherwise go to Step 5.

Step 5. If $r \geq \mu$, decrease r, such as, let $r := r/10$, go to Step 3; otherwise, go to Step 6.

Step 6. Stop and x_k^* is the obtained global minimizer.

Finally, some numerical examples illustrated the efficiency and stability of the local and global optimization methods.

In this thesis, we apply the idea and the results mentioned in [45] to cubic, quartic, and further to general unconstrained and constrained polynomial programming problems. We try to derive necessary global optimality conditions for these problems which are generally stronger than KKT conditions. Hence, the obtained new local minimizers may improve some KKT points.

Chapter 2.

Global optimality conditions and optimization methods for cubic programming problems with mixed variables (MCP)

Multivariate cubic polynomial programming problems, as special cases of the general polynomial optimization, have a lot of practical applications in real world. In this chapter, some necessary local optimality conditions and some necessary global optimality conditions for cubic polynomial programming problems with mixed variables are established. Then, some local optimization methods, including a weakly local optimization method for general problems with mixed variables and a strongly local optimization method for cubic polynomial programming problems with mixed variables, are proposed by exploiting these necessary local optimality conditions and necessary global optimality conditions. A global optimization method is proposed for cubic polynomial programming problems by combining these local optimization methods together with an auxiliary function. Some numerical examples are also given to illustrate that these approaches are very efficient.

2.1. Introduction

We consider cubic polynomial programming problems with mixed variables which are denoted by (MCP) in this chapter. Problems of the form (MCP) arise in many areas of applications, such as finance and agricultural researches [21]. Especially Hanoch and Levy [44] as well as Levy and Sarnat [49] have shown that Markowitz's model on portfolio selection [49] can be appropriately or perfectly described as a cubic utility function. More applications of cubic polynomial programming problems can be found in [120]. Problems (MCP) also cover quadratic programming problems with box or binary constraints; see [3, 124]. Moreover, we know that the problem (MCP) is NP-hard. In fact, even the binary quadratic problem is NP-hard [99]. As the cubic programming problem can be regarded as adding some third order monomials to quadratic optimization, (MCP) is also NP-hard. These motivate us to solve (MCP).

General polynomial programming problems can be solved by SDP or SOS relaxation methods [67–69, 77–80]. As we surveyed in Chapter 1, so far the most effective use of SDP relaxations has been for the quadratic programming problems [28, 77, 93, 139]. As special cases of polynomial programming problems, problems (MCP) have also been studied by many researchers. In [21], a specialization of the convex simplex method for cubic polynomial programming problems was presented, the main idea of which is selecting a direction of improvement by observing the partial derivative and choosing an optimal step by minimizing the objective function in that direction. Recently, [25] has converted indefinite cubic polynomial programming problems into convex optimization problems by some linear and homeomorphisms transformations.

We know that the necessary local optimality conditions are the main tools for the development of efficient numerical methods in local optimization. Although [126] provided a necessary and sufficient global optimality condition for general polynomial programming problems, as it mentioned, the condition is difficult to check for general large dimensional problems since the condition involves in solving a sequence of semidefinite programs. References [3, 20, 43, 57, 58, 64–66, 76, 91, 103, 124, 125, 152]focus on global optimality conditions for the problems with quadratic objective function subject to different constraints, such as box constraints, binary constraints, quadratic constraints, linear constraints and mixed variables. Recently, [45] established a new local optimization method for quadratic programming problems with mixed variables $(MIQP)$ by using the necessary global optimality condition. It also gave a new global optimization method for $(MIQP)$ by combining the new local optimization method, a sufficient global optimality condition together with an auxiliary function. Also, [136] discussed some global optimality conditions for a special kind of cubic polynomial optimization problems where the cubic objective function contains no third order cross terms. In this chapter, we will first investigate some necessary local optimality conditions and some necessary global optimality conditions for problems (MCP), which are very easy to check. Then, we will propose some new local optimization methods by using the proposed necessary local optimality conditions and the necessary global optimality conditions. A novel global optimization method is then proposed to solve problems (MCP) by combining these local optimization methods together with an auxiliary function. Some numerical examples are also presented to indicate the significance of our optimality conditions and show the efficiency of our optimization methods.

2.2. Necessary optimality conditions for (MCP)

Consider the following optimization of a multivariate third order (cubic) polynomial programming problem with mixed variables:

$$\text{(MCP)} \qquad \min \quad f(x) = \sum_{\substack{j,l,r=0 \\ l \geq j, r \geq l}}^{n} c_{j,l,r} x_j x_l x_r$$

$$s.t. \tag{2.1}$$

$$x_i \in [u_i, v_i], i = 1, \ldots, m, \; x_i \in \{u_i, v_i\}, i = m+1, \ldots, n,$$

where m is a nonnegative integer number, $x = (x_1, x_2, \cdots, x_n)^T \in R^n$, $x_0 \equiv 1$, u_i,v_i,$c_{j,l,r}$ $\in R$ and $u_i < v_i$ for any $i = 1, \ldots, n$, R^n is the n-dimensional Euclidean space and R is the real line.

In this section, we will derive some necessary optimality conditions including necessary local optimality conditions and necessary global optimality conditions for the problem (MCP). First, we present some notations that will be used throughout this chapter. For any $i = 1, \ldots, n$, let

$$\begin{aligned} S_i : &= \begin{cases} [u_i, v_i], & i = 1, \ldots, m, \\ \{u_i, v_i\}, & i = m+1, \ldots, n, \end{cases} \\ \bar{S}_i : &= [u_i, v_i], i = 1, \ldots, n, \end{aligned}$$

$$S : = \prod_{i=1}^{n} S_i, \tag{2.2}$$

$$\bar{S} : = \prod_{i=1}^{n} \bar{S}_i. \tag{2.3}$$

For giving some definitions, consider the following general mathematical optimization problem (P):

$$(P) \qquad \min \quad f(x) \qquad s.t. \qquad x \in S,$$

where $f(x)$ is continuous differentiable on $\bar{S}$, S and $\bar{S}$ are defined by (2.2) and (2.3), respectively. For any $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_n)^T \in S$, we denote

$$\begin{aligned}
N_i(\bar{x}) : &= \{\bar{x} + z_i e_i \mid z_i \in \{u_i - \bar{x}_i, v_i - \bar{x}_i\} \setminus \{0\}\}, \text{ for } i = 1, \ldots, n, \\
\delta_i(\bar{x}) : &= \begin{cases} \min\{v_i - \bar{x}_i, \bar{x}_i - u_i\}, & \text{if } \bar{x}_i \in (u_i, v_i) \\ v_i - u_i, & \text{otherwise} \end{cases}, \\
\delta(\bar{x}) : &= \min\{\delta_i(\bar{x}), i = 1, \ldots, m\},
\end{aligned} \tag{2.4}$$

where $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)^T$, the ith element is 1 and the others are 0. For any $i = 1, \ldots, m$ and for any $0 < \delta \leq \delta(\bar{x})$, denote

$$N_{i,\delta}(\bar{x}) : = \left\{ \bar{x} + \alpha e_i \left| \begin{array}{ll} \alpha \in (0, \delta), & \text{if } \bar{x}_i = u_i, \\ \alpha \in (-\delta, 0), & \text{if } \bar{x}_i = v_i, \\ \alpha \in (-\delta, \delta), & \text{if } \bar{x}_i \in (u_i, v_i) \end{array} \right. \right\}$$

and let

$$\hat{N}_\delta(\bar{x}) : = \{x = (x_1, \cdots, x_m, \bar{x}_{m+1}, \cdots, \bar{x}_n) \in S \big| \|x - \bar{x}\| < \delta\}, \tag{2.5}$$

$$N_\delta(\bar{x}) : = \hat{N}_\delta(\bar{x}) \cup_{i=1}^n N_i(\bar{x}) \cup \{\bar{x}\}.$$

Obviously, if $\delta \leq \delta(\bar{x})$, then $N_\delta(\bar{x}) \subset S$ and $|N_i(\bar{x})| \leq 2$ for $i = 1, \ldots, n$, where $|N_i(\bar{x})|$ means the number of the points in $N_i(\bar{x})$.

Definition 10. *Let $\bar{x} \in S$. For $\delta > 0$ such that $\delta \leq \delta(\bar{x})$, $N_\delta(\bar{x})$ is said to be a neighborhood of $\bar{x}$ with respect to S.*

Definition 11. *Let $\bar{x} \in S$. $\bar{x}$ is said to be a local minimizer of the problem (P) (local maximizer of $f(x)$ on S), iff there exists a positive number δ satisfying $\delta \leq \delta(\bar{x})$ such that*

$f(\bar{x}) \le f(x)$ ($f(\bar{x}) \ge f(x)$) for any $x \in N_\delta(\bar{x})$; furthermore, $\bar{x}$ is said to be a strictly local minimizer of the problem (P) (strictly local maximizer of $f(x)$ on S), iff $f(\bar{x}) < f(x)$ ($f(\bar{x}) > f(x)$) for any $x \in N_\delta(\bar{x}) \setminus \{\bar{x}\}$.

Definition 12. *Let $\bar{x} \in S$ and let $h(y) := f(y_1, \ldots, y_m, \bar{x}_{m+1}, \ldots, \bar{x}_n)$, where $y = (y_1, \ldots, y_m)^T \in \prod_{i=1}^{m}[u_i, v_i]$. $y^* = (y_1^*, \ldots, y_m^*)^T$ is said to be a traditional local minimizer of $h(y)$ on $\prod_{i=1}^{m}[u_i, v_i]$ iff there exists a positive number δ satisfying $\delta \le \delta(\bar{x})$ such that $h(y) \ge h(y^*)$ for any $y = (y_1, \ldots, y_m)^T \in \prod_{i=1}^{m}[u_i, v_i]$ satisfying $(y_1, \ldots, y_m, \bar{x}_{m+1}, \ldots, \bar{x}_n)^T \in \hat{N}_\delta(\bar{x})$, where $\hat{N}_\delta(\bar{x})$ is defined by (2.5).*

Definition 13. *Let $\bar{x} \in S$. $\bar{x}$ is said to be a global minimizer of the problem (P) iff $f(x) \ge f(\bar{x})$ for any $x \in S$.*

For $\bar{x} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)^T \in S$, and for any $i = 1, \ldots, n$, we define

$$
\begin{aligned}
m_{\bar{x}} &:= \{i \mid \bar{x}_i \in (u_i, v_i), i = 1, \ldots, m\}, \\
\widetilde{\widetilde{x}}_i &:= \begin{cases} -1, & \text{if } \bar{x}_i = u_i \\ 1, & \text{if } \bar{x}_i = v_i \\ \operatorname{sign}(\nabla f(\bar{x}))_i, & \text{if } u_i < \bar{x}_i < v_i \end{cases}, \\
b_{\bar{x}_i} &:= \widetilde{\widetilde{x}}_i (\nabla f(\bar{x}))_i, \\
b_{\bar{x}} &:= (b_{\bar{x}_1}, \ldots, b_{\bar{x}_n})^T, \\
\theta_{i,\bar{x}} &:= \begin{cases} \min\left\{ \begin{array}{c} c_{i,i,i}(u_i - \bar{x}_i) + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}, \\ c_{i,i,i}(v_i - \bar{x}_i) + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2} \end{array} \right\}, & i \in m_{\bar{x}} \\ -\widetilde{\widetilde{x}}_i c_{i,i,i}(v_i - u_i)^2 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(v_i - u_i), & \text{otherwise} \end{cases}, \qquad (2.6) \\
\theta_{\bar{x}} &:= (\theta_{1,\bar{x}}, \cdots, \theta_{n,\bar{x}})^T, \\
\eta_{i,\bar{x}} &:= \widetilde{\widetilde{x}}_i \frac{1}{16 c_{i,i,i}} [\frac{\partial^2 f(\bar{x})}{\partial x_i^2}]^2, \text{ for } c_{i,i,i} \neq 0, \qquad (2.7) \\
y_{i,\bar{x}} &:= \bar{x}_i - \frac{1}{4 c_{i,i,i}} \frac{\partial^2 f(\bar{x})}{\partial x_i^2}, \text{ for } c_{i,i,i} \neq 0, \qquad (2.8)
\end{aligned}
$$

$$\begin{aligned}\alpha_{i,\bar{x}}: &= \begin{cases}\eta_{i,\bar{x}}, & \text{if } \widetilde{\bar{x}}_i c_{i,i,i} < 0, y_{i,\bar{x}} \in (u_i, v_i) \text{ and } i \in \{1, \ldots, m\} \setminus m_{\bar{x}} \\ \theta_{i,\bar{x}}, & \text{otherwise },\end{cases} \qquad (2.9) \\ \alpha_{\bar{x}}: &= (\alpha_{1,\bar{x}}, \cdots, \alpha_{n,\bar{x}})^T,\end{aligned}$$

where $\mathrm{sign}(\nabla f(\bar{x}))_i := \begin{cases}-1, & (\nabla f(\bar{x}))_i < 0 \\ 0, & (\nabla f(\bar{x}))_i = 0 \\ 1, & (\nabla f(\bar{x}))_i > 0\end{cases}$.

In the following, we will first give a necessary local optimality condition for the problem (P) and the problem (MCP).

Theorem 3. **(Necessary local optimality condition for** (P)**)** *Let* $\bar{x} \in S$*. If* $\bar{x}$ *is a local minimizer of the problem* (P)*, then the following condition* $[LNCP]$ *holds:*

$$[LNCP] \qquad \begin{cases} b_{\bar{x}_i} \leq 0, & \forall i \in \{1, \ldots, m\} \\ f(\bar{x}) \leq f(x), & \forall x \in \cup_{i=1}^{n} N_i(\bar{x}).\end{cases}$$

Proof: By definition 11, we know that $\bar{x}$ is a local minimizer of the problem (P) if and only if there exists a positive number δ satisfying $\delta \leq \delta(\bar{x})$ such that $f(\bar{x}) \leq f(x)$ for any $x \in N_\delta(\bar{x})$. By $\cup_{i=1}^{n} N_i(\bar{x}) \subset N_\delta(\bar{x})$, we get that

$$f(\bar{x}) \leq \min\{f(x) \mid x \in \cup_{i=1}^{n} N_i(\bar{x})\}.$$

By $\cup_{i=1}^{m} N_{i,\delta}(\bar{x}) \subset N_\delta(\bar{x})$, we have that

$$f(x) \geq f(\bar{x}), \forall x \in \cup_{i=1}^{m} N_{i,\delta}(\bar{x}),$$

which implies that

$$b_{\bar{x}_i} = \widetilde{\bar{x}}_i (\nabla f(\bar{x}))_i \leq 0, \forall i \in \{1, \ldots, m\}.$$

In fact, for any $i = 1, \ldots, m$,

$$\begin{aligned} & f(x) \geq f(\bar{x}), \forall x \in N_{i,\delta}(\bar{x}) \\ \Rightarrow \quad & \exists \lambda_i, \mu_i \geq 0 \text{ such that } \begin{cases} (\nabla f(\bar{x}))_i + \lambda_i - \mu_i = 0 \\ \lambda_i(\bar{x}_i - v_i) = 0 \\ \mu_i(\bar{x}_i - u_i) = 0 \end{cases} \\ \Leftrightarrow \quad & b_{\bar{x}_i} = \tilde{\tilde{x}}_i(\nabla f(\bar{x}))_i \leq 0. \end{aligned}$$

Hence, condition $[LNCP]$ holds. $\square$

Corollary 1. (Necessary local optimality condition for (MCP)) *Let $\bar{x} \in S$. If $\bar{x}$ is a local minimizer of the problem (MCP), then the following condition $[LNC]$ holds:*

$$[LNC] \qquad \begin{cases} b_{\bar{x}_i} \leq 0, \quad \forall i \in \{1, \ldots, m\}, \\ b_{\bar{x}} \leq \theta_{\bar{x}}. \end{cases}$$

Proof: Let $\bar{x} \in S$ be a local minimizer of the problem (MCP). By Theorem 3, we have $b_{\bar{x}_i} \leq 0, \forall i \in \{1, \ldots, m\}$.

Moreover, $\bar{x}$ is a local minimizer of the problem (MCP) implies that

$$f(\bar{x}) \leq \min\{f(x) \mid x \in \cup_{i=1}^n N_i(\bar{x})\}.$$

We can easily verify that

$$\begin{aligned} & & f(\bar{x}) &\leq \min\{f(x) \mid x \in \cup_{i=1}^n N_i(\bar{x})\} \\ \text{and} & & b_{\bar{x}_i} &\leq 0,\ i = 1, \ldots, m \\ & & \Rightarrow b_{\bar{x}_i} &\leq \theta_{i,\bar{x}},\ \forall i = 1, \ldots, n. \end{aligned}$$

In fact, for any $i = 1, \ldots, n$, for any $x \in N_i(\bar{x})$, we have $x = \bar{x} + z_i e_i$, where $z_i \in$

$\{u_i - \bar{x}_i, v_i - \bar{x}_i\} \setminus \{0\}$, and $e_i = (0, ..., 0, 1, 0, ...0)^T$, the ith component is 1, and the others are 0.

(a). If $\bar{x}_i = u_i$, then $x_i = v_i$. By $\bar{x}$ is a local minimizer of the problem (MCP), we have

$$\begin{aligned} f(x) - f(\bar{x}) &= c_{i,i,i}(v_i - u_i)^3 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(v_i - u_i)^2 + (\nabla f(\bar{x}))_i(v_i - u_i) \geq 0 \\ &\Leftrightarrow c_{i,i,i}(v_i - u_i)^2 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(v_i - u_i) + (\nabla f(\bar{x}))_i \geq 0 \\ &\Leftrightarrow -(\nabla f(\bar{x}))_i \leq c_{i,i,i}(v_i - u_i)^2 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(v_i - u_i) \\ &\Leftrightarrow b_{\bar{x}_i} \leq \theta_{i,\bar{x}}. \end{aligned}$$

(b). If $\bar{x}_i = v_i$, then $x_i = u_i$. By $\bar{x}$ is a local minimizer of the problem (MCP), we have

$$\begin{aligned} f(x) - f(\bar{x}) &= c_{i,i,i}(u_i - v_i)^3 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(u_i - v_i)^2 + (\nabla f(\bar{x}))_i(u_i - v_i) \geq 0 \\ &\Leftrightarrow c_{i,i,i}(u_i - v_i)^2 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(u_i - v_i) + (\nabla f(\bar{x}))_i \leq 0 \\ &\Leftrightarrow (\nabla f(\bar{x}))_i \leq -c_{i,i,i}(v_i - u_i)^2 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(v_i - u_i) \\ &\Leftrightarrow b_{\bar{x}_i} \leq \theta_{i,\bar{x}}. \end{aligned}$$

(c). If $\bar{x}_i \in (u_i, v_i)$, then $x_i \in \{u_i, v_i\}, b_{\bar{x}_i} = 0$. By $\bar{x}$ is a local minimizer of the problem (MCP), we have

$$\begin{aligned} f(x) - f(\bar{x}) &= c_{i,i,i}(x_i - \bar{x}_i)^3 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(x_i - \bar{x}_i)^2 + (\nabla f(\bar{x}))_i(x_i - \bar{x}_i) \geq 0 \\ &\Leftrightarrow c_{i,i,i}(x_i - \bar{x}_i)^3 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(x_i - \bar{x}_i)^2 \geq 0 \\ &\Leftrightarrow c_{i,i,i}(x_i - \bar{x}_i) + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2} \geq 0 \\ &\Leftrightarrow b_{\bar{x}_i} \leq \theta_{i,\bar{x}}. \end{aligned}$$

Hence, condition $[LNC]$ holds. □

Remark 1. $a)$ *Let* $\bar{x} \in S$. *If* $m = 0$ *and* $\bar{x}$ *is a local minimizer of* (MCP)*, then the following condition* $[LNCD]$ *holds:*

$$[LNCD] \qquad b_{\bar{x}} \leq \theta_{\bar{x}}.$$

$b)$ *Let* $\bar{x} \in S$. *If* $m = n$ *and* $\bar{x}$ *is a local minimizer of* (MCP)*, then the following condition* $[LNCC]$ *holds:*

$$[LNCC] \qquad b_{\bar{x}} \leq 0 \text{ and } b_{\bar{x}} \leq \theta_{\bar{x}}.$$

Now we will discuss a necessary global optimality condition for the problem (MCP).

Theorem 4. (Necessary global optimality condition for (MCP)**)** *Let* $\bar{x} \in S$. *If* $\bar{x}$ *is a global minimizer of the problem* (MCP)*, then the following condition* $[GNC]$ *holds:*

$$[GNC] \qquad b_{\bar{x}_i} \leq 0, \ \forall i \in \{1, \ldots, m\} \text{ and } b_{\bar{x}} \leq \alpha_{\bar{x}}.$$

Proof: Let $\bar{x} \in S$. If $\bar{x}$ is a global minimizer of the problem (MCP), then for any $x = (\bar{x}_1, \ldots, \bar{x}_{i-1}, x_i, \bar{x}_{i+1}, \ldots, \bar{x}_n)^T \in S, \forall i = 1, \ldots, n,$

$$\begin{aligned} & f(x) - f(\bar{x}) \\ = \ & c_{i,i,i}(x_i - \bar{x}_i)^3 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(x_i - \bar{x}_i)^2 + (\nabla f(\bar{x}))_i(x_i - \bar{x}_i) \geq 0. \end{aligned} \tag{2.10}$$

Now we can prove that (2.10) is equivalent to $[GNC]$. For any $i = 1, \ldots, m$, we consider the following cases:

1°. If $\bar{x}_i = u_i$, then (2.10) is equivalent to

$$g_{i,\bar{x}}(x_i) := c_{i,i,i}(x_i - \bar{x}_i)^2 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(x_i - \bar{x}_i) + (\nabla f(\bar{x}))_i \geq 0, \ \forall x_i \in (u_i, v_i],$$

which means that

$$\min_{x_i \in [u_i, v_i]} g_{i,\bar{x}}(x_i) \geq 0.$$

We can easily verify that

$$\min_{x_i \in [u_i, v_i]} g_{i,\bar{x}}(x_i) = \min\{0, \alpha_{i,\bar{x}}\} + (\nabla f(\bar{x}))_i.$$

Here we just need to verify that

$$\min_{x_i \in [u_i, v_i]} \left(c_{i,i,i}(x_i - \bar{x}_i)^2 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(x_i - \bar{x}_i) \right) = \min\{0, \alpha_{i,\bar{x}}\}.$$

In fact, obviously,

$$\min_{x_i \in [u_i, v_i]} \left(c_{i,i,i}(x_i - \bar{x}_i)^2 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(x_i - \bar{x}_i) \right) \leq 0$$

and if $c_{i,i,i} > 0$,

$$\begin{aligned} & \left(c_{i,i,i}(x_i - \bar{x}_i)^2 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(x_i - \bar{x}_i) \right) \\ = \quad & c_{i,i,i} \left[(x_i - u_i) + \frac{1}{4c_{i,i,i}} \frac{\partial^2 f(\bar{x})}{\partial x_i^2} \right]^2 - \frac{1}{16 c_{i,i,i}} \left[\frac{\partial^2 f(\bar{x})}{\partial x_i^2} \right]^2 \\ = \quad & c_{i,i,i}(x_i - y_{i,\bar{x}})^2 + \eta_{i,\bar{x}}, \end{aligned}$$

where $y_{i,\bar{x}}$ and $\eta_{i,\bar{x}}$ are defined by (2.8) and (2.7), respectively.

Hence, if moreover $y_{i,\bar{x}} \in (u_i, v_i)$, then

$$\min_{x_i \in [u_i, v_i]} \left(c_{i,i,i}(x_i - \bar{x}_i)^2 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(x_i - \bar{x}_i) \right) = \eta_{i,\bar{x}} = \alpha_{i,\bar{x}}.$$

We can easily verify that in the other cases (which include (1) $c_{i,i,i} > 0$ but $y_{i,\bar{x}} \notin (u_i, v_i)$, and (2) $c_{i,i,i} \leq 0)$,

$$
\begin{aligned}
& \min_{x_i \in [u_i, v_i]} \left(c_{i,i,i}(x_i - \bar{x}_i)^2 + \frac{1}{2} \frac{\partial^2 f(\bar{x})}{\partial x_i^2} (x_i - \bar{x}_i) \right) \\
= \;& \min\{0, c_{i,i,i}(v_i - u_i)^2 + \frac{1}{2} \frac{\partial^2 f(\bar{x})}{\partial x_i^2} (v_i - u_i)\} \\
= \;& \min\{0, \theta_{i,\bar{x}}\} \\
= \;& \min\{0, \alpha_{i,\bar{x}}\},
\end{aligned}
$$

where $\theta_{i,\bar{x}}$ is defined by (2.6).

Hence, (2.10) is equivalent to

$$
\min\{0, \alpha_{i,\bar{x}}\} + (\nabla f(\bar{x}))_i \geq 0 \Leftrightarrow \widetilde{\bar{x}}_i (\nabla f(\bar{x}))_i \leq \min\{0, \alpha_{i,\bar{x}}\}.
$$

2°. If $\bar{x}_i = v_i$, then (2.10) is equivalent to

$$
g_{i,\bar{x}}(x_i) := c_{i,i,i}(x_i - \bar{x}_i)^2 + \frac{1}{2} \frac{\partial^2 f(\bar{x})}{\partial x_i^2} (x_i - \bar{x}_i) + (\nabla f(\bar{x}))_i \leq 0, \; \forall x_i \in [u_i, v_i),
$$

which means that

$$
\min_{x_i \in [u_i, v_i]} [-g_{i,\bar{x}}(x_i)] \geq 0.
$$

We can easily verify that

$$
\min_{x_i \in [u_i, v_i)} [-g_{i,\bar{x}}(x_i)] = \min\{0, \alpha_{i,\bar{x}}\} - (\nabla f(\bar{x}))_i.
$$

The proof is similar as the proof when $\bar{x}_i = u_i$. Hence, (2.10) is equivalent to that

$$
\min\{0, \alpha_{i,\bar{x}}\} - (\nabla f(\bar{x}))_i \geq 0 \Leftrightarrow \widetilde{\bar{x}}_i (\nabla f(\bar{x}))_i \leq \min\{0, \alpha_{i,\bar{x}}\}.
$$

3°. If $u_i < \bar{x}_i < v_i$, then (2.10) is equivalent to

$$\begin{cases} c_{i,i,i}(x_i-\bar{x}_i)^2 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(x_i-\bar{x}_i) + (\nabla f(\bar{x}))_i \le 0, & \forall x_i \in [u_i, \bar{x}_i) \\ c_{i,i,i}(x_i-\bar{x}_i)^2 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(x_i-\bar{x}_i) + (\nabla f(\bar{x}))_i \ge 0, & \forall x_i \in (\bar{x}_i, v_i] \end{cases}$$
$$\Leftrightarrow \begin{cases} (\nabla f(\bar{x}))_i = 0, \\ c_{i,i,i}(x_i-\bar{x}_i) + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2} \ge 0, \forall x_i \in [u_i, v_i], x_i \ne \bar{x}_i \end{cases}$$
$$\Leftrightarrow \begin{cases} (\nabla f(\bar{x}))_i = 0, \\ \min_{x_i \in [u_i, v_i]} \left[c_{i,i,i}(x_i-\bar{x}_i) + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2} \right] \ge 0 \end{cases}.$$

Obviously, we have that

$$\begin{aligned} & \min_{x_i \in [u_i, v_i]} \left[c_{i,i,i}(x_i-\bar{x}_i) + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2} \right] \\ = & \min\{c_{i,i,i}(u_i-\bar{x}_i) + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}, c_{i,i,i}(v_i-\bar{x}_i) + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}\} \\ = & \theta_{i,\bar{x}} \\ = & \alpha_{i,\bar{x}}. \end{aligned}$$

Hence $\min\{0, \alpha_{i,\bar{x}}\} = 0 = \widetilde{\bar{x}}_i(\nabla f(\bar{x}))_i$.

For $i = m+1, \ldots, n$, consider the following cases:

4°. If $\bar{x}_i = u_i$, then (2.10) is equivalent to

$$\begin{aligned} & g_{i,\bar{x}}(x_i) := c_{i,i,i}(x_i-\bar{x}_i)^2 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(x_i-\bar{x}_i) + (\nabla f(\bar{x}))_i \ge 0, \text{ for } x_i = v_i \\ \Leftrightarrow\ & (\nabla f(\bar{x}))_i \ge -\alpha_{i,\bar{x}} \\ \Leftrightarrow\ & \widetilde{\bar{x}}_i(\nabla f(\bar{x}))_i \le \alpha_{i,\bar{x}}. \end{aligned}$$

5°. If $\bar{x}_i = v_i$, then (2.10) is equivalent to

$$
\begin{aligned}
& g_{i,\bar{x}}(x_i) := c_{i,i,i}(x_i-\bar{x}_i)^2 + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(x_i-\bar{x}_i) + (\nabla f(\bar{x}))_i \le 0, \text{ for } x_i = u_i \\
\Leftrightarrow\ & (\nabla f(\bar{x}))_i \le \alpha_{i,\bar{x}} \\
\Leftrightarrow\ & \widetilde{\bar{x}}_i(\nabla f(\bar{x}))_i \le \alpha_{i,\bar{x}}.
\end{aligned}
$$

Hence, if $\bar{x}$ is a global minimizer of (MCP), then the condition $[GNC]$ holds. □

Remark 2. *Let $\bar{x} \in S$, and let $h(y) := f(y_1,\dots,y_m,\bar{x}_{m+1},\dots,\bar{x}_n)$, where $y = (y_1,\dots,y_m)^T \in \prod_{i=1}^{m}[u_i,v_i]$ and $f(x)$ is decided by (2.1). $\bar{x}$ is a local minimizer of the problem (MCP) implies that $\bar{y} = (\bar{x}_1,\dots,\bar{x}_m)^T$ is a traditional local minimizer of $h(y)$ on $\prod_{i=1}^{m}[u_i,v_i]$. Then the following KKT condition holds: for any $i = 1,\cdots,m$, $\exists\lambda_i \ge 0$ and $\mu_i \ge 0$, such that*

$$
\begin{aligned}
(\nabla f(\bar{x}))_i + \lambda_i - \mu_i &= 0 \\
\lambda_i(\bar{x}_i - v_i) &= 0 \\
\mu_i(\bar{x}_i - u_i) &= 0,
\end{aligned}
$$

which is equivalent to

$$[KKT] \qquad b_{\bar{x}_i} = \widetilde{\bar{x}}_i(\nabla f(\bar{x}))_i \le 0,\ i = 1,\cdots,m.$$

Obviously, we have that

$$[GNC] \Rightarrow [LNC] \Rightarrow [KKT].$$

But

$$[KKT] \nRightarrow [LNC] \nRightarrow [GNC].$$

To prove $[GNC] \Rightarrow [LNC]$, by (2.9), we just need to prove that $\eta_{i,\bar{x}} \le \theta_{i,\bar{x}}$ when $y_{i,\bar{x}}$

$\in (u_i, v_i)$, $\widetilde{\bar{x}}_i c_{i,i,i} < 0$ and $i \in \{1, \ldots, m\} \setminus m_{\bar{x}}$ since in the other cases $\alpha_{i,\bar{x}} = \theta_{i,\bar{x}}$. Actually, we have that $\eta_{i,\bar{x}} < \theta_{i,\bar{x}}$ when $y_{i,\bar{x}} \in (u_i, v_i)$, $\widetilde{\bar{x}}_i c_{i,i,i} < 0$ and $i \in \{1, \ldots, m\} \setminus m_{\bar{x}}$.

In fact, if $\widetilde{\bar{x}}_i c_{i,i,i} < 0$, then

$$\eta_{i,\bar{x}} - \theta_{i,\bar{x}} < 0 \Leftrightarrow$$
$$\tfrac{\widetilde{\bar{x}}_i}{c_{i,i,i}} \left\{ \left[\widetilde{\bar{x}}_i \tfrac{1}{16 c_{i,i,i}} [\tfrac{\partial^2 f(\bar{x})}{\partial x_i^2}]^2 \right] - \left[-\widetilde{\bar{x}}_i c_{i,i,i} (v_i - u_i)^2 + \tfrac{1}{2} \tfrac{\partial^2 f(\bar{x})}{\partial x_i^2} (v_i - u_i) \right] \right\} > 0.$$

And

$$\begin{aligned} & \frac{\widetilde{\bar{x}}_i}{c_{i,i,i}} \left\{ \left[\widetilde{\bar{x}}_i \frac{1}{16 c_{i,i,i}} [\frac{\partial^2 f(\bar{x})}{\partial x_i^2}]^2 \right] - \left[-\widetilde{\bar{x}}_i c_{i,i,i} (v_i - u_i)^2 + \frac{1}{2} \frac{\partial^2 f(\bar{x})}{\partial x_i^2} (v_i - u_i) \right] \right\} \\ = \;& \frac{1}{16 c_{i,i,i}^2} [\frac{\partial^2 f(\bar{x})}{\partial x_i^2}]^2 + (v_i - u_i)^2 - \frac{1}{2} \frac{\widetilde{\bar{x}}_i}{c_{i,i,i}} \frac{\partial^2 f(\bar{x})}{\partial x_i^2} (v_i - u_i) \\ = \;& \left[(v_i - u_i) - \frac{1}{4} \frac{\widetilde{\bar{x}}_i}{c_{i,i,i}} \frac{\partial^2 f(\bar{x})}{\partial x_i^2} \right]^2 . \end{aligned}$$

If $y_{i,\bar{x}} \in (u_i, v_i)$ and $i \in \{1, \ldots, m\} \setminus m_{\bar{x}}$, then we have that

$$\begin{aligned} & (v_i - u_i) - \frac{1}{4} \frac{\widetilde{\bar{x}}_i}{c_{i,i,i}} \frac{\partial^2 f(\bar{x})}{\partial x_i^2} \\ = \;& \begin{cases} v_i - y_{i,\bar{x}} > 0 & \textit{if } \bar{x}_i = u_i \\ y_{i,\bar{x}} - u_i < 0 & \textit{if } \bar{x}_i = v_i \end{cases} . \end{aligned}$$

Hence if $y_{i,\bar{x}} \in (u_i, v_i)$, $\widetilde{\bar{x}}_i c_{i,i,i} < 0$ and $i \in \{1, \ldots, m\} \setminus m_{\bar{x}}$, then $\eta_{i,\bar{x}} - \theta_{i,\bar{x}} < 0$, i.e., $\eta_{i,\bar{x}} < \theta_{i,\bar{x}}$ which means that $[GNC]$ implies $[LNC]$. But the following example illustrates that

$$[KKT] \not\Rightarrow [LNC] \not\Rightarrow [GNC].$$

Example 1. *Consider the problem*

$$\min \qquad f(x) := -2x_1^3 + 2x_1^2 x_2 - x_1^2 + 2x_1 x_2 - 3x_2^2 + 8x_1 + 2x_2$$

$$s.t.$$

$$x_1 \in [-4, 1], \qquad x_2 \in \{-2, 2\}.$$

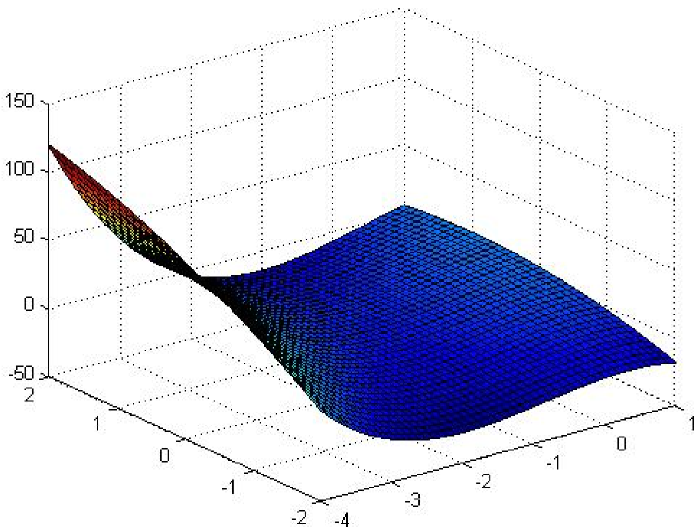
We have $\nabla f(x) = (-6x_1^2 + 4x_1 x_2 - 2x_1 + 2x_2 + 8, 2x_1^2 + 2x_1 - 6x_2 + 2)^T$, $\frac{\partial^2 f(x)}{\partial x_1^2} = -12x_1 + 4x_2 - 2$, $\frac{\partial^2 f(x)}{\partial x_2^2} = -6$. *We consider the following three points:* $\bar{x} = (-2, -2)^T$, $\bar{y} = (1, -2)^T$ *and* $\bar{z} = (-1, 2)^T$. *From figure 2.1, we can see that* $\bar{x} = (-2, -2)^T$ *is the global minimizer and* $\bar{y} = (1, -2)^T$*is a local minimizer of Example 1. It is easy to check that both* $[GNC]$ *and* $[LNC]$ *hold at* $\bar{x}$*, while* $[LNC]$ *holds at* $\bar{y}$*, but* $[GNC]$ *does not hold at* $\bar{y}$*, furthermore,* $[KKT]$ *holds at* $\bar{z}$*, but* $[LNC]$ *does not hold at* $\bar{z}$*.*

In fact, $\nabla f(\bar{x}) = (0, 18)^T$, $b_{\bar{x}_1} = 0$, $b_{\bar{x}_2} = -18$, $\theta_{1,\bar{x}} = 1$, $\theta_{2,\bar{x}} = -12$*;* $\alpha_{1,\bar{x}} = 1$, $\alpha_{2,\bar{x}} = -12$. *Thus* $b_{\bar{x}_1} \leq 0$, $b_{\bar{x}_1} \leq \theta_{1,\bar{x}}$, $b_{\bar{x}_2} \leq \theta_{2,\bar{x}}$ *which means that* $[LNC]$ *holds at* $\bar{x}$*;* $b_{\bar{x}_1} \leq 0$, $b_{\bar{x}_1} \leq \alpha_{1,\bar{x}}$, $b_{\bar{x}_2} \leq \alpha_{2,\bar{x}}$ *which means that* $[GNC]$ *holds at* $\bar{x}$*.*

While $\nabla f(\bar{y}) = (-12, 18)$, $b_{\bar{y}_1} = -12$, $b_{\bar{y}_2} = -18$, $\theta_{1,\bar{y}} = -5$, $\theta_{2,\bar{y}} = -12$*;* $\alpha_{1,\bar{y}} = \eta_{1,\bar{y}} = -15.1250$, $\alpha_{2,\bar{y}} = -12$. *Here* $b_{\bar{y}_1} \leq 0$, $b_{\bar{y}_1} \leq \theta_{1,\bar{y}}$, $b_{\bar{y}_2} \leq \theta_{2,\bar{y}}$ *which means that* $[LNC]$ *holds at* $\bar{y}$*; but* $b_{\bar{y}_1} > \alpha_{1,\bar{y}}$, *so* $[GNC]$ *does not hold at* $\bar{y}$*.*

Furthermore, $\nabla f(\bar{z}) = (0, -10)$, $b_{\bar{z}_1} = 0$, $b_{\bar{z}_2} = -10$, $\theta_{1,\bar{z}} = 5$, $\theta_{2,\bar{z}} = -12$. *Here* $b_{\bar{z}_1} \leq 0$ *which means that* $[KKT]$ *holds at* $\bar{z}$*; but* $b_{\bar{z}_2} > \theta_{2,\bar{z}}$, *so* $[LNC]$ *does not hold at* $\bar{z}$*.*

Figure 2.1.: The behavior of $f(x)$ on $[-4, 1] \times \{-2, 2\}$ in Example 1



Corollary 2. *Let $\bar{x} \in S$. If $m = 0$ and $\bar{x}$ is a global minimizer of (MCP), then the following condition $[GNCD]$ holds:*

$$[GNCD] \qquad b_{\bar{x}} \leq \alpha_{\bar{x}},$$

where $\alpha_{\bar{x}} = \theta_{\bar{x}}$. Hence $[GNCD] = [LNCD]$.

It can be obtained directly from Theorem 4.

Corollary 3. *Let $\bar{x} \in S$. If $m = n$ and $\bar{x}$ is a global minimizer of (MCP), then the following condition $[GNCC]$ holds:*

$$[GNCC] \qquad b_{\bar{x}} \leq 0 \text{ and } b_{\bar{x}} \leq \alpha_{\bar{x}}.$$

It can be obtained directly from Theorem 4.

Remark 3. *If $c_{j,l,r} = 0$ for $j + l + r = 3$, $0 \leq j, l, r \leq 3$, then the problem (MCP) reduces to a quadratic programming problem with mixed variables:*

$$(MQP) \qquad \min \quad f(x) = \frac{1}{2}x^T Ax + a^T x$$
$$s.t.$$
$$x_i \in [u_i, v_i],\ i = 1, \ldots, m,$$
$$x_i \in \{u_i, v_i\},\ i = m + 1, \ldots, n,$$

where $A = (a_{ij})_{n \times n}$ is an $n \times n$ symmetric matrix. In this case,

$$\alpha_{i,\bar{x}} = \theta_{i,\bar{x}} = \begin{cases} \frac{1}{2}a_{ii}, & i \in m_{\bar{x}} \\ \frac{1}{2}a_{ii}(v_i - u_i), & \textit{otherwise} \end{cases}.$$

Then, the necessary global optimality condition $[GNC]$ for the problem (MQP) is equivalent to the following condition:

$$[GNC]' \qquad \begin{cases} b_{\bar{x}_i} \leq 0,\ \forall i \in \{1, \ldots, m\}, \\ b_{\bar{x}_i} \leq \frac{1}{2}a_{ii}(v_i - u_i),\ \forall i \in \{1, \ldots, n\}. \end{cases}$$

When $u_i = -1$ and $v_i = 1$, $[GNC]'$ is just the condition $[NC1]$ given in Theorem 3.7 in [145] *for a quadratic optimization problem with mixed variables.*

2.3. Optimization methods for (MCP)

2.3.1. Weakly local optimization method for (P)

In this subsection, we will design a weakly local optimization method for the problem (P) according to the necessary condition $[LNCP]$.

Definition 14. *Let $\bar{x} \in S$. $\bar{x}$ is said to be a weakly local minimizer of the problem (P) iff $\bar{x}$ satisfies the condition $[LNCP]$.*

Obviously, a local minimizer of the problem (P) is also a weakly local minimizer of the problem (P) since condition $[LNCP]$ is a necessary condition for $\bar{x}$ to be a local minimizer of the problem (P).

Algorithm 3. Weakly local optimization method for (P):$(WLOM)$.

Step 0. Take an initial point $X_1 = (x_1^1, \dots, x_1^n)^T \in S$. Let $\bar{x} := X_1, k := 1$.

Step 1. Check whether the condition $[LNCP]_1$ holds:

$$[LNCP]_1 \quad b_{\bar{x}_i} \leq 0, i = 1, \dots, m.$$

If $[LNCP]_1$ holds, go to Step 3; otherwise go to Step 2.

Step 2. Let $h(y) := f(y_1, \dots, y_m, \bar{x}_{m+1}, \dots, \bar{x}_n)$, where $y = (y_1, \dots, y_m)^T$ and $y \in \prod_{i=1}^{m} [u_i, v_i]$. Find a traditional local minimizer $y^* = (y_1^*, \dots, y_m^*)^T$ of $h(y)$ on $\prod_{i=1}^{m} [u_i, v_i]$ starting from point $\bar{y} = (\bar{x}_1, \dots, \bar{x}_m)^T$ (any traditional (gradient based) local optimization methods can be used to find the traditional local minimizer). Let $k := k + 1, X_k := (y_1^*, \dots, y_m^*, \bar{x}_{m+1}, \dots, \bar{x}_n)^T$ and let $\bar{x} := X_k$, go to Step 3.

Step 3. Check whether the following condition $[LNCP]_2$ holds:

$$[LNCP]_2 \quad f(\bar{x}) \leq f(x), \forall x \in \bigcup_{i=1}^{n} N_i(\bar{x}).$$

If $[LNCP]_2$ does not hold, go to Step 4; otherwise, go to Step 5.

Step 4. Let $x^* = (x_1^*, \dots, x_n^*)^T :=$ argmin $\{f(x) \mid x \in \bigcup_{i=1}^{n} N_i(\bar{x})\}$, let $\bar{x} := x^*$ and go to Step 1.

Step 5. Stop. $\bar{x}$ is a weakly local minimizer of the problem (P).

Theorem 5. *For a given initial point $X_1 \in S$, we can obtain a weakly local minimizer $\bar{x}$ of the problem (P) in finite iteration times by the given weakly local optimization method $(WLOM)$.*

Proof. Firstly, by Remark 2, we know that in *step 2*, if $y^* := (y_1^*, \ldots, y_m^*)^T$ is a traditional local minimizer of $h(y)$ on $\prod_{i=1}^{m}[u_i, v_i]$ starting from the point $(\bar{x}_1, \ldots, \bar{x}_m)^T$, let $k := k+1$, $X_k := (y_1^*, \ldots, y_m^*, \bar{x}_{m+1}, \ldots, \bar{x}_n)^T$ and let $\bar{x} := X_k$, then $b_{\bar{x}_i} \leq 0, \forall i = 1, \ldots, m$.

Secondly, from *step 3* to *step 4*, since $[LNCP]_2$ does not hold and let $x^* = (x_1^*, \ldots, x_n^*)^T :$ $=$ argmin $\{f(x) \mid x \in \bigcup_{i=1}^{n} N_i(\bar{x})\}$, then we must have that $f(x^*) < f(\bar{x})$. In fact, since $[LNCP]_2$ does not hold, there must exist an $i_0 \in \{1, \ldots, n\}$ and a $y_{i_0} \in N_{i_0}(\bar{x})$ such that $f(y_{i_0}) < f(\bar{x})$. By $f(x^*) \leq f(y_{i_0})$, we have that $f(x^*) < f(\bar{x})$.

Here we just need to prove that Algorithm $(WLOM)$ needs only finite iteration times from *step 1* to *step 5*. Let

$$\eta := \min\left\{|f(x) - f(y)| \big| x, y \in \prod_{i=1}^{n}\{u_i, v_i\} \text{ and } f(x) \neq f(y)\right\},$$

$$M := \max\{f(x) \mid x \in \prod_{i=1}^{n}\{u_i, v_i\}\} \text{ and } m := \min\{f(x) \mid x \in \prod_{i=1}^{n}\{u_i, v_i\}\}.$$

If $M = m$, we have $\left\{|f(x) - f(y)| \big| x, y \in \prod_{i=1}^{n}\{u_i, v_i\} \text{ and } f(x) \neq f(y)\right\} = \emptyset$, then we define that $\eta = 0$. If $M \neq m$, then we have that $\eta > 0$. If $\eta = 0$, then condition $[LNCP]_2$ must hold. Thus Algorithm $(WLOM)$ needs only one iteration from *step 1* to *step 5*. Here, we suppose that $\eta > 0$. Then a weakly local minimizer $\bar{x}$ of the problem (P) starting from a given point X_1 can be obtained in at most $\frac{M-m}{\eta} + 1$ steps by Algorithm $(WLOM)$.

Indeed, since $f(x^*) < f(\bar{x})$ from *step 3* $\to$ *step 4*, there are at most $\frac{M-m}{\eta}$ iteration times from *step 3* $\to$ *step 4*. Obviously, the iteration time from *step 1* $\to$ *step 5* is less than or equal to the iteration times from *step 1* $\to$ *step 4* plus 1, and the iteration time from *step 1* $\to$ *step 4* is equal to the iteration times from *step 3* $\to$ *step 4*. Hence, the total iteration time from *step1* to *step 5* is at most $\frac{M-m}{\eta}+1$. $\square$

2.3.2. Strongly local optimization method for (MCP)

In this subsection, we will design a strongly local optimization method for the problem (MCP) according to the global necessary optimality condition $[GNC]$.

Definition 15. *Let $\bar{x}\in S$. $\bar{x}$ is said to be a strongly local minimizer of the problem (MCP) iff $\bar{x}$ satisfies the condition $[GNC]$.*

Obviously, $\bar{x}$ is a strongly local minimizer of the problem $(MCP)\Rightarrow\bar{x}$ is a weakly local minimizer of the problem (MCP). By Example 1, we know that $\bar{y}$ is a weakly local minimizer of the problem $(MCP)\nRightarrow\bar{y}$ is a strongly local minimizer of the problem (MCP).

For the problem (MCP), let

$$N_i'(\bar{x}):\ =\ \{\bar{x}+(z_i-\bar{x}_i)e_i\mid z_i\in\{y_{i,\bar{x}}\}\cap(u_i,v_i)\},i=1,\ldots,m,\qquad(2.11)$$

where $y_{i,\bar{x}}$ is defined by (2.8), e_i with the ith component is 1 and the others are 0. Note that $|N_i'(\bar{x})|\le 1$.

Algorithm 4. Strongly local optimization method for (MCP):$(SLOM)$.

Step 0. Take an initial point $X_1=(x_1^1,\ldots,x_1^n)^T\in S$. Let $\bar{x}:=X_1,k:=1$.

Step 1. Check whether the condition $[GNC]_1$ holds:

$$[GNC]_1\quad b_{\bar{x}_i}\le 0,i=1,\ldots,m.$$

If $[GNC]_1$ holds, go to Step 3; otherwise go to Step 2.

Step 2. Let $h(y):=f(y_1,\ldots,y_m,\bar{x}_{m+1},\ldots,\bar{x}_n)$, where $y=(y_1,\ldots,y_m)^T$ and $y\in\prod_{i=1}^{m}[u_i,v_i]$. Find a traditional local minimizer $y^*=(y_1^*,\ldots,y_m^*)^T$ of $h(y)$ on $\prod_{i=1}^{m}[u_i,v_i]$

starting from point $\bar{y} = (\bar{x}_1, \ldots, \bar{x}_m)^T$ (any traditional (gradient based) local optimization methods can be used to find the traditional local minimizer). Let $k := k + 1$, and let $X_k := (y_1^*, \ldots, y_m^*, \bar{x}_{m+1}, \ldots, \bar{x}_n)^T$. Let $\bar{x} := X_k$, go to Step 3.

Step 3. Check whether the following condition $[GNC]_2$ holds:

$$[GNC]_2 \quad b_{\bar{x}_i} \leq \alpha_{i,\bar{x}}, \forall i = 1, \ldots, n.$$

If $[GNC]_2$ does not hold, go to Step 4; otherwise, go to Step 5.

Step 4. Let

$$x^* = (x_1^*, \ldots, x_n^*)^T := \text{ argmin } \{f(x) \mid x \in \cup_{i=1}^n N_i(\bar{x}) \cup_{i=1}^m N_i'(\bar{x})\},$$

let $\bar{x} := x^*$ and goto Step 1.

Step 5. Stop. $\bar{x}$ is a strongly local minimizer of the problem (MCP).

Theorem 6. *For a given initial point $X_1 \in S$, we can obtain a strongly local minimizer $\bar{x}$ of the problem (MCP) in finite iteration times by the given strongly local optimization method $(SLOM)$.*

Proof. The proof is similar as the proof of Theorem 5. Here we just need to replace $[LNCP]_1$ and $[LNCP]_2$ by $[GNC]_1$ and $[GNC]_2$, respectively, and replace η, M and m by

$$\min\left\{|f(x) - f(y)| \big| x, y \in \prod_{i=1}^m \{u_i, v_i, \{y_{i,\bar{x}}\} \cap (u_i, v_i)\} \prod_{i=m+1}^n \{u_i, v_i\}, f(x) \neq f(y)\right\},$$

$$\max\left\{f(x) \mid x \in \prod_{i=1}^m \{u_i, v_i, \{y_{i,\bar{x}}\} \cap (u_i, v_i)\} \prod_{i=m+1}^n \{u_i, v_i\}\right\},$$
$$\min\left\{f(x) \mid x \in \prod_{i=1}^m \{u_i, v_i, \{y_{i,\bar{x}}\} \cap (u_i, v_i)\} \prod_{i=m+1}^n \{u_i, v_i\}\right\},$$

respectively.

$\square$

Remark 4. *In Algorithm 3 and Algorithm 4, in step 2, it is very easy to obtain a traditional local minimizer of $h(y)$ on $\prod_{i=1}^{m}[u_i, v_i]$ since any traditional (gradient based) local optimization methods, such as the Newton method, the Quasi-Newton method and the Conjugate gradient method can be used here. In section 2.4, the optimization subroutine within the optimization Toolbox in Matlab is used to find the traditional local minimizers. In Algorithm 3, in step 4, it is easy to find the point x^* such that $x^* =$ argmin$\{f(x) \mid x \in \bigcup_{i=1}^{n} N_i(\bar{x})\}$, i.e., $f(x^*) \le f(x)$ for any $x \in \bigcup_{i=1}^{n} N_i(\bar{x})$ since $|\bigcup_{i=1}^{n} N_i(\bar{x})| \le 2n$. Similarly, in Algorithm 4, in step 4, it is also easy to find the point x^* such that $x^* =$ argmin$\{f(x) \mid x \in \bigcup_{i=1}^{n} N_i(\bar{x}) \bigcup_{i=1}^{m} N_i'(\bar{x})\}$, i.e., $f(x^*) \le f(x)$ for any $x \in \bigcup_{i=1}^{n} N_i(\bar{x}) \bigcup_{i=1}^{m} N_i'(\bar{x})$ since $|\bigcup_{i=1}^{n} N_i(\bar{x}) \bigcup_{i=1}^{m} N_i'(\bar{x})| \le 2n + m$.*

2.3.3. Global optimization method for (MCP)

To introduce the global optimization method, in this chapter we will use the auxiliary function which was presented by (1.5) in Chapter 1. For the properties of this auxiliary function, see Chapter 1. Note, the K-K-T point defined in property 3 in Chapter 1 and the weakly local minimizer defined in this chapter are the same thing.

In the following, we will introduce a global optimization method to find a global minimizer of the problem (MCP). This method combines the weakly local optimization method for the problem (P) and the strongly local optimization method for the problem (MCP) and the auxiliary function $F_{r,\bar{x}}(x)$ which was presented by (1.5) in Chapter 1. The auxiliary function is used to escape the current local minimizer and to find a better feasible point of the problem (MCP).

Algorithm 5. Global optimization method for (MCP):(GOM).

Step 0. Take an initial point $x_1 \in S$, a sufficiently small positive number μ, and an initial $r_1 > 0$. Set $r := r_1, k := 1$.

Step 1. Use the strongly local optimization method $(SLOM)$ to solve the problem (MCP) starting from x_k. Let x_k^* be the obtained strongly local minimizer of the problem (MCP).

Step 2. Construct the following auxiliary function

$$F_{r,x_k^*}(x) = \frac{1}{\|x - x_k^*\|^2 + 1} g_r\Big(f(x) - f(x_k^*)) + f_r(f(x) - f(x_k^*)\Big).$$

Consider the following problem:

$$\begin{aligned} \min \quad & F_{r,x_k^*}(x) \\ s.t. \quad & x \in S. \end{aligned} \tag{2.12}$$

Let $\bar{x}_k := x_k^*$, $\delta \leq \delta(\bar{x}_k)$ and $i := 1$, go to Step 3, where $\delta(\bar{x}_k)$ is defined by (2.4).

Step 3. Let $\bar{x}_k^i \in N_{i,\delta}(\bar{x}_k) \setminus \{\bar{x}_k^i\}$. If $f(\bar{x}_k^i) < f(x_k^*)$, let $x_{k+1} := \bar{x}_k^i, k := k + 1$, go to Step 1; otherwise go to Step 4.

Step 4. Use the weakly local optimization method $(WLOM)$ to solve the problem (2.12) starting from $\bar{x}_k^i$. Let $\bar{x}_k^*$ be the obtained weakly local minimizer of the problem (2.12). If $f(\bar{x}_k^*) < f(x_k^*)$, let $x_{k+1} := \bar{x}_k^*, k := k + 1$, go to Step 1; otherwise, let $i := i + 1$, if $i \leq m$, go to Step 3, else go to Step 5.

Step 5. If $r \geq \mu$, decrease r, such as, let $r := r/10$, go to Step 2; otherwise, go to Step 6.

Step 6. Stop and x_k^* is the obtained global minimizer or approximate global minimizer of the problem (MCP).

The numerical examples given in the following Section illustrate that the global minimization method **Algorithm 5** is very efficient and stable.

2.4. Numerical examples

In this section, we apply Algorithm 5 to the following test examples. In the following examples, we take $\mu = r_1 = 0.01$.

x_k: the $k-$th initial point;

$f(x_k)$: the function value of $f(x)$ at the $k-$th initial point x_k;

x_k^*: the $k-$th strongly local minimizer of the problem (MCP) starting from x_k;

$f(x_k^*)$: the function value of $f(x)$ at x_k^*;

Example 2. *Consider the problem*

$$\begin{aligned} \min \quad & f(x) := 4x_3^3 + x_1x_2x_4 + 3x_1^2x_2 + 2x_1x_2^2 - 5x_2x_4 - 2x_1^2 + x_2^2 \\ & \qquad - x_1x_2 - 2x_3 - 7x_2 \\ s.t. \quad & \\ & x_1, x_2 \in [-3, 5], \qquad x_3, x_4 \in \{-3, 5\}. \end{aligned}$$

Table 2.1 records the numerical results of solving Example 2 by **Algorithm 5**. *From it, we see the first strongly local minimizer $x_1^* = (-2.6923, 5, -3, 5)^T$ starting from the initial point $x_1 = (5, 5, 5, 5)^T$ is the global minimizer of Example 2 by* **Algorithm 5***, which illustrates that the strongly local optimization method is efficient. The other first strongly local minimizers starting from the other initial points: $x_1 = (1, 1, 5, -3)^T, x_1 = (1.4, 1.9, -3, -3)^T$ and $x_1 = (3.2, -2.1, 5, 5)^T$ are not the global minimizer, then we use the auxiliary function to find the second initial points, and the second strongly local minimizers are the global minimizer of Example 2 by* **Algorithm 5**.

Table 2.1.: Numerical results for Example 2

k	x_k	$f(x_k)$	x_k^*	$f(x_k^*)$
1	$\begin{pmatrix} 5 \\ 5 \\ 5 \\ 5 \end{pmatrix}$	1030.0000	$\begin{pmatrix} -2.6923 \\ 5 \\ -3 \\ 5 \end{pmatrix}$	-331.2308
2	$\begin{pmatrix} 1 \\ 1 \\ 5 \\ -3 \end{pmatrix}$	498.0000	$\begin{pmatrix} -3 \\ -3 \\ -3 \\ -3 \end{pmatrix}$	-306.0000
	$\begin{pmatrix} -2.6923 \\ 5 \\ -3 \\ 5 \end{pmatrix}$	-331.2308	$\begin{pmatrix} -2.6923 \\ 5 \\ -3 \\ 5 \end{pmatrix}$	-331.2308
3	$\begin{pmatrix} 1.4 \\ 1.9 \\ -3 \\ -3 \end{pmatrix}$	-76.4700	$\begin{pmatrix} -3 \\ -3 \\ -3 \\ -3 \end{pmatrix}$	-306.0000
	$\begin{pmatrix} -2.6923 \\ 5 \\ -3 \\ 5 \end{pmatrix}$	-331.2308	$\begin{pmatrix} -2.6923 \\ 5 \\ -3 \\ 5 \end{pmatrix}$	-331.2308
4	$\begin{pmatrix} 3.2 \\ -2.1 \\ 5 \\ 5 \end{pmatrix}$	477.9600	$\begin{pmatrix} 5 \\ -2.86 \\ -3 \\ 5 \end{pmatrix}$	-242.2000
	$\begin{pmatrix} -2.6923 \\ 5 \\ -3 \\ 5 \end{pmatrix}$	-331.2308	$\begin{pmatrix} -2.6923 \\ 5 \\ -3 \\ 5 \end{pmatrix}$	-331.2308

Example 3. *Consider the problem*

$$\begin{aligned} \min \quad & f(x) := 3x_1^3 + 4x_2^3 + x_3^3 + 2x_4^3 - x_2x_3x_4 - 2x_1x_2x_3 - 3x_1^2x_4 \\ & \qquad - 4x_1x_2^2 - 2x_1^2 + x_2^2 + x_1x_2 - 2x_1x_3 - x_2x_4 - 2x_1 \\ s.t. & \\ & x_1, x_2 \in [-3, 5], \qquad x_3, x_4 \in \{-3, 5\}. \end{aligned}$$

Table 2.2 records the numerical results of solving Example 3 by **Algorithm 5**. *From it, we see the first strongly local minimizer* $x_1^* = (4.9641, -3, -3, 5)^T$ *starting from the initial point* $x_1 = (2.5, -2.5, 5, 5)^T$ *is the global minimizer of Example 3 by* **Algorithm 5**, *which illustrates that the strongly local optimization method is efficient. The other first strongly local minimizers starting from the other initial points are not the global minimizer, then we use the auxiliary function to find the second initial points or the third initial points, and the second strongly local minimizers or the third strongly local minimizers are the global minimizer of Example 3 by* **Algorithm 5**.

Table 2.2.: Numerical results for Example 3

k	x_k	$f(x_k)$	x_k^*	$f(x_k^*)$
1	$\begin{pmatrix} 2.5 \\ -2.5 \\ 5 \\ 5 \end{pmatrix}$	298.1250	$\begin{pmatrix} 4.9641 \\ -3 \\ -3 \\ 5 \end{pmatrix}$	-221.0359
2	$\begin{pmatrix} -2.6 \\ 2.7 \\ -3 \\ -3 \end{pmatrix}$	-0.3100	$\begin{pmatrix} 1.7705 \\ -3 \\ -3 \\ -3 \end{pmatrix}$	-217.2440
	$\begin{pmatrix} 4.9641 \\ -3 \\ -3 \\ 5 \end{pmatrix}$	-221.0359	$\begin{pmatrix} 4.9641 \\ -3 \\ -3 \\ 5 \end{pmatrix}$	-221.0359
3	$\begin{pmatrix} 1 \\ 1 \\ 5 \\ -3 \end{pmatrix}$	79.0000	$\begin{pmatrix} 0.9364 \\ -3 \\ -3 \\ -3 \end{pmatrix}$	-203.0281
	$\begin{pmatrix} 1.7705 \\ -3 \\ -3 \\ -3 \end{pmatrix}$	-217.2440	$\begin{pmatrix} 1.7705 \\ -3 \\ -3 \\ -3 \end{pmatrix}$	-217.2440
	$\begin{pmatrix} 4.9641 \\ -3 \\ -3 \\ 5 \end{pmatrix}$	-221.0359	$\begin{pmatrix} 4.9641 \\ -3 \\ -3 \\ 5 \end{pmatrix}$	-221.0359
4	$\begin{pmatrix} -2 \\ 2 \\ 5 \\ -3 \end{pmatrix}$	239.0000	$\begin{pmatrix} 0.9364 \\ -3 \\ -3 \\ -3 \end{pmatrix}$	-203.0281

continue goes here…

k	x_k	$f(x_k)$	x_k^*	$f(x_k^*)$
	$\begin{pmatrix} 1.7705 \\ -3 \\ -3 \\ -3 \end{pmatrix}$	-217.2440	$\begin{pmatrix} 1.7705 \\ -3 \\ -3 \\ -3 \end{pmatrix}$	-217.2440
	$\begin{pmatrix} 4.9641 \\ -3 \\ -3 \\ 5 \end{pmatrix}$	-221.0359	$\begin{pmatrix} 4.9641 \\ -3 \\ -3 \\ 5 \end{pmatrix}$	-221.0359

Example 4. *Consider the problem*

$$\begin{aligned} \min \quad & f(x) := (x_8 - 1) + 2(x_9 - x_8)^2 + 3(x_3 - x_2)^3 + 4(x_2 - x_{10})^3 \\ & +(x_2 - x_1)^2 + (x_4 - x_3) + 5(x_5 - x_4)^2 + (x_6 - x_5)^3 + (x_7 - x_6)^3 \\ s.t. \quad & \\ & x_1, \cdots, x_7 \in [-3, 5], \qquad x_8, x_9, x_{10} \in \{-3, 5\}. \end{aligned}$$

Table 2.3 records the numerical results of solving Example 4 by **Algorithm 5**. *From Table 2.3, we see that the first strongly local minimizer starting from the initial points:* $x_1 = (3, 0, 3, 0, 3, 0, 3, 5, -3, -3)^T$ *and* $x_1 = (5, 4, 3, 2, 1, 2, 3, -3, 5, 5)^T$ *are the global minimizer of Example 4, which illustrates that the strongly local optimization method is efficient. The other first strongly local minimizers starting from the other initial points are not the global minimizer, then we use the auxiliary function to find the second initial points, and the second strongly local minimizers are the global minimizer of Example 4 by* **Algorithm 5**.

Table 2.3.: Numerical results for Example 4

k	x_k	$f(x_k)$	x_k^*	$f(x_k^*)$

continue goes here...

k	x_k	$f(x_k)$	x_k^*	$f(x_k^*)$
1	$\begin{pmatrix} 3 \\ 0 \\ 3 \\ 0 \\ 3 \\ 0 \\ 3 \\ 5 \\ -3 \\ -3 \end{pmatrix}$	372.0	$\begin{pmatrix} -3 \\ -3 \\ -2.6666 \\ 4.3227 \\ 4.4227 \\ 5 \\ -3 \\ -3 \\ -3 \\ 5 \end{pmatrix}$	-2556.7
2	$\begin{pmatrix} 5 \\ 4 \\ 3 \\ 2 \\ 1 \\ 2 \\ 3 \\ -3 \\ 5 \\ 5 \end{pmatrix}$	124.0	$\begin{pmatrix} -3 \\ -3 \\ -2.6667 \\ 4.3226 \\ 4.4226 \\ 5 \\ -3 \\ -3 \\ -3 \\ 5 \end{pmatrix}$	-2556.7
3	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 5 \\ 5 \\ -3 \end{pmatrix}$	112.0	$\begin{pmatrix} -3 \\ -3 \\ -2.6667 \\ 4.8999 \\ 5 \\ -3 \\ -2.9944 \\ 5 \\ 5 \\ 5 \end{pmatrix}$	-2548.3
	$\begin{pmatrix} -3 \\ -3 \\ -2.6667 \\ 4.3226 \\ 4.4226 \\ 5 \\ -3 \\ -3 \\ -3 \\ 5 \end{pmatrix}$	-2556.7	$\begin{pmatrix} -3 \\ -3 \\ -2.6667 \\ 4.3226 \\ 4.4226 \\ 5 \\ -3 \\ -3 \\ -3 \\ 5 \end{pmatrix}$	-2556.7
4	$\begin{pmatrix} -1 \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 5 \\ 5 \\ 5 \end{pmatrix}$	-484.0	$\begin{pmatrix} -3 \\ -3 \\ -2.6666 \\ 4.3226 \\ 4.4226 \\ 5 \\ -3 \\ 5 \\ 5 \\ 5 \end{pmatrix}$	-2548.7

continue goes here…

k	x_k	$f(x_k)$	x_k^*	$f(x_k^*)$
	$\begin{pmatrix} -3 \\ -3 \\ -2.6667 \\ 4.3226 \\ 4.4226 \\ 5 \\ -3 \\ -3 \\ -3 \\ 5 \end{pmatrix}$	-2556.7	$\begin{pmatrix} -3 \\ -3 \\ -2.6667 \\ 4.3226 \\ 4.4226 \\ 5 \\ -3 \\ -3 \\ -3 \\ 5 \end{pmatrix}$	-2556.7

Example 5. *Consider the problem*

$$\begin{aligned} \min \quad & f(x) := 2(x_{10} - x_9^2)(1 - x_8) + 3(x_1 - x_{10}^2)(1 - x_9) \\ & \quad + 4(x_2 - x_1^2)(1 - x_{10}) + 5(x_3 - x_2^2)(1 - x_1) \\ & \quad + 6(x_4 - x_3^2)(1 - x_2) + 7(x_5 - x_4^2)(1 - x_3) \\ & \quad + 8(x_6 - x_5^2)(1 - x_4) + 9(x_7 - x_6^2)(1 - x_5) \\ s.t. \quad & \\ & x_1, \cdots, x_7 \in [-3, 5], \qquad x_8, x_9, x_{10} \in \{-3, 5\}. \end{aligned}$$

Table 2.4 records the numerical results of solving Example 5 by **Algorithm 5**. *From Table 2.4, we can see that we obtained three global minimizers which are: (-3,5,-3,5,-3,5,-3,-3, -3,5)^T, (-3,-3,-3, -3,-3,5,-3-3,-3,-3)^T and (-3,5,-3,5,-3,5,-3,-3,-3,-3)^T with the optimal value* -2432.0000. *The global minimizer $\bar{x}$ =(-3,5,-3,5,-3,5,-3,-3,-3,5)^T is just the first strongly local minimizer of Example 5 by* **Algorithm 5** *starting from the initial point (0,0,0,0,0,0,0,5,5,5)^T, which illustrates that the strongly local optimization method is efficient.*
The global minimizer $\bar{y}$ =(-3,-3,-3,-3,-3,5,-3,-3,-3,-3)^T is the first strongly local minimizer of Example 5 by **Algorithm 5** *starting from initials (-2,-2,-2,-2,-2,-2,-2,5,5,5)^T and (2,2,1,-1, -1,0,3,-3,5,5)^T, which illustrates that the strongly local optimization method is efficient.*
The global minimizer $\bar{z}$ =(-3,5,-3,5,-3,5,-3,-3,-3,-3)^T is the second strongly local minimiz-

ers of Example 5 by **Algorithm 5** *starting from the initial points (0,1,2,3,4,5,-1,5,5,-3)^T, (5,5,-3,5,5,-3,-3,5,5,-3)^T and (1,-1,2,-2,3,-3,0,-3,-3,-3)^T.*

Table 2.4.: Numerical results for Example 5

k	x_k	$f(x_k)$	x_k^*	$f(x_k^*)$
1	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 5 \\ 5 \\ 5 \end{pmatrix}$	460.0000	$\begin{pmatrix} -3 \\ 5 \\ -3 \\ 5 \\ -3 \\ 5 \\ -3 \\ -3 \\ -3 \\ 5 \end{pmatrix}$	-2432.0000
2	$\begin{pmatrix} -2 \\ -2 \\ -2 \\ -2 \\ -2 \\ -2 \\ -2 \\ 5 \\ 5 \\ 5 \end{pmatrix}$	-50.0000	$\begin{pmatrix} -3 \\ -3 \\ -3 \\ -3 \\ -3 \\ 5 \\ -3 \\ -3 \\ -3 \\ -3 \end{pmatrix}$	-2432.0000
3	$\begin{pmatrix} 2 \\ 2 \\ 1 \\ -1 \\ -1 \\ 0 \\ 3 \\ -3 \\ 5 \\ 5 \end{pmatrix}$	213.0000	$\begin{pmatrix} -3 \\ -3 \\ -3 \\ -3 \\ -3 \\ 5 \\ -3 \\ -3 \\ -3 \\ -3 \end{pmatrix}$	-2432.0000
4	$\begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ -1 \\ 5 \\ 5 \\ -3 \end{pmatrix}$	1266.0000	$\begin{pmatrix} 5 \\ -1 \\ -3 \\ 5 \\ -3 \\ 5 \\ -3 \\ -3 \\ 5 \\ -3 \end{pmatrix}$	-2224.0000

continue goes here…

k	x_k	$f(x_k)$	x_k^*	$f(x_k^*)$
	$\begin{pmatrix} -3 \\ 5 \\ -3 \\ 5 \\ -3 \\ 5 \\ -3 \\ -3 \\ -3 \\ -3 \end{pmatrix}$	-2432.0000	$\begin{pmatrix} -3 \\ 5 \\ -3 \\ 5 \\ -3 \\ 5 \\ -3 \\ -3 \\ -3 \\ -3 \end{pmatrix}$	-2432.0000
5	$\begin{pmatrix} 5 \\ 5 \\ -3 \\ 5 \\ 5 \\ -3 \\ -3 \\ 5 \\ 5 \\ -3 \end{pmatrix}$	1376.0000	$\begin{pmatrix} 5 \\ -3 \\ 5 \\ -1 \\ -3 \\ 5 \\ -3 \\ -3 \\ 5 \\ -3 \end{pmatrix}$	-2128.0000
	$\begin{pmatrix} -3 \\ 5 \\ -3 \\ 5 \\ -3 \\ 5 \\ -3 \\ -3 \\ -3 \\ 5 \end{pmatrix}$	-2432.0000	$\begin{pmatrix} -3 \\ 5 \\ -3 \\ 5 \\ -3 \\ 5 \\ -3 \\ -3 \\ -3 \\ -3 \end{pmatrix}$	-2432.0000
6	$\begin{pmatrix} 1 \\ -1 \\ 2 \\ -2 \\ 3 \\ -3 \\ 0 \\ -3 \\ -3 \\ -3 \end{pmatrix}$	-415.0000	$\begin{pmatrix} 5 \\ -3 \\ 5 \\ -3 \\ 5 \\ -0.4444 \\ 5 \\ -3 \\ 5 \\ -3 \end{pmatrix}$	-2091.1000
	$\begin{pmatrix} -3 \\ 5 \\ -3 \\ 5 \\ -3 \\ 5 \\ -3 \\ -3 \\ -3 \\ 5 \end{pmatrix}$	-2432.0000	$\begin{pmatrix} -3 \\ 5 \\ -3 \\ 5 \\ -3 \\ 5 \\ -3 \\ -3 \\ -3 \\ -3 \end{pmatrix}$	-2432.0000

2.5. Conclusion

Cubic polynomial programming problems with mixed variables (MCP) are considered in this chapter. We proposed a necessary local optimality condition for general problems with mixed variables and proposed necessary local and global optimality conditions for (MCP). As well-known, the traditional local optimization methods are proposed according to KKT conditions for optimization problems with continuous variables. In this chapter, we designed a weakly local optimization method for general problems with mixed variables according to the necessary local optimality condition and a strongly local optimization method for (MCP) according to the necessary global optimality condition. Moreover, a novel global optimization method has been designed to solve (MCP) by combining local optimization methods together with an auxiliary function.

Chapter 3.

Global optimality conditions and optimization methods for quartic programming problems $(QPOP)$

In this chapter multivariate quartic programming problems (QPOP) are considered. Problems (QPOP) arise in various practical applications and are proved to be NP-hard. We discuss a necessary global optimality condition for the problem (QPOP). Then we present a new (strongly or $\varepsilon-$strongly) local optimization method according to the necessary global optimality condition, which may escape and improve some KKT points. Finally we design a global optimization method for the problem (QPOP) by combining the new (strongly or $\varepsilon-$strongly) local optimization method and an auxiliary function. Numerical examples show that our algorithms are efficient and stable.

3.1. Introduction

In this chapter, we consider the following fourth order (quartic) polynomial programming problems:

$$
\begin{aligned}
(QPOP) \qquad \min \quad & f(x) = \sum_{\substack{i,j,k,l=0 \\ j \geq i, k \geq j, l \geq k}}^{n} c_{ijkl} x_i x_j x_k x_l \\
s.t. \quad & x_i \in [u_i, v_i], i = 1, \ldots, n,
\end{aligned}
\tag{3.1}
$$

where $x_0 \equiv 1$, $u_i, v_i, c_{ijkl} \in R$ and $u_i < v_i \ for \ any \ i = 1, \ldots, n$, n is a positive integer number. Throughout of this chapter, let $X := \{(x_1, \ldots, x_n)^T \mid x_i \in [u_i, v_i], i = 1, \ldots, n\}$. The motivation is from two aspects. One is that problems (QPOP) have a wide range of practical applications. To take describing complicated objects for example, previous research was confined to fitting curves in the plane and surfaces in 3-D with conics, e.g., implicit polynomials of degree 2 which are restricted [33]. the authors in [33] justified fourth-degree polynomials for 2-D curves and 3-D surfaces and illustrated that a nice range of shapes that can be represented by fourth-degree implicit polynomials. In [88], Qi and Teo raised the concept of normal polynomial and showed that the multivariate polynomials resulting from signal processing [4], [61], [62], [92] are normal quartic polynomials. Furthermore, the author formulated the sensor network localization problem as finding the global minimizer of a quartic polynomial in [78]. Another example is, many digital communications schemes involve the transmission of constant modulus (CM) signals; hence, several schemes for blind equalization of CM signals have been developed. The direct formulation of the CM equalization problem is a fourth-order multivariate polynomial [15]. In addition, Martin L. Hazelton presented a new model for estimation of origin-destination (O-D) matrices which was actually a quartic polynomial problem on [96]. More examples are referred to [1], [89], [102], [139]. Another motivation is that as is well-known, the polynomial programming problem is NP-hard even when degree is fixed to be four [137], [139].

As special cases of polynomial programming problems, problems (QPOP) have attracted much attention recently, see [1], [89], [139] and [140]. Paper [1] focused on a question that has been open since 1992 when N. Z. Shor asked for the complexity of deciding convexity for quartic polynomials. [1] showed that deciding convexity of polynomials is strongly NP-hard already for polynomials of degree 4. Paper [89] designed a global descent algorithm for normal quartic polynomials to find a global minimizer $(n = 2)$ or an ϵ-global minimizer $(n \geq 3)$. Furthermore, paper [140] extended the global descent algorithm to general normal polynomials. Paper [139] presented a general semidefinite relaxation scheme for quartic homogeneous polynomial optimization under quadratic constraints by using a matrix listing transformation $X = xx^T$ to relax the quartic programming problem with quadratic constraints to a quadratic programming problem with linear constraints.

After we presented necessary global optimality conditions and designed optimization methods for cubic polynomial optimization problems with mixed variables in chapter 3, we try to develop a necessary global optimality condition and optimization methods for the problem (QPOP) in this chapter. We will first discuss a necessary global optimality condition. If a point is a global minimizer, then it is not only a KKT point, but also a global minimizer along any direction. Some specific directions are obtained by using some linear transformations. Along these special directions, the objective function can be simplified into univariate polynomial functions. Obviously, we could easily obtain a global minimizer for a fourth degree univariate polynomial function. Since traditional local optimization method are designed based on KKT conditions, we will present a new (strongly or $\varepsilon-$strongly) local optimization method based on the necessary global optimality condition which may improve some KKT points. Finally, we will design a global optimization method to solve the problem (QPOP) by combining the new local optimization method and an auxiliary function. Numerical examples illustrate the efficiency of the optimization methods proposed in the chapter.

3.2. Necessary global optimality condition for $(QPOP)$

In this section, we will derive a necessary condition for the problem $(QPOP)$.

Definition 16. *[100] Consider the problem of minimizing $f(x)$ over feasible set S, and let $\bar{x} \in S$. If $f(\bar{x}) \leq f(x)$ for all $x \in S$, $\bar{x}$ is called a global minimum. If there exists an $\delta-$neighborhood $N_\delta(\bar{x}) \subset S$ around $\bar{x}$ such that $f(\bar{x}) \leq f(x)$ for each $x \in N_\delta(\bar{x})$, $\bar{x}$ is called a local minimum.*

Remark 5. *Let $\bar{x} \in S$ be a local minimizer of the problem $(QPOP)$. Then the following KKT necessary condition holds: for any $i = 1, \cdots, n$, $\exists \lambda_i \geq 0$ and $\mu_i \geq 0$, such that*

$$\begin{aligned} (\nabla f(\bar{x}))_i + \lambda_i - \mu_i &= 0 \\ \lambda_i(\bar{x}_i - v_i) &= 0 \\ \mu_i(\bar{x}_i - u_i) &= 0 \end{aligned}$$

which is equivalent to

$$[KKT] \qquad \widetilde{\bar{x}}_i(\nabla f(\bar{x}))_i \leq 0,\ i = 1, \cdots, n.$$

where

$$\widetilde{\bar{x}}_i : = \begin{cases} -1, & \text{if } \bar{x}_i = u_i \\ 1, & \text{if } \bar{x}_i = v_i \\ \mathrm{sign}(\nabla f(\bar{x}))_i, & \text{if } u_i < \bar{x}_i < v_i \end{cases},$$

and $\operatorname{sign}((\nabla f(\bar{x}))_i) = \begin{cases} -1, & (\nabla f(\bar{x}))_i < 0 \\ 0, & (\nabla f(\bar{x}))_i = 0 \\ 1, & (\nabla f(\bar{x}))_i > 0 \end{cases}$.

In the following, we will give a necessary global optimality condition for the problem $(QPOP)$. If a point $\bar{x}$ is a global minimizer, then it is not only a KKT point, but also a global minimizer on any line through $\bar{x}$ and within the feasible set X. Some specific lines are obtained by using linear transformations. On these special lines, the objective function can be simplified into univariate quartic functions. Then, we try to find the global minimizers for these univariate quartic functions.

Before we present the necessary global optimality condition, we give lemma 1 for univariate quartic functions.

Let $\psi(y) = a(y-\bar{y})^4 + b(y-\bar{y})^3 + c(y-\bar{y})^2 + d(y-\bar{y})$, $y, \bar{y} \in [l, r]$, where l and r are given real numbers and $l \le r$. We give some notations.

$$\tilde{\bar{y}} : = \begin{cases} -1, & \text{if } \bar{y} = l \\ 1, & \text{if } \bar{y} = r \\ \operatorname{sign}(d), & \text{if } l < \bar{y} < r \end{cases},$$

$$\theta := \begin{cases} \min\left\{a(l-\bar{y})^2 + b(l-\bar{y}) + c, a(r-\bar{y})^2 + b(r-\bar{y}) + c\right\}, \bar{y} \in (l, r) \\ a(r-l)^3 - \tilde{\bar{y}}b(r-l)^2 + c(r-l), \ otherwise \end{cases}$$

$$h(y) := a(y-\bar{y})^3 + b(y-\bar{y})^2 + c(y-\bar{y}).$$

$$\xi := \min \begin{Bmatrix} -\tilde{\bar{y}}h(Y_1) \text{ if } \ Y_1 \in (l, r); \quad -\tilde{\bar{y}}h(Y_2) \text{ if } \ Y_2 \in (l, r); \\ a(r-l)^3 - \tilde{\bar{y}}b(r-l)^2 + c(r-l) \end{Bmatrix}$$

$$\alpha := \begin{cases} \widetilde{\bar{y}}\frac{c^2}{4b}, & \text{if } a = 0, Y_3 \in (l, r) \text{ and } \bar{y} = l \text{ or } r \\ \xi, & \text{if } a \neq 0, \Delta \geq 0 \text{ and } \bar{y} = l \text{ or } r \\ \frac{4ac-b^2}{4a}, & \text{if } a > 0, Y_4 \in (l, r) \text{ and } \bar{y} \in (l, r) \\ \theta, & \text{otherwise} \end{cases},$$

where $Y_1 = \bar{y} + \frac{-2b+\sqrt{\Delta}}{6a}$ and $Y_2 = \bar{y} + \frac{-2b-\sqrt{\Delta}}{6a}$, where $\Delta = 4b^2 - 12ac$. $Y_3 = \bar{y} - \frac{c}{2b}$ and $Y_4 = \bar{y} - \frac{b}{2a}$.

Lemma 1. *$\psi(y) \geq 0$, $\forall y \in [l, r]$ if and only if*

$$\widetilde{\bar{y}}d \leq \min\{0, \alpha\}.$$

Proof: Let

$$\psi(y) = a(y - \bar{y})^4 + b(y - \bar{y})^3 + c(y - \bar{y})^2 + d(y - \bar{y}) \geq \quad 0, \ \forall y \in [l, r] \tag{3.2}$$

We prove that (3.2) is equivalent to

$$\widetilde{\bar{y}}d \leq \min\{0, \alpha\}.$$

by considering the following three cases: $\bar{y} = l$, $\bar{y} = r$ and $l < \bar{y} < r$.

1°. If $\bar{y} = l$, then $y - \bar{y} \geq 0$ and (3.2) is equivalent to

$$\begin{aligned} & a(y - \bar{y})^3 + b(y - \bar{y})^2 + c(y - \bar{y}) + d \geq 0, \forall y \in [l, r], \\ \Leftrightarrow \quad & -d \leq a(y - \bar{y})^3 + b(y - \bar{y})^2 + c(y - \bar{y}), \forall y \in [l, r], \\ \Leftrightarrow \quad & -d \leq \min_{y \in [l, r]} \{a(y - \bar{y})^3 + b(y - \bar{y})^2 + c(y - \bar{y})\} \end{aligned}$$

Let $h(y) = a(y - \bar{y})^3 + b(y - \bar{y})^2 + c(y - \bar{y})$. The minimum of polynomial $h(y)$ lies on either the stationary points (roots of derivative) or the endpoints (l and r).

When $a \neq 0$, the stationary points are $Y_1 = \bar{y} + \frac{-2b+\sqrt{\Delta}}{6a}$ and $Y_2 = \bar{y} + \frac{-2b-\sqrt{\Delta}}{6a}$, where $\Delta = 4b^2 - 12ac \geq 0$. The function values are $-\tilde{\bar{y}}h(Y_1)$ and $-\tilde{\bar{y}}h(Y_2)$.

When $a = 0$, let $p(y) = b(y-\bar{y})^2 + c(y-\bar{y})$, the stationary point is $Y_3 = \bar{y} - \frac{c}{2b}$ and $p(Y_3) = -\frac{c^2}{4b}$.

The function values of $h(y)$ at the endpoints are 0 and $a(r-l)^3 - \tilde{\bar{y}}b(r-l)^2 + c(r-l)$.

$2°$. If $\bar{y} = r$, then $y - \bar{y} \leq 0$ and (3.2) is equivalent to

$$\begin{aligned}
& a(y-\bar{y})^3 + b(y-\bar{y})^2 + c(y-\bar{y}) + d \leq 0, \forall y \in [l,r], \\
\Leftrightarrow\ & d \leq -a(y-\bar{y})^3 - b(y-\bar{y})^2 - c(y-\bar{y}), \forall y \in [l,r], \\
\Leftrightarrow\ & d \leq \min_{y\in[l,r]} \{-a(y-\bar{y})^3 - b(y-\bar{y})^2 - c(y-\bar{y})\}
\end{aligned}$$

We can get the same results in a similar way to case 1.

$3°$. If $l < \bar{y} < r$, then (3.2) is equivalent to

$$\begin{aligned}
& \begin{cases} a(y-\bar{y})^3 + b(y-\bar{y})^2 + c(y-\bar{y}) + d \leq 0, & \forall y \in [l,\bar{y}) \\ a(y-\bar{y})^3 + b(y-\bar{y})^2 + c(y-\bar{y}) + d \geq 0, & \forall y \in (\bar{y},r] \end{cases} \\
\Leftrightarrow\ & \begin{cases} d = \frac{\partial g(\bar{y})}{\partial y} = 0, & \\ q(y) := a(y-\bar{y})^2 + b(y-\bar{y}) + c \geq 0, & \forall y \in [l,r], y \neq \bar{y} \end{cases}
\end{aligned}$$

$$\Leftrightarrow \begin{cases} d = \frac{\partial g(\bar{y})}{\partial y} = 0, \\ \min\limits_{y\in[l,r]} q(y) \geq 0. \end{cases}$$

When $a \neq 0$, the stationary point of $q(y)$ is $Y_4 = \bar{y} - \frac{b}{2a}$ and $p(Y_4) = \frac{4ac-b^2}{4a}$.

The function values of $q(y)$ at the endpoints are $a(l-\bar{y})^2 + b(l-\bar{y}) + c$ and $a(r-\bar{y})^2 + b(r-\bar{y}) + c$.

From the above analysis, we can see (3.2) is equivalent to that

$$\widetilde{\overline{y}}d \leq \min\{0, \alpha\}.$$

$\square$

Next, we present a necessary global optimality condition for the problem $(QPOP)$. Let $\bar{x} \in X$, Q be an invertible matrix, let

$$x := Qy, \;\; g(y) := f(Qy) = f(x), \;\; \bar{y} := Q^{-1}\bar{x},$$

and let $(Q)_i$ represent the ith row of Q, $(Q)_{ij}$ represent the entry of Q in the ith row and the jth column. Then,

$$\begin{aligned}
\frac{\partial g(y)}{\partial y_i} &= (Q)_i \nabla f(x), \\
\frac{\partial^2 g(y)}{\partial y_i^2} &= \sum_{r=1}^{n}\sum_{j=1}^{n}(Q)_{ji}(Q)_{ri}\frac{\partial^2 f(x)}{\partial x_j \partial x_r}, \\
\frac{\partial^3 g(y)}{\partial y_i^3} &= \frac{\partial(\frac{\partial^2 g(y)}{\partial y_i^2})}{\partial y_i} \\
&= \sum_{k=1}^{n}\frac{\partial(\sum\limits_{r=1}^{n}\sum\limits_{j=1}^{n}(Q)_{ji}(Q)_{ri}\frac{\partial^2 f(x)}{\partial x_j \partial x_r})}{\partial x_k}\frac{\partial x_k}{\partial y_i} \\
&= \sum_{k=1}^{n}\sum_{r=1}^{n}\sum_{j=1}^{n}(Q)_{ji}(Q)_{ri}(Q)_{ki}\frac{\partial^3 f(x)}{\partial x_j \partial x_r \partial x_k}.
\end{aligned}$$

$$\frac{\partial^4 g(y)}{\partial y_i^4} = 24 \sum_{\substack{j,l,r,k=0 \\ l\geq j, r\geq l, k\geq r}}^{n} c_{jlrk}(Q)_{ji}(Q)_{li}(Q)_{ri}(Q)_{ki}.$$

Let

$$\begin{aligned}
d_i : &= \frac{\partial g(\bar{y})}{\partial y_i} = (Q)_i \nabla f(\bar{x}), && (3.3)\\
c_i : &= \frac{1}{2}\frac{\partial^2 g(\bar{y})}{\partial y_i^2} = \frac{1}{2}\sum_{r=1}^{n}\sum_{j=1}^{n}(Q)_{ji}(Q)_{ri}\frac{\partial^2 f(\bar{x})}{\partial x_j \partial x_r}, && (3.4)\\
b_i : &= \frac{1}{6}\frac{\partial^3 g(\bar{y})}{\partial y_i^3} = \frac{1}{6}\sum_{k=1}^{n}\sum_{r=1}^{n}\sum_{j=1}^{n}(Q)_{ji}(Q)_{ri}(Q)_{ki}\frac{\partial^3 f(\bar{x})}{\partial x_j \partial x_r \partial x_k}, && (3.5)\\
a_i : &= \frac{1}{24}\frac{\partial^4 g(\bar{y})}{\partial y_i^4} = \sum_{\substack{j,l,r,k=0\\ l\geq j, r\geq l, k\geq r}}^{n} c_{jlrk}(Q)_{ji}(Q)_{li}(Q)_{ri}(Q)_{ki}. && (3.6)
\end{aligned}$$

Let $Y = \{y = Q^{-1}x | x \in \prod_{i=1}^{n}[u_i, v_i]\}$. For $\bar{y} = (\bar{y}_1, \ldots, \bar{y}_n)^T$, let $y = (\bar{y}_1, \cdots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \cdots, \bar{y}_n)^T$ and $x = Qy$. By $x = Qy \in X = \prod_{i=1}^{n}[u_i, v_i]$, we can obtain that

$$\begin{gathered}
u_1 - \sum_{\substack{j=1\\ j\neq i}}^{n}(Q)_{1j}\bar{y}_j \leq (Q)_{1i}y_i \leq v_1 - \sum_{\substack{j=1\\ j\neq i}}^{n}(Q)_{1j}\bar{y}_j,\\
\vdots\\
u_i - \sum_{\substack{j=1\\ j\neq i}}^{n}(Q)_{ij}\bar{y}_j \leq (Q)_{ii}y_i \leq v_i - \sum_{\substack{j=1\\ j\neq i}}^{n}(Q)_{ij}\bar{y}_j,\\
\vdots\\
u_n - \sum_{\substack{j=1\\ j\neq i}}^{n}(Q)_{nj}\bar{y}_j \leq (Q)_{ni}y_i \leq v_n - \sum_{\substack{j=1\\ j\neq i}}^{n}(Q)_{nj}\bar{y}_j.
\end{gathered}$$

Let $\triangle_k = \sum_{\substack{j=1\\ j\neq i}}^{n}(Q)_{kj}\bar{y}_j = \bar{x}_k - (Q)_{ki}\bar{y}_i = \bar{x}_k - (Q)_{ki}(Q^{-1})_i\bar{x}$, $k = 1, \cdots, n$, and let

$$\begin{aligned}
l_i &= \max\left\{\min\left\{\frac{u_1 - \triangle_1}{(Q)_{1i}}, \frac{v_1 - \triangle_1}{(Q)_{1i}}\right\}, \cdots, \min\left\{\frac{u_n - \triangle_n}{(Q)_{ni}}, \frac{v_n - \triangle_n}{(Q)_{ni}}\right\}\right\}, && (3.7)\\
r_i &= \min\left\{\max\left\{\frac{u_1 - \triangle_1}{(Q)_{1i}}, \frac{v_1 - \triangle_1}{(Q)_{1i}}\right\}, \cdots, \max\left\{\frac{u_n - \triangle_n}{(Q)_{ni}}, \frac{v_n - \triangle_n}{(Q)_{ni}}\right\}\right\}. && (3.8)
\end{aligned}$$

Then we can obtain the following results:

$$\begin{aligned}&(1) \quad l_i \leq r_i\\&(2) \quad [l_i, r_i] = \{y_i \mid (\bar{y}_1, \cdots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \cdots, \bar{y}_n)^T \in Y\}.\end{aligned}$$

In fact, (1) for the given $\bar{x} \in X$, let $\bar{y} = (\bar{y}_1, \cdots, \bar{y}_{i-1}, \bar{y}_i, \bar{y}_{i+1}, \cdots, \bar{y}_n)^T = Q\bar{x}$, by the discussion above, we have that $l_i \leq \bar{y}_i \leq r_i$. Hence, $l_i \leq r_i$;
(2) for any $y_i \in [l_i, r_i]$, let $y = (\bar{y}_1, \cdots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \cdots, \bar{y}_n)^T$. By the discussion above, we have that $x = Qy \in X$, i.e., $y \in Y$.
For any $y_i \in \{y_i \mid (\bar{y}_1, \cdots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \cdots, \bar{y}_n)^T \in Y\}$, let $y = (\bar{y}_1, \cdots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \cdots, \bar{y}_n)^T$. Then $x = Qy \in X$. By the discussion above, we have that $l_i \leq y_i \leq r_i$.
For convenience, here respective to the invertible matrix Q, we give some similar notations as those given before lemma 1.

$$\tilde{\bar{x}}_i : \ = \begin{cases} -1, & \text{if } (Q^{-1})_i\bar{x} = l_i \\ 1, & \text{if } (Q^{-1})_i\bar{x} = r_i \\ \text{sign}(d_i), & \text{if } l_i < (Q^{-1})_i\bar{x} < r_i \end{cases},$$

$$\theta_i := \begin{cases} \min\left\{ \begin{array}{l} a_i(l_i - (Q^{-1})_i\bar{x})^2 + b_i(l_i - (Q^{-1})_i\bar{x}) + c_i, \\ a_i(r_i - (Q^{-1})_i\bar{x})^2 + b_i(r_i - (Q^{-1})_i\bar{x}) + c_i \end{array} \right\}, & (Q^{-1})_i\bar{x} \in (l_i, r_i) \\ a_i(r_i - l_i)^3 - \tilde{\bar{x}}_i b_i(r_i - l_i)^2 + c_i(r_i - l_i), & otherwise \end{cases}$$

$$h_i(y_i) := a_i(y_i - (Q^{-1})_i\bar{x})^3 + b_i(y_i - (Q^{-1})_i\bar{x})^2 + c_i(y_i - (Q^{-1})_i\bar{x}).$$

$$\xi_i := \min \left\{ \begin{array}{l} -\tilde{\bar{x}}_i h(Y_{1,i}) \text{ if } Y_{1,i} \in (l_i, r_i); \\ -\tilde{\bar{x}}_i h(Y_{2,i}) \text{ if } Y_{2,i} \in (l_i, r_i); \\ a_i(r_i - l_i)^3 - \tilde{\bar{x}}_i b_i (r_i - l_i)^2 + c_i(r_i - l_i) \end{array} \right\}$$

$$\alpha_i : \;= \begin{cases} \tilde{\bar{x}} \frac{c_i^2}{4b_i}, & \text{if } a_i = 0, Y_{3,i} \in (l_i, r_i) \text{ and } (Q^{-1})_i \bar{x} = l_i \text{ or } r_i \\ \xi_i, & \text{if } a_i \neq 0, \Delta_i \geq 0 \text{ and } (Q^{-1})_i \bar{x} = l_i \text{ or } r_i \\ \frac{4a_i c_i - b_i^2}{4a_i}, & \text{if } a_i > 0, Y_{4,i} \in (l_i, r_i) \text{ and } (Q^{-1})_i \bar{x} \in (l_i, r_i) \\ \theta_i, & \text{otherwise} \end{cases},$$

where $Y_{1,i} = (Q^{-1})_i \bar{x} + \frac{-2b_i + \sqrt{\Delta_i}}{6a_i}$ and $Y_{2,i} = (Q^{-1})_i \bar{x} + \frac{-2b_i - \sqrt{\Delta_i}}{6a_i}$, where $\Delta_i = 4b_i^2 - 12a_i c_i$. $Y_{3,i} = (Q^{-1})_i \bar{x} - \frac{c_i}{2b_i}$ and $Y_{4,i} = (Q^{-1})_i \bar{x} - \frac{b_i}{2a_i}$.

Theorem 7. (Necessary global optimality condition for $(QPOP)$**)** *Let $\bar{x} \in S$ and Q be any given invertible matrix. If $\bar{x}$ is a global minimizer of $(QPOP)$, then for any $i = 1, \ldots, n$, the following conditions hold:*

$$[GNC]_i \quad \tilde{\bar{x}}_i d_i \leq \min\{0, \alpha_i\}.$$

Proof: Let $\bar{x}$ be a global minimizer of the problem $(QPOP)$. Let $\bar{y} = Q^{-1}\bar{x}$. Then for any $i = 1, \ldots, n$, let $y = (\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T$, $y_i \in [l_i, r_i]$ and $x = Qy$, we have that $x \in X$. Hence, $f(x) - f(\bar{x}) \geq 0$. Furthermore,

$$\begin{aligned} & f(x) - f(\bar{x}) \\ = \; & \frac{1}{24} \frac{\partial^4 g(\bar{y})}{\partial y_i^4} (y_i - \bar{y}_i)^4 + \frac{1}{6} \frac{\partial^3 g(\bar{y})}{\partial y_i^3} (y_i - \bar{y}_i)^3 + \frac{1}{2} \frac{\partial^2 g(\bar{y})}{\partial y_i^2} (y_i - \bar{y}_i)^2 + (\nabla g(\bar{y}))_i (y_i - \bar{y}_i) \\ = \; & a_i (y_i - \bar{y}_i)^4 + b_i (y_i - \bar{y}_i)^3 + c_i (y_i - \bar{y}_i)^2 + d_i (y_i - \bar{y}_i), \end{aligned}$$

where a_i, b_i, c_i and d_i are defined by (3.6), (3.5), (3.4) and (3.3). By Lemma 1, $f(x) - f(\bar{x}) =$

$a_i(y_i-\bar{y}_i)^4+b_i(y_i-\bar{y}_i)^3+c_i(y_i-\bar{y}_i)^2+d_i(y_i-\bar{y}_i)\geq 0, \forall y_i\in[l_i,r_i]$ if and only if $[GNC]_i$ holds. $\square$

Remark 6. *If $Q=I$, where I is the identity matrix, then a_i,b_i,c_i,d_i,l_i,r_i, and $(Q^{-1})_i\bar{x}$ given in the condition $[GNC]_i$ are determined as following:*

$$\begin{array}{rcl} d_i &=& \dfrac{\partial f(\bar{x})}{\partial x_i}, \\ c_i &=& \dfrac{1}{2}\dfrac{\partial^2 f(\bar{x})}{\partial x_i^2}, \\ b_i &=& \dfrac{1}{6}\dfrac{\partial^3 f(\bar{x})}{\partial x_i^3}, \\ a_i &=& \dfrac{1}{24}\dfrac{\partial^4 f(\bar{x})}{\partial x_i^4}, \\ l_i &=& u_i, \\ r_i &=& v_i, \\ (Q^{-1})_i\bar{x} &=& \bar{x}_i. \end{array}$$

Remark 7. (1) *If the problem $(QPOP)$ reduces to a cubic problem, i.e., no 4th order terms in $(QPOP)$, then for any $i=1,\ldots,n$, $[GNC]_i$ transform to*

$$\tilde{\bar{x}}_i d_i \leq \min\{0,\alpha_i\}. \tag{3.9}$$

where

$$\alpha_i :\ = \begin{cases} \tilde{\bar{x}}_i \frac{c_i^2}{4b_i}, & \text{if } Y_{3,i}\in(l_i,r_i) \text{ and } (Q^{-1})_i\bar{x}=l_i \text{ or } r_i \\ \theta_i, & \text{otherwise} \end{cases}$$

$a_i = 0$, b_i *is the coefficient of* x_i^3 *and*

$$\theta_i := \begin{cases} \min\left\{\begin{array}{l} b_i(l_i - (Q^{-1})_i\bar{x}) + c_i \\ b_i(r_i - (Q^{-1})_i\bar{x}) + c_i \end{array}\right\}, \ (Q^{-1})_i\bar{x} \in (l_i, r_i) \\ -\bar{\bar{x}}_i b_i (r_i - l_i)^2 + c_i(r_i - l_i), \ otherwise \end{cases}$$

Others just remain the same. We can see the condition (3.9) extends the condition given in Corollary 2.3 in reference [144] which is just the special case of (3.9) when $Q = I$.

(2) *If the problem* $(QPOP)$ *reduces to a quadratic problem, i.e., nether 4th nor 3th order terms in* $(QPOP)$*, then for any* $i = 1, \ldots, n$, $[GNC]_i$ *transform to*

$$\widetilde{\bar{x}}_i d_i \leq \min\{0, \alpha_i\}. \tag{3.10}$$

where

$$\alpha_i := \theta_i$$

$a_i = b_i = 0$, c_i *is the coefficient of* x_i^2 *and*

$$\theta_i := \begin{cases} c_i \geq 0, \ (Q^{-1})_i\bar{x} \in (l_i, r_i) \\ c_i(r_i - l_i), \ otherwise \end{cases}$$

Others just remain the same. We can see the condition (3.10) extends the condition $[NC1]$ *given in Theorem 3.7 in reference [145] which is just the special case of (3.10) when* $Q = I$ *if we just consider the continuous variables other than discrete variables.*

(3) *Obviously, when* $Q = I$*, for any* $i = 1, \ldots, n$*, conditions* $[GNC]_i$ *include*

$$\widetilde{\bar{x}}_i d_i \leq 0,$$

which is the $[KKT]$ condition.

3.3. Optimization methods for $(QPOP)$

3.3.1. Strongly or $\varepsilon-$strongly local optimization method for $(QPOP)$

In this subsection, we will design a new local optimization method (called strongly or $\varepsilon-$strongly local optimization method) for the problem $(QPOP)$ according to the necessary global optimality condition $[GNC]_i$ for any $i = 1, \cdots, n$.

Definition 17. *Let $\bar{x} \in X$ and Q be an invertible matrix. $\bar{x}$ is said to be a strongly local minimizer of the problem $(QPOP)$ with respect to Q iff $\bar{x}$ satisfies the necessary global optimality condition $[GNC]_i$ for any $i = 1, \cdots, n$.*

Definition 18. *Let $\bar{x} \in X$ and Q be an invertible matrix. $\bar{x}$ is said to be a $\varepsilon-$strongly local minimizer of the problem $(QPOP)$ with respect to Q iff for any $i = 1, \cdots, n$, either $\bar{x}$ satisfies the condition $[GNC]_i$ or there exists a point $x_i^* \in X$, such that x_i^* satisfies the condition $[GNC]_i$ and $|f(\bar{x}) - f(x_i^*)| \le \varepsilon$.*

Let $\bar{x} \in X$, Q be an invertible matrix, and let

$$N_i : \; = \; \{\bar{y} + z_i e_i \mid z_i \in \{l_i - \bar{y}_i, r_i - \bar{y}_i\} \setminus \{0\}\}, \text{ for } i = 1, \ldots, n, \tag{3.11}$$

$$\begin{aligned} P_i := \quad & \{Y_{1,i}, Y_{2,i} \mid a_i \neq 0, \triangle_i \ge 0, \bar{y}_i = l_i \text{ or } r_i\} \cup \\ & \{Y_{3,i} \mid a_i = 0, \bar{y}_i = l_i \text{ or } r_i\} \cup \\ & \{Y_{4,i} \mid a_i > 0, \bar{y}_i \in (l_i, r_i)\}, \end{aligned} \tag{3.12}$$

$$N_i' : \; = \; \{\bar{y} + (z_i - \bar{y}_i) e_i \mid z_i \in P_i \cap (l_i, r_i)\}, i = 1, \ldots, n, \tag{3.13}$$

where $\bar{y} = Q^{-1}\bar{x}$, l_i and r_i are defined by (3.7) and (3.8).

Note that $|N_i| \leq 2$ and $|N_i'| \leq 2$ for $i = 1, \ldots, n$, where $|N_i|$ and $|N_i'|$ means the number of the points in N_i and N_i'.

Remark 8. *From Theorem 7, we know that, for any given invertible matrix Q, $[GNC]_i$ is satisfied for any $i = 1, \ldots, n$. However, in our algorithm, we only randomly select N invertible matrices $Q_i, \cdots, Q_N$, and we always choose $Q_1 = I$, the identity matrix.*

Algorithm 6. *Strongly or $\varepsilon-$strongly local optimization method for $(QPOP)$:$(SLOM)$.*

Step 0. *Take an initial point $x_0 \in X$. Let $Q_1 = I$,Q_2, $\cdots$, Q_s, $\cdots$, Q_N be any invertible matrices given randomly, where I is the identity matrix. Let ε be a small positive number. Let $s := 1$, $Q := Q_s$ and $i = 1$. Let $x^* = (x_1^*, \cdots, x_n^*)$ be a local minimizer or KKT point of $f(x)$ on $\prod_{i=1}^{n}[u_i, v_i]$ starting from x_0. Let $\bar{x} := (x_1^*, \cdots, x_n^*)$, and go to Step 1;*

Step 1. *Check whether the condition holds:*

$$[GNC]_i \quad \tilde{\bar{x}}_i d_i \leq min\{0, \alpha_i\}$$

If this condition holds, go to Step 2; otherwise, go to Step 3;

Step 2. *If $i := n$, go to Step 4; otherwise, let $i := i + 1$ and go to Step 1;*

Step 3. *Let $\bar{y} = Q^{-1}\bar{x} = (\bar{y}_1, \ldots, \bar{y}_i, \ldots, \bar{y}_n)^T$ and $y = (\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T$. Let $\bar{y}_i^* := argmin\{f(Qy)|y \in N_i \bigcup N_i'\}$, where N_i and N_i' are defined by (3.11) and (3.13), respectively. Let $\bar{y}^* = (\bar{y}_1, \cdots, \bar{y}_{i-1}, \bar{y}_i^*, \bar{y}_{i+1} \cdots, \bar{y}_n)$. Let $\bar{x}^* := Q\bar{y}^*$. Let $x^* = (x_1^*, \cdots, x_n^*)$ be a local minimizer or KKT point of $f(x)$ on $\prod_{i=1}^{n}[u_i, v_i]$ starting from $\bar{x}^*$. If $f(x^*) < f(\bar{x}) - \varepsilon$, let $\bar{x} := x^*$, $i := 1$ and $s = 1$, go to Step 1; otherwise, let $i := i + 1$ and go to Step 1.*

Step 4. *Let $s := s + 1$. If $s > N$, go to Step 5; otherwise, let $Q := Q_s$ and $i := 1$, go to Step 1;*

Step 5. *Stop.* $\bar{x}$ *is a strongly or* $\varepsilon-$*strongly local minimizer with respect to* Q_s, $s = 1, \cdots, N$.

Theorem 8. *For a given initial point* $x_0 \in X$, *we can obtain a strongly or* $\varepsilon-$ *strongly local minimizer* $\bar{x}$ *of the problem* $(QPOP)$ *in finite iteration times by the given strongly local optimization method* $(SLOM)$.

Proof. First, we can prove that this algorithm must stop in finite iteration times.
Let $M := max\{f(x) \mid x \in X\}$ and $m := min\{f(x) \mid x \in X\}$. For the given Q_s, there are at most $n\frac{M-m}{\varepsilon}$ iteration times from *step 1* to *step 3*. In fact, for the given Q_s and given i, if $[GNC]_i$ holds or if $f(x^*) \geq f(\bar{x}) - \varepsilon$, then we will change the i into $i+1$; only when $[GNC]_i$ does not hold and $f(x^*) < f(\bar{x}) - \varepsilon$, we will change i to 1 in *step 3* and go to *step 1*. For the same Q_s, when we change i to 1, the objection function value will decrease at least ε. Hence, there are at most $\frac{M-m}{\varepsilon}$ times to change i to 1 in *step 3*. The total iteration time from *step 1* to *step 3* is at most $n\frac{M-m}{\varepsilon}$. Since we have N numbers of Q_s , this algorithm must stop at most $Nn\frac{M-m}{\varepsilon}$ iteration times.
Second, let L be the set of all the KKT points of the problem $(QPOP)$, and let $L_f := \{f(x) \mid x \in L\}$. We can prove that
(1) If L_f is a finite set, then we can obtain a strongly local minimizer in finite iteration times when ε is a very small number. In fact, let $\eta := min\{|f(x) - f(y)| \mid x, y \in L$ and $f(x) \neq f(y)\}$. Since L_f is a finite set, we have that $\eta > 0$. When $\varepsilon < \eta$, we know that $f(x^*) < f(\bar{x}) - \varepsilon$ in *step 3* is equivalent to $f(x^*) < f(\bar{x})$. Hence, for the given Q_s and given i, if $[GNC]_i$ holds, then we will change the i into $i+1$; if $[GNC]_i$ does not hold in *step 1* which means that $f(\bar{x}) > min\{f(Qy)|y \in N_i \bigcup N_i^{'}\}$, then in *step 3*, we will find a point $\bar{y}_i^*$ such that $f(Q\bar{y}^*) = min\{f(Qy)|y \in N_i \bigcup N_i^{'}\}$. Hence, we have that $f(x^*) < f(\bar{x})$ since $f(x^*) \leq f(Q\bar{y}^*) < f(\bar{x})$ and we have $x^* \in L$. Therefore, for the given Q_s and given i, if $[GNC]_i$ does not hold in *step 1*, then we can obtain a new KKT point x^* such that $f(x^*) < f(\bar{x})$ which also satisfies that $f(x^*) < f(\bar{x}) - \varepsilon$. Hence, for the given Q_s, we can find a point $\bar{x}$ which satisfies the condition $[GNC]_i, i = 1, \ldots, n$ in at most $n\frac{M-m}{\varepsilon}$ iteration

times. Therefore, in finite iteration times, we can obtain a strongly local minimizer of the problem $(QPOP)$ for all $Q_s, s = 1, \ldots, N$.

(2) If L_f is an infinite set, then we can obtain an $\varepsilon-$strongly local minimizer in finite iteration times.

By the algorithm, for the given Q_s and given i, if $[GNC]_i$ holds or if $f(x^*) \geq f(\bar{x}) - \varepsilon$, then we will change the i into $i+1$; if $[GNC]_i$ does not hold and $f(x^*) < f(\bar{x}) - \varepsilon$, then in *step 3*, we will find a point $\bar{y}_i^*$ such that $f(Q\bar{y}^*) = min\{f(Qy)|y \in N_i \bigcup N_i'\}$, where $\bar{y}_i^*$ satisfies condition $[GNC]_i$. Since this algorithm must stop in finite steps, the final obtained point $\bar{x}$ must satisfy the following condition: for the given Q_s and given i, $[GNC]_i$ holds or $f(Q\bar{y}^*) \geq f(x^*) \geq f(\bar{x}) - \varepsilon$, where $\bar{y}_i^*$ satisfies the condition $[GNC]_i$. Hence $\bar{x}$ is an $\varepsilon-$strongly local minimizer of the problem $(QPOP)$. □

Remark 9. *In Algorithm 6, in Step 0 and Step 3, it is very easy to obtain a KKT point or a local minimizer of $f(x)$ on $\prod_{i=1}^{n}[u_i, v_i]$, such as the Newton method, the Quasi-Newton method, the Conjugate gradient method and the line search method can be used here. In section 3.4, the optimization subroutine within the optimization Toolbox in Matlab is used to find a KKT point or a local minimizer. In step 3, it is easy to find the point $\bar{y}_i^*$ such that $\bar{y}_i^* = argmin\{f(Qy) \mid y_i \in N_i \bigcup N_i'\}$, since we can easily find N_i and N_i' by (3.11) and (3.13), respectively, and since $|N_i \bigcup N_i'| \leq 4$.*

3.3.2. Global optimization method for $(QPOP)$

In this subsection, we will design a global optimization method for the problem $(QPOP)$ by combining the strongly or $\varepsilon-$strongly local optimization method and an auxiliary function. The local optimization methods have been extensively developed. However the difficulty is how to escape a local minimizer to a better one. The filled function method is one of the well-known and practical methods to settle this difficulty. The filled function is used to escape the current local minimizer and to find a better feasible point. In this chapter, we will use the auxiliary function which was presented by (1.2) in Chapter 1. For the properties of this auxiliary function, see Chapter 1.

In the following, we will introduce a global optimization method to find a global minimizer of the problem $(QPOP)$. The procedure of this global optimization method in the following consists of three phase circle:

Phase 1: (Strongly Local Search) Start from a given feasible point x_k and use strongly local minimization method *Algorithm 6* to search for a strongly local minimizer x_k^*.

Phase 2: (Local Search) Construct auxiliary function $F_{q,r,c,x_k^*}(x)$. Find a KKT point or a local minimizer $\bar{x}_{q,r,c,x_k^*}$ of function $F_{q,r,c,x_k^*}(x)$.

Phase 3: (Global Search) If $\bar{x}_{q,r,c,x_k^*}$ is better than x_k^*, then let $k := k + 1$, $x_k := \bar{x}_{q,r,c,x_k^*}$ and return to Phase 1. Otherwise, stop the iteration process and return the incumbent local optimal solution x_k^* as a global optimal solution to the problem.

Algorithm 7. *Global optimization method for $(QPOP)$:(GOM).*

Step 0. *Set $M = 10^{10}$, $\mu := 10^{-10}$ and $k_0 = 2n$. Set $e_i = (0, \cdots, 0, 1, 0, \cdots, 0), i = 1, \ldots, n$, where the ith component is 1 and the others are 0, and $e_{n+i} = (0, \cdots, 0, -1, 0, \cdots, 0), i = 1, \ldots, n$, where the ith component is -1 and the others are 0. Let $r_0 := 1$, $c_0 := 1$, $q_0 := 10^5$, $\delta_0 := \frac{1}{2}$, $k := 1$, $i := 1$ and $r := r_0$. Let x_1^0 be an initial point and let $x_0^* := x_1^0$, go to Step 1;*

Step 1. *Use the strongly local optimization method $(SLOM)$ to solve the problem $(QPOP)$ starting from x_k^0. Let x_k^* be the obtained strongly or $\varepsilon-$strongly local minimizer of the problem $(QPOP)$. If $f(x_k^*) \geq f(x_0^*)$ ($k > 1$), then go to Step 5; otherwise (including $f(x_k^*) \geq f(x_0^*)$ when $k = 1$ or $f(x_k^*) < f(x_0^*)$ when $k \geq 1$) let $q := q_0$, $c := c_0$, $r := r_0$, $\delta := \delta_0$, $i := 1$ and $x_0^* = x_k^*$, $k := k + 1$, then go to Step 2;*

Step 2. *Let $\bar{x}_k^* := x_0^* + \delta e_i$. If $\bar{x}_k^* \notin S$, goto step 3. Otherwise, if $f(\bar{x}_k^*) < f(x_0^*)$, then set $x_{k+1}^0 := \bar{x}_k^*$, and $x_0^* := \bar{x}_k^*$, $k := k + 1$ and go to Step 1; else go to Step 4;*

Step 3. *If* $\delta < \mu$*, go to Step 8; otherwise, let* $\delta = \frac{\delta}{2}$ *and go to Step 2.*

Step 4. *If* $f(x_0^*) \leq f(\bar{x}_k^*) \leq f(x_0^*) + 1$*, then go to Step 5; otherwise let* $\delta = \frac{\delta}{2}$ *go to Step 2;*

Step 5. *Let*

$$F_{q,r,c,x_0^*}(x) = q\left(exp(-\frac{\|x - x_0^*\|^2}{q})g_{r,c}\Big(f(x) - f(x_0^*)\Big) + h_{r,c}\Big(f(x) - f(x_0^*)\Big)\right).$$

Solve the problem:

$$\begin{aligned} \min \quad & F_{q,r,c,x_0^*}(x) \\ s.t. \quad & x \in X \end{aligned}$$

starting from the initial point $\bar{x}_k^*$*. Let* $\bar{x}_{q,r,c,x_k^*}$ *be a KKT point or a local minimizer. Then set* $x_{k+1}^0 := \bar{x}_{q,r,c,x_k^*}$ *and go to Step 1;*

Step 6. *If* $q < M$*, then increase* q *(in the following examples, let* $q := 10q$*), then go to Step 5; otherwise go to Step 7;*

Step 7. *If* $c < M$*, then increase* c *(in the following examples, let* $c := 10c$*), and let* $q := q_0$*, then go to Step 5; otherwise go to Step 8;*

Step 8. *If* $i < k_0$*, then let* $i := i + 1$*,* $q := q_0$*,* $c := c_0$*,* $\delta := \delta_0$*, go to Step 2; otherwise go to Step 9;*

Step 9. *If* $r > \mu$*, then decrease* r *(in the following examples, let* $r := \frac{r}{10}$*), let* $i := 1$*,* $q := q_0$*,* $c := c_0$*,* $\delta := \delta_0$*, then go to Step 2; otherwise, stop and* x_0^* *is the obtained global minimizer or approximate global minimizer of the problem* $(QPOP)$*.*

Notes: The global optimization method applies the filled function method which belongs to a heuristic one. This method can gradually improve the current local minimizer. Although it cannot guarantee the result must be a global one, the numerical examples given in the following Section illustrate that the global minimization method Algorithm 7 is very efficient and stable.

3.4. Numerical examples

First, we apply our Algorithms to all examples $(n \geq 3)$ given in reference [89].

We notice although examples in [89] are unconstrained problems, they satisfy the following condition: $f(x) \to +\infty$ as $\|x\| \to +\infty$ in [149]. Hence the global minimizers must exist in a big enough box set. We changed all examples $(n \geq 3)$ in [89] to box constrained programming problems, say $x_i \in [-500, 500]$, $i = 1, \cdots, n$.

Only by *Algorithm* 6, can we solve all examples given in reference [89]. For Question 38 and 63, we can obtain better solutions than the 'global' minima given in reference [89] which shows that the strongly local optimization method *Algorithm* 6 is efficient (see Example 6 and Example 7).

Notations:

x_k: an initial point

$\bar{x}_k$: a local minimizer starting from x_k

$f(\bar{x}_k)$: the function value of $f(x)$ at $\bar{x}_k$

Q: the linear transformation matrix which can improve the local minimizer

$\bar{x}_k^*$: a strongly local minimizer starting from $\bar{x}$

$f(\bar{x}_k^*)$: the function value of $f(x)$ at $\bar{x}^*$

Notes: In *Algorithm* 6, when dimension of objective function is small, we take N a bit larger; when dimension is large, we take N a bit smaller. Such as, in Example 6, Example 7 and Example 8, we take $N = 20$; in Example 9 and Example 10, we take $N = 10$.

Example 6. *Consider the following problem ((*$Q63$*) in [89])*

$$\begin{aligned} \min \quad f(x) := \ & 9x_1^4 + 7x_2^4 + x_3^4 + 4x_4^4 + 9x_5^4 + 9x_6^4 + 8x_1^2 + 2x_1x_3 + 6x_1x_4 + 18x_1x_5 + \\ & 18x_1x_6 + 18x_2x_3 + 10x_2x_4 + 4x_2x_5 + 12x_2x_6 + 4x_3^2 + 2x_3x_4 + 2x_3x_5 + \\ & 16x_3x_6 + 16x_4x_5 + 2x_5^2 + 2x_5x_6 + 8x_6^2 + 5x_1 + 8x_2 + 6x_3 + 9x_4 + 9x_5 \\ s.t. \quad & \\ & x_1, x_2, x_3, x_4, x_5, x_6 \in [-500, 500]. \end{aligned}$$

Table 3.1 records the numerical results.

Table 3.1.: Numerical results for Example 6

k	x_k	$\bar{x}_k$	$f(\bar{x}_k)$	Q	$\bar{x}_k^*$	$f(\bar{x}_k^*)$
1	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -0.67779608 \\ 0.91575270 \\ -1.67672937 \\ -1.12932064 \\ 0.76949691 \\ 0.74098543 \end{pmatrix}$	-31.78036845	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} -0.67779608 \\ 0.91575270 \\ -1.67672937 \\ -1.12932064 \\ 0.76949691 \\ 0.74098543 \end{pmatrix}$	-31.78036845
2	$\begin{pmatrix} -500 \\ -500 \\ -500 \\ -500 \\ -500 \\ -500 \end{pmatrix}$	$\begin{pmatrix} 0.53687007 \\ -1.02589244 \\ 1.36531333 \\ 0.84526991 \\ -0.89744922 \\ -0.60054219 \end{pmatrix}$	-19.93119153	$\begin{pmatrix} -4 & -53 & 72 & 93 & -42 & 82 \\ 75 & -27 & 9 & 4 & 57 & 22 \\ -36 & 38 & -8 & -76 & -89 & 9 \\ 12 & -93 & -70 & 83 & -58 & 51 \\ -51 & 38 & -49 & -9 & 56 & 76 \\ -73 & -48 & -22 & 81 & 40 & 79 \end{pmatrix}$	$\begin{pmatrix} -0.67779608 \\ 0.91575270 \\ -1.67672937 \\ -1.12932064 \\ 0.76949691 \\ 0.74098543 \end{pmatrix}$	-31.78036845
3	$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} -0.36396908 \\ -1.02830653 \\ 0.56323224 \\ 0.94870966 \\ -0.70756437 \\ 0.47089712 \end{pmatrix}$	-16.27241850	$\begin{pmatrix} 61 & 89 & -83 & -93 & 7 & 90 \\ -59 & -52 & 2 & -47 & -86 & 51 \\ 32 & 40 & -50 & -40 & -70 & -95 \\ -42 & -72 & -25 & -59 & 37 & 49 \\ 37 & -11 & 18 & 27 & -33 & -33 \\ -38 & 29 & 100 & 43 & 69 & 61 \end{pmatrix}$		
		$\begin{pmatrix} 0.53687007 \\ -1.02589244 \\ 1.36531333 \\ 0.84526991 \\ -0.89744922 \\ -0.60054219 \end{pmatrix}$	-19.93119153	$\begin{pmatrix} -4 & -53 & 72 & 93 & -42 & 82 \\ 75 & -27 & 9 & 4 & 57 & 22 \\ -36 & 38 & -8 & -76 & -89 & 9 \\ 12 & -93 & -70 & 83 & -58 & 51 \\ -51 & 38 & -49 & -9 & 56 & 76 \\ -73 & -48 & -22 & 81 & 40 & 79 \end{pmatrix}$	$\begin{pmatrix} -0.67779608 \\ 0.91575270 \\ -1.67672937 \\ -1.12932064 \\ 0.76949691 \\ 0.74098543 \end{pmatrix}$	-31.78036845

The global minimizer given by reference [89] is (-0.363974062, -1.028303631, 0.563190492, 0.9486927097, -0.707559149, 0.470942714)^T with the optimal value -16.27241853.

Only by using *Algorithm* 6, we attain the global minimizer (-0.6778, 0.9158, -1.6766, -1.1294, 0.7695, 0.7410)^T with the optimal value -31.7804, which improves the results given in [89]

Example 7. *Consider the following problem ($(Q38)$ in [89])*

$$\begin{aligned}
\min \quad f(x) := \quad & 76x_1^4 + 172x_1^3x_2 + 176x_1^3x_3 + 285x_1^2x_2^2 + 247x_1^2x_3^2 + 360x_1^2x_2x_3 + \\
& 204x_1x_2^3 + 342x_1x_2^2x_3 + 420x_1x_2x_3^2 + 236x_1x_3^3 + 93x_2^4 + 182x_2^3x_3 + \\
& 293x_2^2x_3^2 + 182x_2x_3^3 + 126x_3^4 + 6 + 76x_1^3 - 57x_1^2x_2 - 80x_1^2x_3 - 92x_1^2 + \\
& 81x_1x_2^2 + 77x_1x_2x_3 - 87x_1x_2 + 50x_1x_3^2 + 74x_1x_3 - 60x_1 + 19x_2^3 - \\
& 68x_2^2x_3 + 78x_2^2 + 34x_2x_3^2 + 66x_2x_3 - 53x_2 + 59x_3^3 + 28x_3^2 + 38x_3 \\
s.t. \quad & \\
& x_1, x_2, x_3 \in [-500, 500]
\end{aligned}$$

Table 3.2 records the numerical results.

Table 3.2.: Numerical results for Example 7

k	x_k	$\bar{x}_k$	$f(\bar{x}_k)$	Q	$\bar{x}_k^*$	$f(\bar{x}_k^*)$
1	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0.7528 \\ 0.2874 \\ -0.8229 \end{pmatrix}$	-1.0224e+002	$\begin{pmatrix} -77 & -32 & 51 \\ 0 & 17 & -49 \\ 92 & -56 & 1 \end{pmatrix}$	$\begin{pmatrix} -7.2391 \\ 1.7116 \\ 5.0460 \end{pmatrix}$	$-1.7734e+004$
2	$\begin{pmatrix} -500 \\ -500 \\ -500 \end{pmatrix}$	$\begin{pmatrix} 0.7528 \\ 0.2874 \\ -0.8229 \end{pmatrix}$	-1.0224e+002	$\begin{pmatrix} -77 & -32 & 51 \\ 0 & 17 & -49 \\ 92 & -56 & 1 \end{pmatrix}$	$\begin{pmatrix} -7.2391 \\ 1.7116 \\ 5.0460 \end{pmatrix}$	$-1.7734e+004$
3	$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} -7.2391 \\ 1.7116 \\ 5.0460 \end{pmatrix}$	$-1.7734e+004$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} -7.2391 \\ 1.7116 \\ 5.0460 \end{pmatrix}$	$-1.7734e+004$

The global minimizer given by reference [89] is (0.752808377, 0.287362024, -0.822919492)^T with optimal value -102.236381.

Only by using *Algorithm* 6, we get global minimizer (-7.23913534, 1.71164243, 5.04604642)^T with the optimal value $-1.77339078e+004$, which improves the results given in [89]

Example 8. *Consider the problem: Dixon and Price Function [86]*

$$\begin{aligned} \min \quad & f(x) := (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2 \\ s.t. \quad & x_i \in [-10, 10],\ i = 1, 2, \cdots, n \end{aligned}$$

For $n = 5$, the optimal value of this function is 0 and this function has two global minimizers. By Algorithm *6, we obtain two minimizers $\bar{x}_1^*$ =(1.0000, 0.7071, 0.5946, 0.5452, -0.5221)^T and $\bar{x}_2^*$ =(1.0000, 0.7071, 0.5946, 0.5452, 0.5221)^T with the same function value* $7.7335e-009$*. Table 3.3 records the numerical results.*

For $n = 10$, the optimal value of this function is 0 and this function has two global minimizers. By Algorithm *6, we can not obtain the global minimizer. But by* Algorithm *7, we obtain two minimizers $\bar{x}_1^*$ =(1.0000, 0.7071, 0.5946, 0.5453, 0.5221, 0.5109, 0.5054, 0.5027, 0.5014, -0.5007)^T and $\bar{x}_2^*$ =(1.0000, 0.7071, 0.5946, 0.5453, 0.5221, 0.5109, 0.5054, 0.5027, 0.5014, 0.5007)^T with the same function value* $9.6757e-014$*. Table 3.4 records the numerical results.*

Table 3.3.: Numerical results for Example 8 with $n = 5$

k	x_k	$\bar{x}_k$	$f(\bar{x}_k)$	Q	$\bar{x}_k^*$	$f(\bar{x}_k^*)$
1	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0.3333 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	0.6667	$\begin{pmatrix} 3 & -1 & -3 & -5 & 3 \\ -4 & -3 & 5 & 1 & -2 \\ 0 & -5 & -2 & 1 & 5 \\ -5 & -5 & 5 & 4 & 4 \\ 3 & 5 & -2 & -3 & 2 \end{pmatrix}$	$\begin{pmatrix} 1.0000 \\ 0.7071 \\ 0.5946 \\ 0.5452 \\ -0.5221 \end{pmatrix}$	$7.7335e-009$
2	$\begin{pmatrix} 10 \\ 10 \\ 10 \\ 10 \\ 10 \end{pmatrix}$	$\begin{pmatrix} 1.0000 \\ 0.7071 \\ 0.5946 \\ 0.5452 \\ -0.5221 \end{pmatrix}$	$7.7335e-009$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1.0000 \\ 0.7071 \\ 0.5946 \\ 0.5452 \\ -0.5221 \end{pmatrix}$	$7.7335e-009$
3	$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0.3333 \\ 0.0085 \\ 0.0223 \\ 0.1043 \\ 0.2283 \end{pmatrix}$	0.6666	$\begin{pmatrix} -4 & 2 & -3 & -2 & -3 \\ -1 & 1 & 0 & -5 & 2 \\ -3 & -3 & 1 & -1 & -3 \\ 0 & -5 & -4 & -2 & -4 \\ -2 & -5 & -4 & 1 & 5 \end{pmatrix}$	$\begin{pmatrix} 1.0000 \\ 0.7071 \\ 0.5946 \\ 0.5452 \\ 0.5221 \end{pmatrix}$	$5.5342e-008$

Next, we try to solve two moderately large scale quartic polynomial programming problems given in [81] by our algorithms. The computation was implemented on a Linux Desktop of

Table 3.4.: Numerical results for Example 8 with $n = 10$

k	x_k	$\bar{x}_k$	$f(\bar{x}_k)$	Q	$\bar{x}_k^*$	$f(\bar{x}_k^*)$
1	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0.3333 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	0.6667	$I_{10\times 10}$	$\begin{pmatrix} 0.3333 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	0.6667
		$\begin{pmatrix} 1.0000 \\ 0.7071 \\ 0.5946 \\ 0.5453 \\ 0.5221 \\ 0.5109 \\ 0.5054 \\ 0.5027 \\ 0.5014 \\ -0.5007 \end{pmatrix}$	$9.6757e-014$	$I_{10\times 10}$	$\begin{pmatrix} 1.0000 \\ 0.7071 \\ 0.5946 \\ 0.5453 \\ 0.5221 \\ 0.5109 \\ 0.5054 \\ 0.5027 \\ 0.5014 \\ -0.5007 \end{pmatrix}$	$9.6757e-014$
2	$\begin{pmatrix} 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \end{pmatrix}$	$\begin{pmatrix} 0.3333 \\ -0.0000 \\ 0.0000 \\ 0.0001 \\ 0.0060 \\ 0.0550 \\ 0.1658 \\ 0.2879 \\ 0.3794 \\ 0.4356 \end{pmatrix}$	0.6667	$I_{10\times 10}$	$\begin{pmatrix} 0.3333 \\ -0.0000 \\ 0.0000 \\ 0.0001 \\ 0.0060 \\ 0.0550 \\ 0.1658 \\ 0.2879 \\ 0.3794 \\ 0.4356 \end{pmatrix}$	0.6667
		$\begin{pmatrix} 1.0000 \\ 0.7071 \\ 0.5946 \\ 0.5453 \\ 0.5221 \\ 0.5109 \\ 0.5054 \\ 0.5027 \\ 0.5014 \\ 0.5007 \end{pmatrix}$	$9.6757e-014$	$I_{10\times 10}$	$\begin{pmatrix} 1.0000 \\ 0.7071 \\ 0.5946 \\ 0.5453 \\ 0.5221 \\ 0.5109 \\ 0.5054 \\ 0.5027 \\ 0.5014 \\ 0.5007 \end{pmatrix}$	$9.6757e-014$
3	$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 1.0000 \\ 0.7071 \\ 0.5946 \\ 0.5453 \\ 0.5221 \\ 0.5109 \\ 0.5054 \\ 0.5027 \\ 0.5014 \\ -0.5007 \end{pmatrix}$	$9.6757e-014$	$I_{10\times 10}$	$\begin{pmatrix} 1.0000 \\ 0.7071 \\ 0.5946 \\ 0.5453 \\ 0.5221 \\ 0.5109 \\ 0.5054 \\ 0.5027 \\ 0.5014 \\ -0.5007 \end{pmatrix}$	$9.6757e-014$

3.8GB memory and 2.8GHz CPU frequency in [81], while the computation was implemented on a Microsoft Windows XP Desktop of 3.46GB memory and 2.99GHz CPU frequency in our thesis.

Example 9. *Consider the following problem (Example 4.10 [81])*

$$\begin{aligned} \min \quad f(x) := & \sum_{1\le i<j\le n} (x_i x_j + x_i^2 x_j - x_j^3 - x_i^2 x_j^2) \\ s.t. & \\ & x_i \in [-1, 1], i = 1, \cdots, n \end{aligned}$$

For $n = 50$, the global optimal function value given in [81] is -1250. However the corresponding minimizer was not obtained in [81]. A local minimizer with a greater objective value -1232 was given in [81]. While only by By Algorithm *6, with $Q = I$, we have the following results.*

From the starting point $x_0 = \underbrace{(0.5, \cdots, 0.5)}_{50}$, we get a global minimizer by taking around 10 minutes::

$$(\underbrace{1, \cdots, 1}_{9}, \underbrace{-1, \cdots, -1}_{21}, 1, -1, 1, 1, \underbrace{-1, \cdots, -1}_{3}, \underbrace{1, \cdots, 1}_{13})$$

with the optimal value -1250.

From the starting point $x_0 = \underbrace{(-1, \cdots, -1)}_{50}$, we get a global minimizer by taking around 7 minutes:

$$(\underbrace{1, \cdots, 1}_{5}, \underbrace{-1, \cdots, -1}_{24}, 1, -1, \underbrace{1, \cdots, 1}_{19})$$

with the optimal value -1250.

From the starting point $x_0 = \underbrace{(1, \cdots, 1)}_{50}$ and $x_0 = \underbrace{(-0.5, \cdots, -0.5)}_{50}$, we get a global minimizer by taking around 30 minutes:

$$(\underbrace{-1, \cdots, -1}_{25}, \underbrace{1, \cdots, 1}_{25})$$

with the optimal value -1250.

Example 10. *Consider the following problem (Example 5.1 [81])*

$$\begin{aligned} \min \quad f(x) := \;& x_1^4 + \cdots + x_n^4 + \sum_{1 \le i < j < k \le n} x_i x_j x_k \\ s.t. \quad & \\ & x_i \in [-100, 100], i = 1, \cdots, n \end{aligned}$$

For $n = 20$*, the global optimal function value given in [81] is* $-2.2267e{+}007$*. By* Algorithm *6, we have the same results.*

From the starting point $x_0 = \underbrace{(1, \cdots, 1)}_{20}$ *and* $x_0 = \underbrace{(-1, \cdots, -1)}_{20}$*, by* Algorithm *6 with* $Q = I$*, we get a global minimizer within one minute:*

$$\underbrace{(-42.75, \cdots, -42.75)}_{20}$$

with the optimal value $-2.2267e + 007$*.*

From the starting point $x_0 = \underbrace{(0, \cdots, 0)}_{20}$ *by* Algorithm *6 with* $Q =$

$$
\begin{pmatrix}
7 & 3 & -1 & 5 & -3 & -7 & -8 & 7 & 6 & 1 & 3 & -4 & -9 & -10 & -10 & -9 & -7 & 9 & -10 & -5 \\
9 & -10 & -2 & -5 & 7 & 6 & 10 & 3 & -2 & -4 & -3 & 9 & -5 & 8 & 5 & 4 & -2 & 9 & 1 & -4 \\
-8 & 7 & 6 & 0 & 2 & -4 & -10 & -3 & -5 & 5 & 7 & -1 & 6 & 9 & 0 & -10 & 7 & -9 & 8 & 2 \\
9 & 9 & 6 & 4 & 1 & 1 & 6 & 0 & -2 & -7 & 1 & -7 & -10 & 6 & 0 & -9 & 6 & 5 & 4 & -5 \\
3 & 4 & -7 & 8 & 9 & -7 & 7 & -2 & -8 & 4 & -3 & 9 & 9 & -8 & 8 & 0 & -9 & -5 & -7 & 7 \\
-8 & 5 & 0 & 10 & -4 & 2 & 8 & -9 & -8 & -7 & 9 & 10 & 5 & -5 & 2 & -8 & -2 & -2 & -3 & 10 \\
-575 & -1 & 1 & 5 & -5 & -9 & -5 & 9 & -3 & 8 & -1 & 0 & -3 & 2 & 7 & 1 & 1 & -1 & 5 & \\
1 & -2 & 3 & -8 & 5 & 3 & -2 & -8 & 10 & 3 & 1 & -8 & 2 & 4 & 8 & 7 & -2 & 9 & 10 & -3 \\
10 & 3 & 4 & -7 & -3 & 4 & -5 & -7 & 2 & 6 & 3 & -5 & -6 & -8 & 6 & 5 & 3 & -2 & -7 & 2 \\
10 & -7 & 5 & -5 & 1 & 5 & 6 & -5 & -9 & -9 & 2 & -2 & -1 & 5 & 2 & -7 & 3 & 10 & 7 & -8 \\
-7 & 4 & -5 & 7 & -9 & -1 & -1 & -2 & -6 & 9 & -6 & 2 & 10 & -8 & -7 & 3 & -4 & -4 & 3 & 9 \\
10 & -10 & 4 & -5 & -9 & -9 & 9 & -9 & -3 & 6 & -4 & -5 & 1 & 3 & -5 & 0 & -1 & 4 & -3 & 8 \\
10 & -5 & 3 & 7 & 1 & -6 & -7 & 8 & 7 & 0 & -1 & 2 & 0 & 0 & 8 & 10 & -10 & 3 & -6 & 7 \\
0 & -10 & -7 & -5 & 6 & 9 & -5 & 9 & -10 & -1 & -6 & 4 & -6 & 6 & -10 & 3 & 10 & 1 & -2 & -5 \\
6 & -8 & -8 & 9 & 9 & -7 & -7 & 0 & -10 & -1 & 7 & -6 & 0 & 5 & 0 & 6 & -7 & 4 & 0 & 2 \\
-8 & 7 & 0 & -3 & -8 & 7 & -8 & 0 & -7 & -4 & -6 & -8 & 3 & 8 & -7 & -1 & -8 & 3 & -8 & -10 \\
-2 & 4 & 10 & -6 & 1 & 1 & 8 & -3 & 3 & 0 & -6 & -4 & 4 & 8 & 10 & -1 & -3 & -7 & 2 & -2 \\
9 & -4 & -3 & -5 & -1 & 10 & 2 & 8 & 5 & 0 & -7 & -4 & -2 & -3 & 4 & 7 & -6 & -8 & -6 & -4 \\
6 & 9 & 2 & 2 & -10 & -9 & 1 & -3 & 3 & 7 & -6 & -2 & -3 & 4 & 0 & -9 & 0 & 10 & -2 & -7 \\
10 & -10 & -6 & -1 & -3 & -1 & -7 & -8 & -1 & 6 & -1 & 0 & 10 & -6 & -1 & -8 & -3 & -7 & 2 & -7
\end{pmatrix}
$$

we get a global minimizer within 5 minutes:

$$\underbrace{(-42.75, \cdots, -42.75)}_{20}$$

with the optimal value $-2.2267e + 007$*, where* Q *is taken randomly by computer.*

3.5. Conclusion

Quartic polynomial programming problems $(QPOP)$ are considered in this chapter. We proposed a necessary global optimality condition for $(QPOP)$. Then, we designed a strongly or $\varepsilon-$strongly local optimization method for $(QPOP)$ according to the necessary global optimality condition. Finally, a global optimization method has been designed to solve $(QPOP)$ by combining the local optimization method and an auxiliary function. The Numerical examples illustrate the efficiency of the optimization methods proposed in the chapter.

Chapter 4.

Global optimality conditions and optimization methods for general polynomial programming problems (GP)

This chapter is concerned with general polynomial programming problems with box constraints which are denoted by (GP). First, a necessary global optimality condition for problems (GP) is given. Then we design a local optimization method by using the necessary global optimality condition to obtain some strongly or $\varepsilon-$strongly local minimizers which substantially improve some KKT points. Finally, a global optimization method, by combining the new local optimization method and an auxiliary function, is designed. Numerical examples show that our methods are efficient and stable.

4.1. Introduction

Problems (GP) which belong to nonlinear programming problems have a wide range of applications. These include engineering design, investment science, control theory, network distribution, signal processing and location-allocation contexts [5], [6], [11], [17], [46], [50], [88]. Many famous test functions are polynomial functions, such as Rosenbrock, Wood, Powell quartic, Six-hump camelback and Goldstein and Price functions [18]. Moreover, some functions, for example, sin, log and radicals, can be reformulated into polynomial functions, which extends the applications of polynomial programming problems [135]. The problems (GP) are NP-hard [68]. Indeed, even quadratic programming problems are NP-hard [139]. The problems (GP) have attracted a lot of attention, including quadratic, cubic, homogenous or normal quartic as special cases.

Existing methods for solving problems (GP) include algebraic methods [59], [63] , [123] and various convex relaxation methods [35], [48], [67], [79], [80], [107]. Algebraic algorithms tried to find all the critical points and then compared the function values of the polynomial at these points. Although these methods usually provide good approximation, the computation cost is huge [30]. For the idea of convex relaxation methods, please refer to the paper [93]. Among various convex relaxation methods, semidefinite programming (SDP) and sum of squares (SOS) relaxations are very popular. As we surveyed in Chapter 1, we know that solving large scale SDP problems still remains a computational challenge.

Besides global optimization methods, more and more researchers concentrate on global optimality conditions for problems (GP). [126] provided a necessary and sufficient global optimality condition for problems (GP), as it mentioned, the condition is difficult to check since the condition involves solving a sequence of semidefinite programs. Furthermore [127] presented global optimality conditions for polynomial optimization over box or bivalent constraints by using separable polynomial relaxations. However, We notice that it is not easy to decompose a polynomial function to the sum of a separable polynomial function and an SOS-convex polynomial function.

After we built up knowledge from cubic and quartic programming problems in chapter 2 and chapter 3, we will focus on the problem (GP) given below in this chapter.

$$(GP) \qquad \min \quad f(x) = \sum_{\substack{j_1,j_2,\cdots,j_n \geq 0 \\ j_1+j_2+\cdots+j_n \leq n}} c_{j_1,j_2,\cdots,j_n} x_1^{j_1} x_2^{j_2} \cdots x_n^{j_n}$$
$$s.t. \quad x_i \in [u_i, v_i], i = 1, \ldots, n,$$

where n is a nonnegative integer number, $x = (x_1, x_2, \cdots, x_n)^T \in R^n$, $u_i, v_i, c_{j_1,j_2,\cdots,j_n} \in R$ and $u_i < v_i$ for any $i = 1, \ldots, n$. Throughout this chapter, we let $X := \{x = (x_1, \ldots, x_n)^T \mid x_i \in [u_i, v_i], i = 1, \ldots, n\}$. We will first discuss a necessary global optimality condition for the problem (GP). Then a new local optimization method will be designed for the problem (GP) according to the necessary global optimality condition, which may improve some KKT points. Finally, we will design a global optimization method to solve the problem (GP) by combining the new local optimization method and an auxiliary function. Numerical examples illustrate the efficiency of the optimization methods proposed in the chapter.

4.2. Preliminary

Definition 19. *[100] Consider the problem of minimizing $f(x)$ over feasible set X, and let $\bar{x} \in X$. Let $B_\delta(\bar{x}) = \{x \| \|x - \bar{x}\| < \delta\}$ and $N_\delta(\bar{x}) = B_\delta(\bar{x}) \bigcap X$. If $f(\bar{x}) \leq f(x)$ for all $x \in X$, $\bar{x}$ is called a global minimum. If there exists an $\delta-$neighborhood $N_\delta(\bar{x}) \subset X$ around $\bar{x}$ such that $f(\bar{x}) \leq f(x)$ for each $x \in N_\delta(\bar{x})$, $\bar{x}$ is called a local minimum.*

Firstly, we will review KKT necessary conditions for (GP).

Let $\bar{x}$ be a local minimizer of (GP). Then there exist scalars a_i and b_i such that

$$[KKT]\quad \begin{cases} (\nabla f(\bar{x}))_i + a_i - b_i = 0, \\ a_i(\bar{x}_i - v_i) = 0 \quad for\ i = 1, \cdots, n, \\ b_i(-\bar{x}_i + u_i) = 0 \quad for\ i = 1, \cdots, n, \\ a_i \geq 0 \quad for\ i = 1, \cdots, n, \\ b_i \geq 0 \quad for\ i = 1, \cdots, n. \end{cases}$$

In this chapter, we try to give a necessary global optimality condition for the problem (GP) according to the following points. If a point $\bar{x}$ is a global minimizer, then it is not only a KKT point, but also a global minimizer on any line through $\bar{x}$ and within the feasible set X; Some specific lines can be obtained by using linear transformations. On these special lines, the objective function can be simplified into univariate polynomial functions. Then, the necessary and sufficient global optimality conditions for these univariate polynomial problems construct a necessary global optimality condition for the problem (GP).

In the following, let us introduce some relevant properties of the univariate polynomial which will be used later.

Consider the following polynomial with real coefficients:

$$p(x) = \sum_{i=0}^{n} \alpha_i x^i,\ x \in [a, b].$$

We know that the number of distinct real roots of a polynomial in an interval can be obtained by using Sturm's theorem.

Definition 20. *[128] Consider the polynomial function $p(x)$. Let $p_1(x) = p'(x)$ (the derivative of $p(x)$). Let us seek the greatest common divisor p_n of p and p_1 with the help of Euclid's* *algorithm:*

$$p = q_1 p_1 - p_2,$$
$$p_1 = q_2 p_2 - p_3,$$
$$\cdots\cdots\cdots\cdots$$
$$p_{n-2} = q_{n-1} p_{n-1} - p_n,$$
$$p_{n-1} = q_n p_n.$$

The sequence $p, p_1, \cdots, p_{n-1}, p_n$ is called the Sturm sequence of the polynomial p.

Theorem 9. (Sturm Theorem) *[128] Consider the polynomial function $p(x)$. Let $V_p(x)$ be the number of sign changes in the Sturm sequence*

$$p(x), p_1(x), \cdots, p_n(x).$$

The number of the roots of p (without taking multiplicities into account) confined between a and b, where $p(a) \neq 0$, $p(b) \neq 0$ and $a < b$, is equal to $V_p(a) - V_p(b)$.

Remark 10. *(P27 in ([128])) In this theorem, we use the notion of* **number of sign changes** *in the sequence $a_0, a_1, \cdots, a_n$, where $a_0 a_n \neq 0$. The* **number of sign changes** *is determined as follows: all the zero terms of the sequence considered are deleted and, for the remaining non-zero terms, one counts the number of pairs of neighboring terms of different sign.*

In Sturm's theorem, we do not know any information about the multiplicity of every multiple root for a polynomial in an interval. Reference [87] discusses more information about multiple roots and furthermore gives a necessary and sufficient condition for "a polynomial only have even multiplicity roots or only have odd multiplicity roots in a given interval". Next we will introduce some relevant notations and this necessary and sufficient condition.

Denote x_i, $i = 1, 2, 3, \cdots, l$ as all distinct real roots of $p(x)$ in an interval $[a, b]$ and the corresponding multiplicities as m_i, $i = 1, 2, 3, \cdots, l$, respectively. Let

$$K = max\{m_1, m_2, \cdots, m_l\}. \tag{4.1}$$

Denote $p^0(x) = p(x)$ and denote $p^i(x)$ as a greatest common divisor of $p^{i-1}(x)$ and $(p^{i-1}(x))'$, $i = 1, 2, \cdots, K$. For a polynomial $p(x)$, K is fixed but unknown. In our following algorithm, we do not need to know the exact value of K. We know that K satisfies that $p^K(x) \equiv constant$, which is used as the termination criterion in the algorithm.

Lemma 2. *([87]) Suppose that $p(a)p(b) \neq 0$. $p^K(x) \equiv constant$. Then polynomial $p(x)$ has no odd multiplicity roots in an interval $[a, b]$ if and only if*

$$V_{p^{2i}}(a) - V_{p^{2i}}(b) = V_{p^{2i+1}}(a) - V_{p^{2i+1}}(b),\ i = 0, 1, 2, \cdots, [\frac{K-1}{2}].$$

where $V_p(x)$ is defined in Theorem 9.

Proposition 1. *Let $p(x) \not\equiv 0$ be a polynomial with real coefficients. Suppose that $p(a)p(b) \neq 0$. $p^K(x) \equiv constant$. $p(x) \geq 0$ ($p(x) \leq 0$), $\forall x \in [a, b]$ if and only if $p(a) > 0$ ($p(a) < 0$), and $p(x)$ has no odd multiplicity root in (a, b), i.e., the following equations hold:*

$$V_{p^{2i}}(a) - V_{p^{2i}}(b) = V_{p^{2i+1}}(a) - V_{p^{2i+1}}(b),\ i = 0, 1, 2, \cdots, [\frac{K-1}{2}]. \tag{4.2}$$

Proof: We only prove the case of $p(x) \geq 0, \forall x \in [a, b]$. For the case of $p(x) \leq 0, \forall x \in [a, b]$, the proof is similar.

Firstly, we prove the necessary condition. If $p(x) \geq 0$, $\forall x \in [a, b]$ and $p(a) \neq 0$, then we must have that $p(a) > 0$. Suppose that $p(x)$ has an odd multiplicity root x_1 in (a, b) and the multiplicity is m, i.e., there exists a polynomial $q(x)$ such that $p(x) = (x - x_1)^m q(x)$ and $q(x_1) \neq 0$. By the continuity of $q(x)$, there exists a small real number $\delta > 0$, such that

$x_1+\delta \in (a,b)$ and $x_1-\delta \in (a,b)$ and $q(x_1+\delta)q(x_1-\delta) > 0$. However $p(x_1+\delta)p(x_1-\delta) = -\delta^{2m}q(x_1+\delta)q(x_1-\delta) < 0$, which contradicts $p(x) \geq 0$ for any $x \in [a,b]$.

Secondly, we prove the sufficient condition. Suppose that $x_1, \ldots, x_l$ are all the roots of $p(x)$ in (a,b) and the multiplicity corresponding to the roots $x_i, i = 1, \ldots, l$ are $m_1, \ldots, m_l$, respectively. If $p(x)$ has no odd multiplicity root in (a,b), then all the $m_i, i = 1, \ldots, l$ are even. Then there exists a polynomial $q(x)$ such that $p(x) = (x-x_1)^{m_1} \ldots (x-x_l)^{m_l} q(x)$ and $q(x) \neq 0$ for any $x \in (a,b)$. Furthermore, we can prove that if $p(a) > 0$, then $q(x) > 0$ for any $x \in [a,b]$. In fact, obviously, we have that $q(a) > 0$. If there exists an $x \in (a,b]$ such that $q(x) < 0$, then there must exist an $\bar{x} \in (a,x)$ such that $q(\bar{x}) = 0$ which contradicts $q(x) \neq 0$ for any $x \in (a,b)$. For any $x \in [a,b]$, we have that $p(x) = (x-x_1)^{m_1} \ldots (x-x_l)^{m_l} q(x) \geq 0$ since $m_i, i = 1, \ldots, l$ are even and $q(x) > 0$ for any $x \in [a,b]$. □

In Proposition 1, $p(a)p(b) \neq 0$ is required. If $p(a)p(b) = 0$, we can introduce the following function $\bar{p}(x)$:

$$\bar{p}(x) = \begin{cases} p(x), & if\ p(a)p(b) \neq 0 \\ p(x)/[(x-a)^s(b-x)^t], & if\ p(a)p(b) = 0, \end{cases}$$

where s and t are multiplicities of roots a and b, respectively ($s = 0$ or $t = 0$ means a or b is not root). Obviously, $\bar{p}(a)\bar{p}(b) \neq 0$. We can obtain the following Proposition 2.

Proposition 2. *$p(x) \not\equiv 0$ is a polynomial with real coefficients. $p^K(x) \equiv constant$. $p(x) \geq 0$ $(p(x) \leq 0)$ $\forall x \in [a,b]$ if and only if $\bar{p}(a) > 0$ $(\bar{p}(a) < 0)$, and the following equations hold:*

$$V_{\bar{p}^{2i}}(a) - V_{\bar{p}^{2i}}(b) = V_{\bar{p}^{2i+1}}(a) - V_{\bar{p}^{2i+1}}(b),\ i = 0, 1, 2, \cdots, [\frac{K-1}{2}].$$

Proof: Obviously, $x_1, \cdots, x_l$ are roots of $p(x)$ in (a,b) with $m_1, \cdots, m_l$ multiplicity, respectively, if and only if $\bar{p}(x)$ has the same roots $x_1, \cdots, x_l$ and with the same multiplicity $m_i, i = 1, \ldots, l$, respectively. Furthermore, $p(x)\bar{p}(x) \geq 0$ for any $x \in [a,b]$ and

$\bar{p}(a)\bar{p}(b) \neq 0$. Hence, $p(x) \geq 0$ ($p(x) \leq 0$), $\forall x \in [a, b]$ if and only if $\bar{p}(x) \geq 0$ ($\bar{p}(x) \leq 0$) for any $x \in [a, b]$ and $\bar{p}(a)\bar{p}(b) \neq 0$. Moreover, by the definition of K, the K for both functions p and $\bar{p}$ is the same. By Proposition 1, we can have $p(x) \geq 0$ ($p(x) \leq 0$), $\forall x \in [a, b]$ if and only if $\bar{p}(a) > 0$ ($\bar{p}(a) < 0$), and the following equations hold:

$$V_{\bar{p}^{2i}}(a) - V_{\bar{p}^{2i}}(b) = V_{\bar{p}^{2i+1}}(a) - V_{\bar{p}^{2i+1}}(b),\ i = 0, 1, 2, \cdots, [\frac{K-1}{2}].$$

$\square$

The following algorithm can be used to check whether $p(x) \geq 0$ for any $x \in [a, b]$.

Algorithm 8. Step 1. *If $p(a) = 0$, go to Step 2; if $p(b) = 0$, go to Step 3; otherwise, go to Step 4.*

Step 2. $p(x) = \frac{p(x)}{x-a}$, *go to Step 1.*

Step 3. $p(x) = \frac{p(x)}{b-x}$, *go to Step 1.*

Step 4. *If $p(a) < 0$, go to Step 7; otherwise let $p^0 := p$ and go to Step 5.*

Step 5. *Let $p^1 := gcd(p^0, (p^0)')$, $s0 := sturmseq(p^0, x)$ and $s1 := sturmseq(p^1, x)$. If $sturm(s0, x, a, b) = sturm(s1, x, a, b)$, go to Step 6; otherwise, go to Step 7.*

Step 6. *Let $p^0 := gcd(p^1, (p^1)')$. If p^0 is a constant, go to Step 8; otherwise go to Step 5.*

Step 7. *Stop, polynomial p does not satisfy that $p(x) \geq 0$ for any $x \in [a, b]$.*

Step 8. *Stop, polynomial p satisfies that $p(x) \geq 0$ for any $x \in [a, b]$.*

Note, p' is the derivative of p and $gcd(p, q)$ represents greatest common divisor of polynomials p and q. 'sturmseq' and 'sturm' are built-in functions in Maple.

Function '$s := sturmseq(p, x)$' can compute a Sturm sequence s for the polynomial p and function '$sturm(s, x, a, b)$' uses Sturm's theorem to return the number of real roots of polynomial p in the interval $(a, b]$.

4.3. Necessary global optimality condition for (GP)

In this section, we will give a necessary global optimality condition for the problem (GP). Let $\bar{x} \in X$, Q be an invertible matrix, let

$$x := Qy, \;\; g(y) := f(Qy) = f(x), \;\; \bar{y} := Q^{-1}\bar{x},$$

and let $(Q)_i$ represent the ith row of Q, $(Q)_{ij}$ represent the entry of Q in the ith row and the jth column.

Let $Y = \{y = Q^{-1}x | x \in X\}$. For $\bar{y} = (\bar{y}_1, \ldots, \bar{y}_n)^T = Q^{-1}\bar{x}$, let $Y_i := \{y = (\bar{y}_1, \cdots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \cdots, \bar{y}_n)^T \mid y \in Y\}$. Let $\triangle_k = \sum\limits_{\substack{j=1\\ j\neq i}}^{n} (Q)_{kj}\bar{y}_j = \bar{x}_k - (Q)_{ki}\bar{y}_i = \bar{x}_k - (Q)_{ki}(Q^{-1})_i\bar{x}$, $k = 1, \cdots, n$, and let

$$\begin{aligned} l_i &= \max\left\{\min\left\{\frac{u_1 - \triangle_1}{(Q)_{1i}}, \frac{v_1 - \triangle_1}{(Q)_{1i}}\right\}, \cdots, \min\left\{\frac{u_n - \triangle_n}{(Q)_{ni}}, \frac{v_n - \triangle_n}{(Q)_{ni}}\right\}\right\}, \\ r_i &= \min\left\{\max\left\{\frac{u_1 - \triangle_1}{(Q)_{1i}}, \frac{v_1 - \triangle_1}{(Q)_{1i}}\right\}, \cdots, \max\left\{\frac{u_n - \triangle_n}{(Q)_{ni}}, \frac{v_n - \triangle_n}{(Q)_{ni}}\right\}\right\}. \end{aligned}$$

Then we can obtain the following results:

$$\begin{aligned} &(1) \quad l_i \le r_i \\ &(2) \quad [l_i, r_i] = \{y_i \mid (\bar{y}_1, \cdots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \cdots, \bar{y}_n)^T \in Y\}. \end{aligned}$$

Let $G_i(y_i) := f(Qy) - f(Q\bar{y}), y \in Y_i$, which is a univariate polynomial of y_i, for any $i = 1, \cdots, n$. Let

$$\bar{G}_i = \begin{cases} G_i, & if\ G_i(l_i)G_i(r_i) \neq 0 \\ G_i/[(y_i - l_i)^{s(i)}(r_i - y_i)^{t(i)}], & if\ G_i(l_i)G_i(r_i) = 0 \end{cases}$$

where $s(i)$ and $t(i)$ are multiplicities of roots l_i and r_i, respectively. If l_i or r_i is not root of G_i, then $s(i) = 0$ or $t(i) = 0$.

Theorem 10. (Necessary global optimality condition for (GP)) *Let $\bar{x} \in X$ and Q be any given invertible matrix. If $\bar{x}$ is a global minimizer of (GP), then for any $i = 1, \cdots, n$, the following conditions $[NC]_i$ hold:*
$[NC]_i$: $\bar{G}_i(l_i) > 0$, and the following equations hold:

$$V_{\bar{G}_i^{2k}}(l_i) - V_{\bar{G}_i^{2k}}(r_i) = V_{\bar{G}_i^{2k+1}}(l_i) - V_{\bar{G}_i^{2k+1}}(r_i), k = 0, 1, 2, \cdots, [\frac{K_i - 1}{2}]$$

where K_i is defined in (4.1) by taking $p := G_i$.

Proof: Let $\bar{x}$ be a global minimizer of the problem (GP). Then

$$f(x) - f(\bar{x}) \geq 0. \forall x \in X.$$

Let $\bar{y} = Q\bar{x}$. For any $y = (\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T \in Y$, i.e., $y_i \in [l_i, r_i]$, $\forall i = 1, \ldots, n$, let $G_i(y_i) = f(Qy) - f(Q\bar{y})$, $\forall i = 1, \ldots, n$ and let $x = Qy$, we have that $x \in X$ and

$$G_i(y_i) = f(Qy) - f(Q\bar{y}) = f(x) - f(\bar{x}) \geq 0, \forall y_i \in [l_i, r_i]. \tag{4.3}$$

Obviously, each $G_i(y_i)$, $i = 1, \cdots, n$ is a univariate polynomial of y_i.
By Proposition 2, for any $i = 1, \cdots, n$, (4.3) is equivalent to the conditions $[NC]_i$: $\bar{G}_i(l_i) > 0$, and the following equations hold:

$$V_{\bar{G}_i^{2k}}(l_i) - V_{\bar{G}_i^{2k}}(r_i) = V_{\bar{G}_i^{2k+1}}(l_i) - V_{\bar{G}_i^{2k+1}}(r_i), k = 0, 1, 2, \cdots, [\frac{K_i - 1}{2}]$$

□

Remark 11. *In Theorem 10, we do not need to consider the trivial case $G_i(y_i) \equiv 0$, for some $i = 1, \cdots, n$.*

Remark 12. *Actually, the condition $[NC]_i$ given in Theorem 10 is the necessary and sufficient condition for $\bar{y}_i$ to be a global minimizer of the following problem:*

$$\begin{aligned} \min \quad & f(Qy) \\ s.t. \quad & y \in N_i, \end{aligned} \tag{4.4}$$

where

$$N_i := \{\bar{y} + (z_i - \bar{y}_i)e_i \mid z_i \in [l_i, r_i]\}, \tag{4.5}$$

In particular, if $Q = I$, where I is the identity matrix, then the problem is:

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & x \in \{\bar{x} + (z_i - \bar{x}_i)e_i \mid z_i \in [u_i, v_i]\}, \end{aligned} \tag{4.6}$$

where e_i is the ith unit vector (the n dimensional vector with the ith component equals to one and the other components equal to zero).

Remark 13. (1) *If the problem (GP) reduces to a quartic polynomial programming problem $(QPOP)$, then for any $i = 1, \ldots, n$, $[NC]_i$ is equivalent to the following condition:*

$$\widetilde{\bar{x}}_i d_i \leq \min\{0, \alpha_i\}, \tag{4.7}$$

which is given by Theorem 7 in chapter 3, since both conditions $[NC]_i$ and (4.7), for $i = 1, \cdots, n$, are equivalent to $f(x) - f(\bar{x}) \geq 0$, $\forall x = Q(\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T$, where $y_i \in [l_i, r_i]$. For the notations therein, see Theorem 7 in chapter 3.

(2) *If the problem* (GP) *reduces to a cubic polynomial programming problem, then for any* $i = 1, \ldots, n$, $[NC]_i$ *is equivalent to*

$$\widetilde{\bar{x}}_i d_i \leq \min\{0, \alpha_i\}, \tag{4.8}$$

which is given by Remark 7 (1) in chapter 3. For the notations therein, see Remark 7 (1) in chapter 3. The condition (4.8) extends the condition of Corollary 3 in chapter 2 which is just the special case of (4.8) when $Q = I$.

(3) *If the problem* (GP) *reduces to a quadratic polynomial optimization problem, then for any* $i = 1, \ldots, n$, $[NC]_i$ *is equivalent to*

$$\widetilde{\bar{x}}_i d_i \leq \min\{0, \alpha_i\}, \tag{4.9}$$

which is given by Remark 7 (2) in chapter 3. For the notations therein, see Remark 7 (2) in chapter 3. The condition (4.9) extends the condition for continuous variables of Proposition 2.1 in [45] which is just the special case of (4.9) when $Q = I$.

(4) *The necessary global optimality condition for the problem* (GP) *includes KKT necessary conditions. In fact, when* $Q = I$, *we know that* $[NC]_i$ *is equivalent to (4.3). From (4.3), we have*

$$\begin{aligned} G_i(x_i) &= f(x) - f(\bar{x}) \\ &= \frac{1}{n!}\frac{\partial^n f(\bar{x})}{\partial x_i^n}(x_i - \bar{x}_i)^n + \frac{1}{(n-1)!}\frac{\partial^{n-1} f(\bar{x})}{\partial x_i^{n-1}}(x_i - \bar{x}_i)^{n-1} \\ &\quad + \cdots + \frac{1}{2}\frac{\partial^2 f(\bar{x})}{\partial x_i^2}(x_i - \bar{x}_i)^2 + (\nabla f(\bar{x}))_i(x_i - \bar{x}_i) \\ &\geq 0 \end{aligned}$$

where $x_i \in (u_i, v_i)$.

when $\bar{x}_i = u_i$*,*

$$\begin{aligned} -(\nabla f(\bar{x}))_i &\leq \min_{x_i \in [u_i, v_i]} \{ \frac{1}{n!} \frac{\partial^n f(\bar{x})}{\partial x_i^n} (x_i - \bar{x}_i)^{n-1} + \cdots + \frac{1}{2} \frac{\partial^2 f(\bar{x})}{\partial x_i^2} (x_i - \bar{x}_i) \} \\ &\leq 0 \end{aligned}$$

when $\bar{x}_i = v_i$*,*

$$\begin{aligned} (\nabla f(\bar{x}))_i &\leq \min_{x_i \in [u_i, v_i]} -\{ \frac{1}{n!} \frac{\partial^n f(\bar{x})}{\partial x_i^n} (x_i - \bar{x}_i)^{n-1} + \cdots + \frac{1}{2} \frac{\partial^2 f(\bar{x})}{\partial x_i^2} (x_i - \bar{x}_i) \} \\ &\leq 0 \end{aligned}$$

when $\bar{x}_i \in (u_i, v_i)$*,*

$$(\nabla f(\bar{x}))_i = 0.$$

The above condition is just the KKT condition $[KKT]$*.*

In the following, we will discuss a necessary and sufficient condition for a special polynomial programming problem.

Definition 21. *[127] A function* $f : R^n \to R$ *is a separable polynomial if* $f(x) = \sum_{i=1}^{n} f_i(x_i)$*, where* $x = (x_1, \cdots, x_n)$ *and each* f_i *is a polynomial on* R*. The set of all the separable polynomial functions with degree at most* d *on* R^n *is denoted by*

$$S_d = \{ f \in R[x] : f(x) = \sum_{i=1}^{n} \sum_{j=0}^{d} f_{ij} x_i^j, x = (x_1, \cdots, x_n) \}. \tag{4.10}$$

For the problem (GP), if $f \in S_d$, let $\bar{x} = (\bar{x}_1, \cdots, \bar{x}_n) \in X$, let $G_i(x_i) := \sum_{j=0}^{d} f_{ij}(x_i^j - \bar{x}_i^j)$

for $i = 1, \cdots, n$, and let

$$\bar{G}_i = \begin{cases} G_i, & if\ G_i(u_i)G_i(v_i) \neq 0 \\ G_i/[(x_i - u_i)^{s(i)}(v_i - x_i)^{t(i)}], & if\ G_i(u_i)G_i(v_i) = 0 \end{cases}$$

where $s(i)$ and $t(i)$ are multiplicities of roots u_i and v_i, respectively ($s(i) = 0$ or $t(i) = 0$ means u_i or v_i is not root). Then, we have the following Corollary.

Corollary 4. *(Global optimality characterization) Let $\bar{x} \in X$. $\bar{x}$ is a global minimizer of the problem (GP) if and only if the following conditions hold: for any $i = 1, \cdots, n$, $\bar{G}_i(u_i) > 0$, and the following equations hold:*

$$V_{\bar{G}_i^{2k}}(u_i) - V_{\bar{G}_i^{2k}}(v_i) = V_{\bar{G}_i^{2k+1}}(u_i) - V_{\bar{G}_i^{2k+1}}(v_i), k = 0, 1, 2, \cdots, [\frac{K_i - 1}{2}]$$

where K_i is defined in (4.1) by taking $p := G_i$.

Proof:

$$\begin{aligned} f(x) - f(\bar{x}) &= \sum_{i=1}^{n} \sum_{j=0}^{d} f_{ij}(x_i^j - \bar{x}_i^j) \geq 0,\ \forall x \in X \\ \Leftrightarrow \quad G_i(x_i) &:= \sum_{j=0}^{d} f_{ij}(x_i^j - \bar{x}_i^j) \geq 0,\ \forall i = 1, \cdots, n \end{aligned} \tag{4.11}$$

By Proposition 2, for any $i = 1, \cdots, n$, (4.11) is equivalent to the conditions : $\bar{G}_i(u_i) > 0$, and the following equations hold:

$$V_{\bar{G}_i^{2k}}(u_i) - V_{\bar{G}_i^{2k}}(v_i) = V_{\bar{G}_i^{2k+1}}(u_i) - V_{\bar{G}_i^{2k+1}}(v_i), k = 0, 1, 2, \cdots, [\frac{K_i - 1}{2}]$$

□

Remark 14. *When $u_i = -1, v_i = 1, \forall i = 1, \ldots, n$, the necessary and sufficient global optimality condition given in Corollary 4 is equivalent to the condition given in Theorem*

2.1 in paper [127] with box constraint, which are just different expressions, since both are a necessary and sufficient condition to a global minimizer of separable polynomial problem with box constraint. This can also be seen from Remark 13 (3) and Corollary 2.1 in paper [127].

4.4. Optimization methods for (GP)

4.4.1. Strongly or $\varepsilon-$strongly local optimization method for (GP)

In this section, we will introduce a strongly or $\varepsilon-$strongly local optimization method for the problem (GP) according to the necessary global optimality conditions $[NC]_i, i = 1, \ldots, n$.

Definition 22. *Let $\bar{x} \in X$ and Q be an invertible matrix. $\bar{x}$ is said to be a strongly local minimizer of the problem (GP) with respect to Q iff $\bar{x}$ satisfies the necessary global optimality conditions $[NC]_i$, for any $i = 1, \cdots, n$.*

Definition 23. *Let $\bar{x} \in X$ and Q be an invertible matrix. $\bar{x}$ is said to be an $\varepsilon-$strongly local minimizer of the problem (GP) with respect to Q iff for any $i = 1, \cdots, n$, either $\bar{x}$ satisfies the condition $[NC]_i$ or there exists a point $x_i^* \in X$, such that x_i^* satisfies the condition $[NC]_i$ and $|f(\bar{x}) - f(x_i^*)| \leq \varepsilon$.*

Remark 15. *From Theorem 10, we know that, for any given invertible matrix Q, $[NC]_i$ is satisfied for any $i = 1, \ldots, n$. However, in our algorithm, we only randomly select N invertible matrices $Q_1, \cdots, Q_N$, and we always choose $Q_1 = I$, the identity matrix.*

Let $\bar{x} \in X$ and Q be an invertible matrix. Let $\bar{y} = Q^{-1}\bar{x} = (\bar{y}_1, \ldots, \bar{y}_i, \ldots, \bar{y}_n)^T$, $y = (\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T$ and let $G_i(y_i) := f(Qy) - f(\bar{x})$, $i = 1, \cdots, n$.

Algorithm 9. *Strongly or $\varepsilon-$strongly local optimization method for (GP):(SLOM).*

Step 0. *Take an initial point $x_0 \in X$. Let $Q_1 = I$, Q_2, $\cdots$, Q_d, $\cdots$, Q_N be any invertible matrices given randomly, where I is the identity matrix. Let ε be a small positive number.*

Let $d := 1$, $Q := Q_d$ *and* $i := 1$. *Let* $x^* = (x_1^*, \cdots, x_n^*)^T$ *be a local minimizer or* KKT *point of* $f(x)$ *on* $\prod_{i=1}^{n}[u_i, v_i]$ *starting from* x_0. *Let* $\bar{x} := x^*$, *and go to Step 1.*

Step 1. *Let* $p := G_i(y_i)$, $a := l_i$ *and* $b := r_i$. *Check whether the condition* $[NC]_i$ *holds:* $p(l_i) > 0$ *and the following equations hold:*

$$V_{p^{2k}}(a) - V_{p^{2k}}(b) = V_{p^{2k+1}}(a) - V_{p^{2k+1}}(b),$$
$$k = 0, 1, 2, \cdots, [\frac{K_i - 1}{2}]$$

by using the Algorithm 8. If this condition holds, go to Step 3; otherwise, go to Step 2.

Step 2. *Let* $\bar{y} = (\bar{y}_1, \ldots, \bar{y}_i, \ldots, \bar{y}_n)^T = Q^{-1}\bar{x}$ *and* $y = (\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T$. *Let* $\bar{y}_i^* := argmin\{f(Qy)|y \in N_i\}$, *where* N_i *is defined by (4.5). Let* $\bar{y}^* = (\bar{y}_1, \cdots, \bar{y}_{i-1}, \bar{y}_i^*, \bar{y}_{i+1} \cdots, \bar{y}_n)$ *and* $\bar{x}^* := Q\bar{y}^*$. *Let* $x^* = (x_1^*, \cdots, x_n^*)$ *be a local minimizer or* KKT *point of* $f(x)$ *on* $\prod_{i=1}^{n}[u_i, v_i]$ *starting from* $\bar{x}^*$. *If* $f(x^*) < f(\bar{x}) - \varepsilon$, *let* $\bar{x} := x^*$, $i := 1$, $d := 1$ *and* $Q := Q_d$ *go to Step 1; otherwise, go to Step 3.*

Step 3. *If* $i := n$, *go to Step 4; otherwise, let* $i := i + 1$ *and go to Step 1.*

Step 4. *Let* $d = d + 1$. *If* $d > N$, *go to Step 5; otherwise, let* $Q := Q_d$ *and* $i := 1$, *go to Step 1.*

Step 5. *Stop.* $\bar{x}$ *is a strongly or* $\varepsilon-$*strongly local minimizer with respect to* Q_d, $d = 1, \cdots, N$.

Theorem 11. *For a given initial point* $x_0 \in X$, *we can obtain a strongly or* $\varepsilon-$*strongly local minimizer* $\bar{x}$ *of the problem* (GP) *in finite iteration times by the given strongly local optimization method* $(SLOM)$.

Proof: The proof is similar to Theorem 8 in Chapter 3. □

Remark 16. *In step 2 in Algorithm 9, we need to find a global minimizer of a univariate polynomial in an interval. To achieve this, we can apply any univariate algorithm, such as* *the methods mentioned in references [38] and [40]. More particularly, we can apply some algorithms for univariate polynomial, such as the methods mentioned in references [16] and [129]. Besides these, we can find the minimizer of univariate polynomial by approximating the roots of derivative. We can apply references [26] and [116] to find the roots of derivative. In our implementation, we use commands 'diff' and 'roots' in Matlab to calculate all stationary points (roots of derivative) and then compare the function values of these stationary points. The point with the smallest function value is the global minimum. Actually, here we do not need to find the exact global minimizer of a univariate polynomial in an interval, we just need to use some approximate method to find an approximate global minimizer* $\bar{y}^*$ *such that* $f(Q\bar{y}^*) < f(Q\bar{y})$ *or the local minimizer of* $f(x)$ *on* X *starting from* $Q\bar{y}^*$ *is better than* $\bar{x}$.

Remark 17. *In step 0 and step 2, we can apply any local optimization algorithm to get a local minimizer or a KKT point, such as the method of Zoutendijk (Case of linear constraints) starting from* $\bar{x}$. *In our implementation, the optimization subroutine fmincon within the optimization Toolbox in Matlab is used as the local search scheme to obtain local minimizers.*

4.4.2. Global optimization method for (GP)

In this subsection, we will design a global optimization method for the problem (GP) by combining the strongly local optimization method and an auxiliary function. In this chapter, we still use the auxiliary function which was presented by (1.2) in Chapter 1. For the properties of this auxiliary function, see Chapter 1.

Algorithm 10. *Global optimization method for* (GP):(GOM).

Step 0. *Set* $M := 10^{10}$, $\mu := 10^{-10}$ *and* $k_0 := 2n$. *Set* $A_{n\times n} := I_{n\times n}$ *and* $B_{n\times 2n} := [A, -A]$. *Let* $r_0 := 1$, $c_0 := 1$, $q_0 := 10^5$ *and* $\delta_0 := \frac{1}{2}$. *Let* $k := 1$, $i := 1$ *and* $r := r_0$. *Let* x_1^0 *be an initial point and* $x_0^* := x_1^0$, *then go to Step 1.*

Step 1. *Use the strongly or* $\varepsilon-$*strongly local optimization method* $(SLOM)$ *to solve the problem* (GP) *starting from* x_k^0. *Let* x_k^* *be the obtained strongly or* $\varepsilon-$*strongly local minimizer of the problem* (GP). *If* $f(x_k^*) \geq f(x_0^*)$, *then go to step 6; otherwise let* $q := q_0$, $c := c_0$, $r := r_0$, $\delta := \delta_0$, $i := 1$ *and* $x_0^* := x_k^*$, $k := k+1$, *then go to Step 2.*

Step 2. *Let* B_i *indicate the* i*th column of* B *and* $\bar{x}_k^* := x_0^* + \delta B_i$. *If* $\bar{x}_k^* \notin X$, *go to Step 3. Otherwise, if* $f(\bar{x}_k^*) < f(x_0^*)$, *then set* $x_{k+1}^0 := \bar{x}_k^*$ *and* $x_0^* := \bar{x}_k^*$, $k := k+1$ *and go to Step 1; else go to Step 4.*

Step 3. *If* $\delta < \mu$, *go to Step 8; otherwise, let* $\delta = \frac{\delta}{2}$ *and go to Step 2.*

Step 4. *If* $f(x_0^*) \leq f(\bar{x}_k^*) \leq f(x_0^*) + 1$, *then go to Step 5; otherwise let* $\delta = \frac{\delta}{2}$ *go to Step 2.*

Step 5. *Let*

$$F_{q,r,c,x_0^*}(x) = q\left(exp(-\frac{\|x - x_0^*\|^2}{q})g_{r,c}\Big(f(x) - f(x_0^*)\Big) + h_{r,c}\Big(f(x) - f(x_0^*)\Big)\right).$$

Solve the problem:

$$\begin{aligned} \min \quad & F_{q,r,c,x_0^*}(x) \\ s.t. \quad & x \in X. \end{aligned} \tag{4.12}$$

by a local search method starting from the initial point $\bar{x}_k^*$. *Let* $\bar{x}_{q,r,c,x_k^*}$ *be the local minimizer obtained. Then set* $x_{k+1}^0 := \bar{x}_{q,r,c,x_k^*}$, $k := k+1$ *and go to Step 1.*

Step 6. *If* $q < M$, *then increase* q *(in the following examples, let* $q := 10q$*), then go to Step 5; otherwise go to Step 7.*

Step 7. *If* $c < M$, *then increase* c *(in the following examples, let* $c := 10c$*), and let* $q := q_0$, *then go to Step 5; otherwise go to Step 8.*

Step 8. *If* $i < k_0$, *then let* $i := i+1$, $q := q_0$, $c := c_0$, $\delta = \delta_0$, *go to Step 2; otherwise go to Step 9.*

Step 9. *If $r > \mu$, then decrease r (in the following examples, let $r := \frac{r}{10}$). Randomly select an orthogonal matrix $A_{n\times n}$ and set $B_{n\times 2n} := [A, -A]$. Let $i := 1$, $q := q_0$, $c := c_0$, $\delta = \delta_0$ and go to Step 2; otherwise, stop and x_0^* is the obtained global minimizer or approximate global minimizer of the problem (GP).*

4.5. Numerical examples

In this section, we apply our two Algorithms: strongly local optimization method(SLOM) and global optimization method(GOM) to twenty one test problems. These test problems include Problems 4.1-4.19 from [97], Problem 4.20 from Example 4.1 and Problem 4.21 from Example 5.2 in the paper [81]. For the detailed information of these problems, see the appendix in the end. Table 4.1 shows summary information of the twenty one test problems that are based on a set of polynomial functions.

Table 4.1.: Test problems for (GP)

Problem number	Name and parameter values	Global minimizer x^*	Optimal value $f(x^*)$
4.1	Beale	$(3,\ 0.5)$	0
4.2	Booth	$(1,\ 3)$	0
4.3	Matyas	$(0,\ 0)$	0
4.4	Goldstein and Price	$(0,\ -1)$	3
4.5	Six-hump Camelback	$(1.7036,\ -0.7961)$	-1.0316
		$(-1.7036,\ 0.7961)$	
4.6	Perm(3,0.5)	$(1,\ 2,\ 3)$	0
4.7	Perm0(3,10)	$(1,\ 1/2,\ 1/3)$	0

continue goes here...

Problem number	Name and parameter values	Global minimizer x^*	Optimal value $f(x^*)$
4.8	Perm(4,0.5)	$(1,\ 2,\ 3,\ 4)$	0
4.9	Perm0(4,10)	$(1,\ 1/2,\ 1/3,\ 1/4)$	0
4.10	Colville	$(1,\ 1,\ 1,\ 1)$	0
4.11	Powersum(8,18,44,114)	$(1,\ 2,\ 2,\ 3)$	0
4.12	Dixon and Price	$x_i = 2^{-\frac{z-1}{z}}, z = 2^{i-1}$	0
4.13	Dixon and Price	$x_i = 2^{-\frac{z-1}{z}}, z = 2^{i-1}$	0
4.14	Trid	$x_i = i(11-i)$	-210
4.15	Rosenbrock	$(1, \cdots, 1)$	0
4.16	Sum Squares	$(0, \cdots, 0)$	0
4.17	Zakharov	$(0, \cdots, 0)$	0
4.18	Powell	$(3,\ -1,\ 0,\ 1,\ 3, \cdots,$ $3,\ -1,\ 0,\ 1)$	0
4.19	Sphere	$(0, \cdots, 0)$	0
4.20	Example 4.1 in [81]	$x_1^{*\dagger 1}$ $x_2^{*\dagger 2}$	7.5586
4.21	Example 5.2 in [81]		$0.0648^{\dagger 3}$

$$
\begin{aligned}
{}^{\dagger 1}x_1^* &= (0.0039, 0.6285, 0.5370, 0.0259, -0.4324, -0.4266, 0.1540, -0.5108,\\
&\quad 0.2172, -0.4029, 0.4400, -0.4307, 0.0230, 0.5378, 0.6285, 0.0039)\\
{}^{\dagger 2}x_2^* &= (0.0039, 0.6285, 0.5378, 0.0230, -0.4307, 0.4400, -0.4029, 0.2172,\\
&\quad -0.5108, 0.1540, -0.4266, -0.4324, 0.0259, 0.5370, 0.6285, 0.0039)
\end{aligned}
$$

†3 is an approximate global optimal value provided in the paper [81] and no corresponding minimizer is mentioned.

For our experiments, we use the optimality gap mentioned in [97] is:

$$GAP = |f(x) - f(x^*)|$$

where x is a heuristic solution obtained by our method and x^* is the optimal solution. We then say that a heuristic solution x is optimal if:

$$GAP \leq \begin{cases} \varepsilon & f(x^*) = 0 \\ \varepsilon \times |f(x^*)| & f(x^*) \neq 0 \end{cases}$$

In our experimentation we set $\varepsilon = 0.001$ as the same of that in [97].

In the table below, some common statistics are included. We randomly select 30 initial points for every problem. The *suc.rate*(success rate) means the success times out of 30. The *best* is the minimum of the results, the *worst* indicates the maximum of the results, and then it follows the *mean*, *median* and *st.dev.*(standard deviation). In some way, these statistics are able to evaluate the search ability and solution accuracy, reliability and convergence as well as stability.

Table 4.2.: Results of algorithms SLOM and GOM for (GP)

Problem number	statistic	SLOM	GOM
4.1	suc.rate	$29/30$	$30/30$
	best	$4.4260e-014$	$4.4260e-014$
	worst	0.7621	$6.3560e-013$
	mean	0.0254	$2.3292e-013$

continue goes here...

Problem number	statistic	SLOM	GOM
	median	$2.3569e-013$	$2.3541e-013$
	st.dev	0.1391	$1.0040e-013$
4.2	suc.rate	$30/30$	$30/30$
	best	$4.0732e-015$	$4.0732e-015$
	worst	$3.0047e-014$	$3.0047e-014$
	mean	$1.0685e-014$	$1.0685e-014$
	median	$1.0302e-014$	$1.0302e-014$
	st.dev	$3.8452e-015$	$3.8452e-015$
4.3	suc.rate	$30/30$	$30/30$
	best	$1.2579e-016$	$1.2579e-016$
	worst	$1.2720e-012$	$1.2720e-012$
	mean	$1.5877e-013$	$1.5877e-013$
	median	$2.1690e-014$	$2.1690e-014$
	st.dev	$2.8049e-013$	$2.8049e-013$
4.4	suc.rate	$30/30$	$30/30$
	best	3.0000	3.0000
	worst	3.0000	3.0000
	mean	3.0000	3.0000
	median	3.0000	3.0000
	st.dev	0	0
4.5	suc.rate	$30/30$	$30/30$
	best	-1.0316	-1.0316
	worst	-1.0316	-1.0316
	mean	-1.0316	-1.0316

continue goes here…

Problem number	statistic	SLOM	GOM
	median	-1.0316	-1.0316
	st.dev	0	0
4.6	suc.rate	$19/30$	$30/30$
	best	$6.4996e-007$	$6.4996e-007$
	worst	0.0034	$8.1093e-007$
	mean	0.0012	$7.0362e-007$
	median	$8.1093e-007$	$6.4996e-007$
	st.dev	0.0017	$7.7179e-008$
4.7	suc.rate	$30/30$	$30/30$
	best	$1.4132e-013$	$1.4132e-013$
	worst	$4.9007e-004$	$1.9212e-012$
	mean	$1.9603e-004$	$6.2854e-013$
	median	$1.0355e-012$	$3.8054e-013$
	st.dev	$2.4419e-004$	$4.4463e-013$
4.8	suc.rate	$13/30$	$29/30$
	best	$1.1067e-006$	$6.6125e-007$
	worst	0.4723	0.0048
	mean	0.0314	$3.7739e-004$
	median	0.0012	$1.1452e-005$
	st.dev	0.0910	$8.7023e-004$
4.9	suc.rate	$24/30$	$30/30$
	best	$3.9667e-013$	$3.9667e-013$
	worst	0.0109	$8.7909e-005$
	mean	0.0025	$1.4309e-005$

continue goes here…

Problem number	statistic	SLOM	GOM
	median	$9.0656e-004$	$4.2752e-012$
	st.dev	0.0043	$2.9628e-005$
4.10	suc.rate	$30/30$	$30/30$
	best	$3.5684e-013$	$3.5684e-013$
	worst	$6.1499e-013$	$6.1499e-013$
	mean	$5.4901e-013$	$5.4901e-013$
	median	$5.7129e-013$	$5.7129e-013$
	st.dev	$5.4297e-014$	$5.4297e-014$
4.11	suc.rate	$30/30$	$30/30$
	best	$1.7637e-008$	$1.7637e-008$
	worst	$4.2910e-004$	$8.2802e-007$
	mean	$4.3089e-005$	$1.9577e-007$
	median	$1.4991e-007$	$1.4924e-007$
	st.dev	$1.3087e-004$	$1.5486e-007$
4.12	suc.rate	$25/30$	$30/30$
	best	$1.8580e-014$	$1.8580e-014$
	worst	0.6667	$1.1907e-013$
	mean	0.1111	$3.4096e-014$
	median	$2.8479e-014$	$2.8479e-014$
	st.dev	0.2527	$1.9009e-014$
4.13	suc.rate	$0/30$	$30/30$
	best	0.6667	$6.6770e-014$
	worst	0.6667	$1.6046e-013$
	mean	0.6667	$1.0319e-013$

continue goes here…

Problem number	statistic	SLOM	GOM
	median	0.6667	$9.6125e-014$
	st.dev	$3.1892e-014$	$2.4841e-014$
4.14	suc.rate	$30/30$	$30/30$
	best	-210.0000	-210.0000
	worst	-210.0000	-210.0000
	mean	-210.0000	-210.0000
	median	-210.0000	-210.0000
	st.dev	$2.5864e-011$	$2.5864e-011$
4.15	suc.rate	$30/30$	$30/30$
	best	$3.7440e-013$	$3.7440e-013$
	worst	$7.2587e-006$	$7.2587e-006$
	mean	$2.4197e-007$	$2.4197e-007$
	median	$1.3327e-011$	$1.3327e-011$
	st.dev	$1.3252e-006$	$1.3252e-006$
4.16	suc.rate	$30/30$	$30/30$
	best	$5.2298e-015$	$5.2298e-015$
	worst	$3.2713e-013$	$3.2713e-013$
	mean	$4.1423e-014$	$4.1423e-014$
	median	$1.3512e-014$	$1.3512e-014$
	st.dev	$7.3224e-014$	$7.3224e-014$
4.17	suc.rate	$30/30$	$30/30$
	best	$4.7531e-016$	$4.7531e-016$
	worst	$7.9295e-015$	$7.9295e-015$
	mean	$2.5000e-015$	$2.5000e-015$

continue goes here…

Problem number	statistic	SLOM	GOM
	median	$1.6761e-015$	$1.6761e-015$
	st.dev	$2.1075e-015$	$2.1075e-015$
4.18	suc.rate	$30/30$	$30/30$
	best	$5.9340e-008$	$5.9340e-008$
	worst	$3.0156e-005$	$3.0156e-005$
	mean	$5.4882e-006$	$5.4882e-006$
	median	$2.4128e-006$	$2.4128e-006$
	st.dev	$8.4603e-006$	$8.4603e-006$
4.19	suc.rate	$30/30$	$30/30$
	best	$4.5263e-016$	$4.5263e-016$
	worst	$1.5938e-012$	$1.5938e-012$
	mean	$1.5661e-013$	$1.5661e-013$
	median	$2.0627e-014$	$2.0627e-014$
	st.dev	$3.0978e-013$	$3.0978e-013$
4.20	suc.rate	$0/30$	$30/30$
	best	7.5711	7.5586
	worst	7.7002	7.5586
	mean	7.6200	7.5586
	median	7.6270	7.5586
	st.dev	0.0351	$7.0247e-014$
4.21	suc.rate	$13/30$	$30/30$
	best	$5.9917e-005$	$1.7511e-006$
	worst	0.0044	$9.8407e-006$
	mean	0.0018	$6.6478e-006$

continue goes here…

Problem number	statistic	SLOM	GOM
	median	0.0015	$6.9778e-006$
	st.dev	0.0016	$3.2757e-006$

It is shown from table 4.2 that GOM can successfully find the global minimizer starting from almost all of the randomly selected 30 initial points for each test problem. Only for Problem 4.8, the success rate for Algorithm GOM is 29 out of 30. For Problem 4.21, we find a better solution than that mentioned in [81]. Overall, Algorithm GOM is very efficient and stable. As a local optimization method, SLOM can also be considered as a competitive algorithm with producing impressive results.

Since SDP and SOS relaxation methods are very popular for polynomial optimization, we try to compare our GOM method with the solver GloptiPoly 3 which is a Matlab/SeDuMi add-on for SDP-relaxations of minimization problems over multivariable polynomial functions subject to polynomial or integer constraints [31, 32].

Table 4.3.: Comparisons between GOM and Gloptipoly 3 for (GP)

Problem number	statistic	GOM	GloptiPoly 3
4.1	suc.rate	$30/30$	$30/30$
	best	$4.4260e-014$	$2.4615e-007$
	worst	$6.3560e-013$	$2.5555e-007$
4.2	suc.rate	$30/30$	$30/30$
	best	$4.0732e-015$	$5.6296e-008$
	worst	$3.0047e-014$	$5.6296e-008$
4.3	suc.rate	$30/30$	$30/30$

continue goes here…

Problem number	statistic	GOM	GloptiPoly 3
	best	$1.2579e-016$	$2.8912e-031$
	worst	$1.2720e-012$	$2.8912e-031$
4.4	suc.rate	$30/30$	$30/30$
	best	3.0000	3.0000
	worst	3.0000	3.0000
4.5	suc.rate	$30/30$	$30/30$
	best	-1.0316	-1.0316
	worst	-1.0316	-1.0316
4.6	suc.rate	$30/30$	$30/30$
			order$=3$
	best	$6.4996e-007$	$1.6287e-005$
	worst	$8.1093e-007$	$1.6287e-005$
4.7	suc.rate	$30/30$	$21/30$
			order$=3$
	best	$1.4132e-013$	$1.7784e-006$
	worst	$1.9212e-012$	-
4.8	suc.rate	$29/30$	$0/30$
	best	$6.6125e-007$	-
	worst	0.0048	-
4.9	suc.rate	$30/30$	$0/30$
	best	$3.9667e-013$	-
	worst	$8.7909e-005$	-
4.10	suc.rate	$30/30$	$30/30$
	best	$3.5684e-013$	$6.3203e-009$

continue goes here…

Problem number	statistic	GOM	GloptiPoly 3
	worst	$6.1499e-013$	$6.3203e-009$
4.11	suc.rate	$30/30$	$0/30$
	best	$1.7637e-008$	-
	worst	$8.2802e-007$	-
4.12	suc.rate	$30/30$	$30/30$
			order$=3$
	best	$1.8580e-014$	$2.1817e-009$
	worst	$1.1907e-013$	$2.1858e-009$
4.13	suc.rate	$30/30$	$0/30$
	best	$6.6770e-014$	-
	worst	$1.6046e-013$	-
4.14	suc.rate	$30/30$	$30/30$
	best	-210.0000	-210.0000
	worst	-210.0000	-210.0000
4.15	suc.rate	$30/30$	$0/30$
	best	$3.7440e-013$	-
	worst	$7.2587e-006$	-
4.16	suc.rate	$30/30$	$30/30$
	best	$5.2298e-015$	$1.8780e-030$
	worst	$3.2713e-013$	$1.8780e-030$
4.17	suc.rate	$30/30$	$0/30$
	best	$4.7531e-016$	-
	worst	$7.9295e-015$	-
4.18	suc.rate	$30/30$	$0/30$

continue goes here…

Problem number	statistic	GOM	GloptiPoly 3
	best	$5.9340e-008$	-
	worst	$3.0156e-005$	-
4.19	suc.rate	$30/30$	$30/30$
	best	$4.5263e-016$	$4.4866e-032$
	worst	$1.5938e-012$	$4.4866e-032$
4.20	suc.rate	$30/30$	$0/30$
	best	7.5586	-
	worst	7.5586	-
4.21	suc.rate	$30/30$	$0/30$
	best	$1.7511e-006$	-
	worst	$9.8407e-006$	-

When we use GloptiPoly 3 to solve non-convex polynomial programming problems, it may not return the global optimum but a lower bound. The default order in GloptiPoly 3 is such that twice the order is greater than or equal to the maximal degree occurring in the polynomial expressions of the original optimization problem. More importantly, the series of optima of SDP-relaxations of increasing orders converges monotonically to the global optimum [31]. However, the computational time increases quickly with the increasing relaxation order and the computer may return 'out of memory' when the order is big enough.

In the table 4.3, we use the solver GloptiPoly 3 to solve Problem 4.1-4.21. We run Gloptipoly 3 30 times for each problem with fixed relaxation order. First, we use the default order to calculate it. If it fails, we increase the order so that the problem may be solved. For example, for Problem 4.7, GloptiPoly 3 fails to solve it until the order equals to 3. Even though the order equals to 3, only 21 out of 30 times succeed. For the other 9 times, GloptiPoly 3 cannot extract the global optimum from the lower bound. If a problem cannot be solved by the solver GloptiPoly 3 with increasing orders from default order to the order making it out of memory, then success rate is 0/30. From the above table, we can see GloptiPoly 3 solves Problem 4.1-4.6, 4.10, 4.12, 4.14, 4.16, and 4.19 successfully. For Problem 4.7, GloptiPoly 3 solves it 21 times successfully out of 30. For the rest problems, GloptiPoly 3 fails, that is GloptiPoly 3 either does not extract the global optimum from the lower bound, or returns 'out of memory'.

For the large scale Problems 4.20 and 4.21, the method for SDP relaxations in large scale polynomial optimization provided in [81] gave global or approximate global optimal values. By our method GOM, for Problem 4.20, we got the same result with that in [81] and for 4.21, we got better result than that in [81].

Note, all computations in the paper were implemented on a Microsoft Windows XP Desktop of 3.46GB memory and 2.99GHz CPU frequency.

4.6. Conclusion

A necessary global optimality condition for the problem (GP) is provided. A new local optimization method is designed according to the necessary global condition. A global optimization method is designed by combining the new local optimization method and an auxiliary function. The numerical examples illustrate that our methods are efficient and stable.

Chapter 5.

Global optimality conditions and optimization methods for general constrained polynomial programming problems (GPP)

The general constrained polynomial programming problems which are denoted by (GPP) are considered in this chapter. Problems (GPP) have a broad range of applications and are proved to be NP-hard. Necessary global optimality conditions for the problem (GPP) are established. Then, a new local optimization method for the problem (GPP) is proposed by exploiting these necessary global optimality conditions. A global optimization method is proposed for the problem (GPP) by combining this local optimization method together with an auxiliary function. Some numerical examples are also given to illustrate that these approaches are very efficient.

5.1. Introduction

Problems (GPP) are widespread in the mathematical modeling of real world systems for a very broad range of applications. Such applications include engineering design, signal processing, speech recognition, material science, investment science, quantum mechanics, allocation and location problems, quadratic assignment and numerical linear algebra [17, 117]. Since polynomial functions are non-convex, the problem (GPP) is NP-hard, even when the objective function is quadratic and the feasible set is a simplex [81].

A classic approach for the problem (GPP) is convex relaxation methods [30,77,81]. Among various convex relaxation methods, semidefinite programming (SDP) and sum of squares (SOS) relaxations are very popular. As we surveyed in Chapter 1, we know that solving large scale SDP problems still remains a computational challenge.

Recently, some researchers applied SDP relaxation methods to some special models. [14] provided approximation methods for complex polynomial optimization. In [14], the objective function takes three forms: multilinear, homogenous polynomial and a conjugate symmetric form. The constraint belongs to three sets: the m-th roots of complex unity, the complex unity and the Euclidean sphere. [23] established some approximation solution methods to solve a quadratically constrained multivariate bi-quadratic optimization. [139] presented a general semidefinite relaxation scheme for general n-variate quartic polynomial optimization under homogeneous quadratic constraints. [117] considered approximation algorithms for optimizing a generic multi-variate homogeneous polynomial function, subject to homogenous quadratic constraints.

Global optimality conditions are very important in global optimization field. References [65, 66, 76, 125] focus on global optimality conditions for the problems with quadratic objective function subject to linear constraints or quadratic constraints. Based on the so-called Positivstellensatz (a polynomial analogue of the transposition theorem for linear systems), it is possible to formulate global necessary and sufficient conditions for problems (GPP) [54]. [67] proved in Theorem 4.2 a sufficient conditions for global optimality in (GPP), which is a special case of global necessary and sufficient conditions proposed in [54]. [126] provided another necessary and sufficient global optimality conditions for (GPP). However all these conditions are complex and difficult to check in practice since the conditions involve solving a sequence of semidefinite programs. As it mentioned in [54], only under the idealized assumptions that all semidefinite programs can be solved exactly, it is possible for these conditions to be checked.

In this chapter, we consider the following problem (GPP).

$$\begin{aligned} (GPP) \qquad \min \quad & f(x) \\ s.t. \quad & g_t(x) \leq 0,\ t = 1, \cdots, m \\ & x \in X, \end{aligned}$$

where $f : X \to R$, $g_t : X \to R$, $t = 1, \cdots, m$, and X is a box with $x_i \in [u_i, v_i], i = 1, \ldots, n$. $S = \{x \in X | g_t(x) \leq 0,\ t = 1, \cdots, m\}$ is feasible set.

In this chapter, we will discuss necessary global optimality conditions for the problem (GPP). These conditions are obtained by studying KKT conditions and a necessary and sufficient condition for a point being a global minimizer for a constrained univariate polynomial programming problem. Then a new strongly local optimization method will be designed for the problem (GPP) according to the necessary global optimality conditions. The new strongly local optimization method improves traditional local optimization method which is based on KKT conditions. Finally, we will design a global optimization method to solve the problem (GPP) by combining the new strongly local optimization method and an auxiliary function. Numerical examples illustrate the efficiency of the optimization methods proposed in the chapter.

5.2. Necessary global optimality conditions for (GPP)

In this section, we will provide necessary global optimality conditions for the problem (GPP). Actually, we construct a point set where the global minimizer lies in. We can obtain the global minimizer by comparing the function values of all points in the set.

First, we consider the following univariate polynomial optimization.

$$\begin{aligned} (UPP) \quad \min \quad & p(x) \\ s.t. \quad & q_t(x) \leq 0,\ t = 1, \cdots, m \\ & x \in [u, v]. \end{aligned}$$

Let $\Omega = \{x \in [u, v] | q_t(x) \leq 0,\ t = 1, \cdots, m\}$.

The problem (UPP) is interesting not only because of the inherent simplicity of the problem structure and rich modeling capabilities, but also because this problem forms the backbone of multi-variate polynomial optimization [129].

For methods to solve the problem (UPP), please refer to [47, 129] and the papers therein. [129] applies the global optimization algorithm (GOP) which proposed for solving constrained nonconvex problems involving quadratic and polynomial functions in the objective function and/or constraints presented in [19] to the special case of polynomial functions of one variable. It illustrates the effectiveness of the algorithm. [47] presents a significant enhancement of reformulation-linearization technique (RLT) and shows empirically that this approach yield very tight lower bounds.

Since the feasible set Ω is a compact set and is not easy to work out, we will construct a new point set $\Omega^0 \subset \Omega$.

Let $\Omega^1 = \{u, v | q_t(u) \le 0, q_t(v) \le 0,\ t = 1, \cdots, m\}$, $\Omega^2 = \{x | \nabla p(x) = 0, q_t(x) < 0,\ t = 1, \cdots, m, x \in (u, v)\}$ and $\Omega_t^3 = \{x | q_t(x) = 0, q_j(x) \le 0,\ j \ne t,\ j = 1, \cdots, m, x \in (u, v)\}$, $t = 1, \cdots, m$. Let

$$\Omega^0 = \Omega^1 \bigcup \Omega^2 \bigcup_{t=1}^{m} \Omega_t^3. \tag{5.1}$$

Remark 18. *Since $p(x)$ and $q_t(x)$, $t = 1, \cdots, m$, are univariate polynomials, we suppose that the degree of $p(x)$ is dp and the degrees of $q_t(x)$, $t = 1, \cdots, m$, are dq_t, $t = 1, \cdots, m$, respectively. We use following methods to work out these point sets Ω^1, Ω^2 and Ω_t^3, $t = 1, \cdots, m$:*

1. *u and v will be kept if $q_t(u) \le 0$, $q_t(v) \le 0$, $t = 1, \cdots, m$. So, $|\Omega^1| \le 2$;*
2. *Calculate all stationary points of $p(x)$ in an interval (u, v) ($\{x \in (u, v) | \nabla p(x) = 0\}$) which will be kept if $q_t(x) < 0$, for all $t = 1, \cdots, m$. So, $|\Omega^2| \le dp - 1$;*
3. *Calculate all roots of $q_t(x)$ in an interval (u, v) ($\{x \in (u, v) | q_t(x) = 0\}$), $t = 1, \cdots, m$, which will be kept if $q_j(x) \le 0$, $j \ne t$, $j = 1, \cdots, m$. So, $|\Omega^3| \le \sum_{t=1}^{m} dq_t$.*

When it comes to finding roots of a univariate polynomial, we refer to the methods proposed in [26] and [116]. In our implementation, we use command 'roots' in Matlab to calculate all roots.

Proposition 3. *For the problem (UPP), let $\bar{x} \in \Omega$. $\bar{x}$ is a global minimizer of (UPP) over Ω if and only if the following condition holds:*

$$p(\bar{x}) \le p(x), \quad \forall x \in \Omega^0, \tag{5.2}$$

where Ω^0 is defined in (5.1).

Proof. $\Rightarrow$ The proof is obvious since $\Omega^0 \subset \Omega$.

$\Leftarrow$ We suppose that $\bar{x}$ is not a global minimizer of $p(x)$ over Ω and x^* is a global minimizer of $p(x)$ over Ω. So we have $p(x^*) < p(\bar{x})$.

From the condition (5.2), we know that $x^* \in \Omega \setminus \Omega^0$ (which means $x^* \in \Omega$ and $x^* \notin \Omega^0$). By $x^* \notin \Omega^1$, we have $x^* \in (u, v)$. By $x^* \notin \bigcup\limits_{t=1}^{m} \Omega_t^3$, we have $q_t(x^*) < 0$, $t = 1, \cdots, m$. By $x^* \notin \Omega^2$, $x^* \in (u, v)$ and $q_t(x^*) < 0$, $t = 1, \cdots, m$, we have $\nabla p(x^*) \neq 0$.

So, we have the following properties. Let $d = -\nabla p(x^*)$. There exists an $s > 0$, such that

1. $x^* + sd \in (u, v)$;

2. $q_t(x^* + sd) < 0$, for all $t = 1, \cdots, m$;

3. $p(x^* + sd) < p(x^*)$

So we can conclude $x^* + sd \in \Omega$ and $p(x^* + sd) < p(x^*)$, which contradicts that x^* is a global minimizer of $p(x)$ over Ω. $\square$

By using Proposition 3, we will give necessary global optimality conditions for the problem (GPP).

Let $\bar{x} \in S$, Q be an invertible matrix, let

$$x := Qy, \;\; F(y) := f(Qy) = f(x), \;\; \bar{y} := Q^{-1}\bar{x},$$

and let $(Q)_i$ represent the ith row of Q, $(Q)_{ij}$ represent the entry of Q in the ith row and the jth column.

Let $Y = \{y = Q^{-1}x | x \in X\}$. For $\bar{y} = (\bar{y}_1, \ldots, \bar{y}_n)^T = Q^{-1}\bar{x}$, let $y = (\bar{y}_1, \cdots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \cdots, \bar{y}_n)^T$. Let $\triangle_k = \sum\limits_{\substack{j=1 \\ j \neq i}}^{n} (Q)_{kj}\bar{y}_j = \bar{x}_k - (Q)_{ki}\bar{y}_i = \bar{x}_k - (Q)_{ki}(Q^{-1})_i\bar{x}$, $k = 1, \cdots, n$, and let

$$\begin{aligned} l_i &= \max\left\{\min\left\{\frac{u_1 - \triangle_1}{(Q)_{1i}}, \frac{v_1 - \triangle_1}{(Q)_{1i}}\right\}, \cdots, \min\left\{\frac{u_n - \triangle_n}{(Q)_{ni}}, \frac{v_n - \triangle_n}{(Q)_{ni}}\right\}\right\}, \\ r_i &= \min\left\{\max\left\{\frac{u_1 - \triangle_1}{(Q)_{1i}}, \frac{v_1 - \triangle_1}{(Q)_{1i}}\right\}, \cdots, \max\left\{\frac{u_n - \triangle_n}{(Q)_{ni}}, \frac{v_n - \triangle_n}{(Q)_{ni}}\right\}\right\}. \end{aligned}$$

Then we can obtain the following results:

$$(1)\quad l_i \le r_i,$$

$$(2)\quad [l_i, r_i] = \{y_i \mid (\bar{y}_1, \cdots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \cdots, \bar{y}_n)^T \in Y\}.$$

Let $G_t(y_i) = g_t(Qy) = g_t(x)$. We have $S_i^1 = \{l_i, r_i | G_t(l_i) \le 0, G_t(r_i) \le 0 | t = 1, \cdots, m\}$, $S_i^2 = \{y_i | \nabla f(Qy) = 0, g_t(Qy) < 0,\ t = 1, \cdots, m, y = (\bar{y}_1, \cdots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \cdots, \bar{y}_n)^T, y_i \in (l_i, r_i)\}$ and $S_{t,i}^3 = \{y_i | g_t(Qy) = 0, g_j(Qy) \le 0,\ j \ne t,\ j = 1, \cdots, m, y = (\bar{y}_1, \cdots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \cdots, \bar{y}_n)^T, y_i \in (l_i, r_i)\}$, $t = 1, \cdots, m$, $\forall i = 1, \cdots, n$. Let

$$S_i^0 = S_i^1 \bigcup S_i^2 \bigcup_{t=1}^{m} S_{t,i}^3. \tag{5.3}$$

Let us review KKT conditions for the problem (GPP).

If $\bar{x}$ is a local optimal solution, then the following KKT conditions hold under some constraint qualifications: there exist nonnegative scalars α_t, $t = 1, \cdots, m$, β_i and γ_i, $i = 1, \cdots, n$, such that

$$[KKT]\begin{cases} \nabla f(\bar{x}) + \sum\limits_{t=1}^{m} \alpha_t \nabla g_t(\bar{x}) + \beta - \gamma = 0, \\ \alpha_t g_t(\bar{x}) = 0,\ t = 1, \cdots, m \\ \beta(x - v) = 0 \\ \gamma(u - x) = 0 \end{cases},$$

where $\beta = (\beta_1, \cdots, \beta_n)^T$ and $\gamma = (\gamma_1, \cdots, \gamma_n)^T$. See [100] for various constraint qualifications, such as Abadie constraint qualification, linearity constraint qualification, Slater's constraint qualification, linear independence constraint qualification, Cottle's constraint qualification, Zangwill's constraint qualification, Kuhn-Tucker's constraint qualification.

Theorem 12. *(Necessary global optimality conditions for (GPP)) Let $\bar{x} \in S$ and Q be any* *invertible matrix. If $\bar{x}$ is a global minimizer of (GPP), then the following conditions hold:*

$$[GNC]\begin{cases} [KKT]\ conditions\ hold\ under\ some\ constraint\ qualifications; \\ [NC]_i:\ f(\bar{x}) \le f(x),\ \forall (Q^{-1})_i x \in S_i^0,\ \forall i = 1, \cdots, n. \end{cases}$$

where S_i^0 is defined in (5.3).

Proof. If $\bar{x}$ is a global minimizer of (GPP), then it is also a local minimizer of (GPP). So under some constraint qualifications, KKT conditions hold.

Next, we prove conditions $[NC]_i$, $i = 1, \cdots, n$ hold. If $\bar{x}$ is a global minimizer of (GPP), then $f(\bar{x}) \le f(x)$, for any $x \in S$.

Let $\bar{y} = Q\bar{x}$. For any $y = (\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T \in Y$, i.e., $y_i \in [l_i, r_i], \forall i = 1, \ldots, n$, let $x = Qy$. Then $x \in X$. So we have $f(Q\bar{y}) \le f(Qy)$, for any $y_i \in [l_i, r_i], \forall i = 1, \ldots, n$. By using Proposition 3, we have the following conditions $[NC]_i$ hold:

$$[NC]_i \quad f(\bar{x}) \le f(x),\ \forall (Q^{-1})_i x \in S_i^0,\ \forall i = 1, \cdots, n.$$

□

Remark 19. *From Theorem 12, we can see the global optimality conditions $[GNC]$ are stronger than KKT conditions, since $[GNC]$ include KKT conditions.*

Next, we take Problem 5.8 in section 5.4 for example to show $[KKT] \not\supseteq [NC]_i$, $\forall i = 1, \cdots, n$ below.

We fix $Q = I$ and choose two points $\bar{x} = (2.3295, 3.1785)^T$ which is a global minimizer and $\bar{y} = (1.5996, 2.8204)^T$ which is a local minimizer. It is easy to check that both $[NC]_i$ and $[KKT]$ hold at $\bar{x}$, while $[KKT]$ holds at $\bar{y}$, but $[NC]_i$ does not hold at $\bar{y}$.

In fact, $\bar{x} \in int(X)$, $\nabla f(\bar{x}) = (-1, -1)^T$ and $g_1(\bar{x}) = g_2(\bar{x}) = 0$, which means $\bar{x} \in S_{t,i}^3 \subset S_i^0$, $t = 1, 2$, $i = 1, 2$.

When $i = 1$ and we fix $\bar{x}_2 = 3.1785$, we have $S_1^1 = \emptyset$, $S_1^2 = \emptyset$, $S_{1,1}^3 = \{2.3295, 0.5179\}$

and $S^3_{2,1} = \{2.3295, 0.6247\}$. *But* $f((0.5179, 3.1785)^T) = -3.6964 > f(\bar{x}) = -5.5080$ *and* $f((0.6247, 3.1785)^T) = -3.8033 > f(\bar{x}) = -5.5080$. *So* $f(\bar{x}) \leq f(x)$, $\forall x \in S^0_1 = S^1_1 \bigcup S^2_1 \bigcup\limits_{t=1}^{2} S^3_{t,1}$.

When $i = 2$ *and we fix* $\bar{x}_1 = 2.3295$, *we have* $S^1_2 = \{0\}$, $S^2_2 = \emptyset$, $S^3_{1,2} = \{3.1785\}$ *and* $S^3_{2,2} = \{3.1785\}$. *But* $f((2.3295, 0)^T) = -2.3295 > f(\bar{x}) = -5.5080$. *So* $f(\bar{x}) \leq f(x)$, $\forall x \in S^0_2 = S^1_2 \bigcup S^2_2 \bigcup\limits_{t=1}^{2} S^3_{t,2}$.

This means conditions $[NC]_i$, $i = 1, 2$, *hold at* $\bar{x}$.

Since $\nabla g_1(\bar{x}) = (-8.1639, 1)^T$ *and* $\nabla g_2(\bar{x}) = (4.6996, 1)^T$, *we can find nonnegative scalars* $\alpha_1 = 0.2876$ *and* $\alpha_2 = 0.7124$ *such that* $[KKT]$ *holds at* $\bar{x}$.

While $\bar{y} \in int(X)$, $\nabla f(\bar{y}) = (-1, -1)^T$ *and* $g_1(\bar{y}) = g_2(\bar{y}) = 0$, *which means* $\bar{y} \in S^3_{t,i} \subset S^0_i$, $t = 1, 2$, $i = 1, 2$.

When $i = 1$ *and we fix* $\bar{y}_2 = 2.8204$, *we have* $S^1_1 = \emptyset$, $S^2_1 = \emptyset$, $S^3_{1,1} = \{2.2808, 1.5996, 0.4004\}$. $f((2.2808, 2.8204)^T) = -5.1012$, $f((1.5996, 2.8204)^T) = -4.4200$ *and* $f((0.4004, 2.8204)^T) = -3.2208$. *So* $f(\bar{y}) \leq f(x)$, $\forall x \in S^0_1 = S^1_1 \bigcup S^2_1 \bigcup\limits_{t=1}^{2} S^3_{t,1}$ *does not hold at* $\bar{y}$. *This means* $[NC]_1$ *does not hold at* $\bar{y}$.

Since $\nabla g_1(\bar{y}) = (3.0723, 1)^T$ *and* $\nabla g_2(\bar{x}) = (-5.3793, 1)^T$, *we can find nonnegative scalars* $\alpha_1 = 0.7548$ *and* $\alpha_2 = 0.2452$ *such that* $[KKT]$ *holds at* $\bar{y}$.

5.3. Optimization methods for (GPP)

5.3.1. New local optimization method for (GPP)

Definition 24. *Let* $\bar{x} \in S$ *and* Q *be an invertible matrix.* $\bar{x}$ *is said to be a strongly local minimizer of the problem* (GPP) *with respect to* Q *iff* $\bar{x}$ *satisfies the necessary global optimality conditions* $[GNC]$.

Definition 25. *Let* $\bar{x} \in S$ *and* Q *be an invertible matrix.* $\bar{x}$ *is said to be a* $\varepsilon-$*strongly local* *minimizer of the problem (GPP) with respect to Q iff KKT conditions hold at $\bar{x}$ and for any $i = 1, \cdots, n$, either $\bar{x}$ satisfies the condition $[NC]_i$ or there exists a point $X_i^* \in S$, such that X_i^* satisfies the condition $[NC]_i$ when $\bar{x}$ is replaced by X_i^*, and $|f(\bar{x}) - f(X_i^*)| \leq \varepsilon$.*

Algorithm 11. *Strongly or $\varepsilon-$strongly local optimization method for (GPP):$(SLOM)$.*

Step 0. *Take an initial point $x_0 \in S$. Let $Q_1 = I$, Q_2, $\cdots$, Q_s, $\cdots$, Q_N be any invertible matrices given randomly, where I is the identity matrix. Let ε be a small positive number. Let $s := 1$ and $Q := Q_s$ and $i = 1$. Let $x^* := (x_1^*, \cdots, x_n^*)^T$ be a local minimizer or KKT point of $f(x)$ on feasible set S starting from $\bar{x}$. Let $\bar{x} := x^*$ and go to Step 1.*

Step 1. *Let $\bar{y} = Q^{-1}\bar{x} = (\bar{y}_1, \ldots, \bar{y}_i, \ldots, \bar{y}_n)^T$, $y = (\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T$ and $x = Qy$. Calculate S_i^1, and then check whether the condition holds:*

$$f(\bar{x}) \leq f(Qy) + \varepsilon,\ \forall y = (\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T \ and\ y_i \in S_i^1.$$

If this condition holds, go to Step 2, otherwise set $\tilde{S} = S_i^1$ and go to Step 4.

Step 2. *Calculate S_i^2, and then check whether the condition holds:*

$$f(\bar{x}) \leq f(Qy) + \varepsilon,\ \forall y = (\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T \ and\ y_i \in S_i^2.$$

If this condition holds, go to Step 3, otherwise set $\tilde{S} = S_i^2$ and go to Step 4.

Step 3. *Set $t = 1$. Calculate $S_{t,i}^3$, and then check whether the condition holds:*

$$f(\bar{x}) \leq f(Qy) + \varepsilon,\ \forall y = (\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T \ and\ y_i \in S_{t,i}^3.$$

If the condition holds, set $t = t + 1$ and repeat to check the condition until $t = m$ and go to Step 5; otherwise set $\tilde{S} = S_{t,i}^3$ and go to Step 4.

Step 4. *Let* $\bar{y}_i^* := argmin\{f(Qy)|y = (\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T$ *and* $y_i \in \tilde{S}\}$ *and* $\bar{y}^* = (\bar{y}_1, \cdots, \bar{y}_{i-1}, \bar{y}_i^*, \bar{y}_{i+1} \cdots, \bar{y}_n)^T$. *Let* $\bar{x}^* := Q\bar{y}^*$. *Let* $x^* = (x_1^*, \cdots, x_n^*)$ *be a local minimizer or* KKT *point of* $f(x)$ *on* S *starting from* $\bar{x}^*$. *If* $f(x^*) < f(\bar{x}) - \varepsilon$, *let* $\bar{x} := x^*$, $i := 1$, $s := 1$ *and* $Q := Q_s$, *go to Step 1; otherwise go to Step 5.*

Step 5. *If* $i := n$, *go to Step 6; otherwise, let* $i := i + 1$ *and go to Step 1.*

Step 6. *Let* $s := s + 1$. *If* $s > N$, *go to Step 7; otherwise, let* $Q := Q_s$ *and* $i := 1$, *go to Step 1.*

Step 7. *Stop.* $\bar{x}$ *is a strongly or* $\varepsilon-$*strongly local minimizer with respect to* Q_s, $s = 1, \cdots, N$.

Remark 20. *In step 0 and step 4, we can apply any local optimization algorithm to get local minimizer or KKT point, such as feasible direction methods, penalty function methods, starting from* $\bar{x}$. *In our implementation, the optimization subroutine 'fmincon' within the optimization Toolbox in Matlab is used as the local search scheme to obtain local minimizers. In step 1, step 2 and step 3, we need to calculate* S_i^1, S_i^2 *and* $S_{t,i}^3$, $t = 1, \cdots, m$. *For any* i, $i \in \{1, \cdots, n\}$, *let* $\bar{x} \in S$, $\bar{y} = Q^{-1}\bar{x}$ *and* $y = (\bar{y}_1, \cdots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \cdots, \bar{y}_n)^T$, *where* $y_i \in [l_i, r_i]$. *Then* $f(Qy)$ *and* $g_t(Qy)$, $t = 1, \cdots, m$, *are univariate polynomials. So, we refer to Remark 18 to calculate these point sets.*

Theorem 13. *For a given initial point* $x_0 \in S$, *we can obtain a strongly or* $\varepsilon-$*strongly local minimizer* $\bar{x}$ *of the problem* (GPP) *in finite iteration times by the given strongly local optimization method* $(SLOM)$.

Proof: First, we can prove that this algorithm must stop in finite iteration times.
Let $M := max\{f(x)|x \in S\}$ and $m := min\{f(x)|x \in S\}$. For the given Q_s, there are at most $n\frac{M-m}{\varepsilon}$ iteration times from *step 1* to *step 5*. In fact, for the given Q_s and given i, if $[NC]_i$ holds or if $f(x^*) \geq f(\bar{x}) - \varepsilon$, then we will change the i into $i + 1$; only when $[NC]_i$ does not hold and $f(x^*) < f(\bar{x}) - \varepsilon$, we will change i to 1 in *step 4* and go to *step 1*. For the same Q_s, when we change i to 1, the objection function value will decrease at least ε.

Hence, there are at most $\frac{M-m}{\varepsilon}$ times to change i to 1 in *step 4*. The total iteration time from *step 1* to *step 5* is at most $n\frac{M-m}{\varepsilon}$. Since we have N numbers of Q_s , this algorithm must stop at most $Nn\frac{M-m}{\varepsilon}$ iteration times.

Second, let L be the set of all the KKT points of the problem (GPP), and let $L_f := \{f(x) \mid x \in L\}$. We can prove that

(1) If L_f is a finite set, then we can obtain a strongly local minimizer in finite iteration times when ε is a very small number. In fact, let $\eta := min\{|f(x) - f(y)| \mid x, y \in L \text{ and } f(x) \neq f(y)\}$. Since L_f is a finite set, we have that $\eta > 0$. When $\varepsilon < \eta$, we know that $f(x^*) < f(\bar{x}) - \varepsilon$ in *step 4* is equivalent to $f(x^*) < f(\bar{x})$. Hence, for the given Q_s and given i, if $[NC]_i$ holds, then we will change the i into $i + 1$; if $[NC]_i$ does not hold in *step 1* or *step 2* or *step 3* which means that $f(\bar{x}) > min\{f(Qy)|y = (\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T \; and \; y_i \in \tilde{S}\}$, then in *step 4*, we will find point $\bar{y}_i^*$ such that $f(Q\bar{y}^*) = min\{f(Qy)|y = (\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T \; and \; y_i \in \tilde{S}\}$. Hence, we have that $f(x^*) < f(\bar{x})$ since $f(x^*) \leq f(Q\bar{y}^*) < f(\bar{x})$ and we have $x^* \in L$. Therefore, for the given Q_s and given i, if $[NC]_i$ does not hold in *step 1* or *step 2* or *step 3*, then we can obtain a new KKT point x^* such that $f(x^*) < f(\bar{x})$ which also satisfies that $f(x^*) < f(\bar{x}) - \varepsilon$. Hence, for the given Q_s, we can find a point $\bar{x}$ which satisfies all the condition $[NC]_i, i = 1, \ldots, n$ in at most $n\frac{M-m}{\varepsilon}$ iteration times. Therefore, in finite times, we can obtain a strongly local minimizer of the problem (GPP) for all $Q_s, s = 1, \ldots, N$.

(2) If L_f is an infinite set, then we can obtain an $\varepsilon-$ strongly local minimizer in finite iteration times.

By the algorithm, for the given Q_s and given i, if $[NC]_i$ holds or if $f(x^*) \geq f(\bar{x}) - \varepsilon$, then we will change the i into $i + 1$; if $[NC]_i$ does not hold and $f(x^*) < f(\bar{x}) - \varepsilon$, then in *step 4*, we will find point $\bar{y}_i^*$ such that $f(Q\bar{y}^*) = min\{f(Qy)|y = (\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T \; and$

$y_i \in \tilde{S}\}$, where $\bar{y}_i^*$ satisfies condition $[NC]_i$. Since this algorithm must stop in finite steps, the final obtained point $\bar{x}$ must satisfy the following condition: for the given Q_s and given i, $[NC]_i$ holds or $f(Q\bar{y}^*) \geq f(x^*) \geq f(\bar{x}) - \varepsilon$, where $\bar{y}_i^*$ satisfies the condition $[NC]_i$. Hence $\bar{x}$ is an $\varepsilon-$ strongly local minimizer of the problem (GPP). $\square$

5.3.2. Global optimization method for (GPP)

In this section, we will design a global optimization method for the problem (GPP) by combining the strongly local optimization method and an auxiliary function. In this chapter, we still use the auxiliary function which was presented by (1.2) in Chapter 1. For the properties of this auxiliary function, see Chapter 1.

Algorithm 12. *Global optimization method for* (GPP)*:*(GOM).

Step 0. *Set* $M := 10^{10}$, $\mu := 10^{-10}$ *and* $k_0 := 2n$. *Set* $A_{n\times n} := I_{n\times n}$ *and* $B_{n\times 2n} := [A, -A]$. *Let* $r_0 := 1$, $c_0 := 1$, $q_0 := 10^5$ *and* $\delta_0 := \frac{1}{2}$. *Let* $k := 1$, $i := 1$ *and* $r := r_0$. *Let* x_1^0 *be an initial point and* $x_0^* := x_1^0$, *then go to Step 1.*

Step 1. *Use the strongly or* $\varepsilon-$*strongly local optimization method* $(SLOM)$ *to solve the problem* (GPP) *starting from* x_k^0. *Let* x_k^* *be the obtained strongly or* $\varepsilon-$*strongly local minimizer of the problem* (GPP). *If* $f(x_k^*) \geq f(x_0^*)$, *then go to step 6; otherwise let* $q := q_0$, $c := c_0$, $r := r_0$, $\delta := \delta_0$, $i := 1$ *and* $x_0^* := x_k^*$, $k := k+1$, *then go to Step 2.*

Step 2. *Let* B_i *indicate the* i*th column of* B *and* $\bar{x}_k^* := x_0^* + \delta B_i$. *If* $\bar{x}_k^* \notin S$, *go to Step 3. Otherwise, if* $f(\bar{x}_k^*) < f(x_0^*)$, *then set* $x_{k+1}^0 := \bar{x}_k^*$ *and* $x_0^* := \bar{x}_k^*$, $k := k+1$ *and go to Step 1; else go to Step 4.*

Step 3. *If* $\delta < \mu$, *go to Step 8; otherwise, let* $\delta = \frac{\delta}{2}$ *and go to Step 2.*

Step 4. *If* $f(x_0^*) \leq f(\bar{x}_k^*) \leq f(x_0^*) + 1$, *then go to Step 5; otherwise let* $\delta = \frac{\delta}{2}$ *go to Step 2.*

Step 5. *Let*

$$F_{q,r,c,x_0^*}(x) = q\left(exp(-\frac{\|x - x_0^*\|^2}{q})g_{r,c}\Big(f(x) - f(x_0^*)\Big) + h_{r,c}\Big(f(x) - f(x_0^*)\Big)\right).$$

Solve the problem:

$$\begin{aligned} \min \quad & F_{q,r,c,x_0^*}(x) \\ s.t. \quad & x \in S. \end{aligned} \tag{5.4}$$

by a local search method starting from the initial point $\bar{x}_k^*$*. Let* $\bar{x}_{q,r,c,x_k^*}$ *be the local minimizer obtained. Then set* $x_{k+1}^0 := \bar{x}_{q,r,c,x_k^*}$*,* $k := k+1$ *and go to Step 1.*

Step 6. *If* $q < M$*, then increase* q *(in the following examples, let* $q := 10q$*), then go to Step 5; otherwise go to Step 7.*

Step 7. *If* $c < M$*, then increase* c *(in the following examples, let* $c := 10c$*), and let* $q := q_0$*, then go to Step 5; otherwise go to Step 8.*

Step 8. *If* $i < k_0$*, then let* $i := i + 1$*,* $q := q_0$*,* $c := c_0$*,* $\delta = \delta_0$*, go to Step 2; otherwise go to Step 9.*

Step 9. *If* $r > \mu$*, then decrease* r *(in the following examples, let* $r := \frac{r}{10}$*). Randomly select an orthogonal matrix* $A_{n\times n}$ *and set* $B_{n\times 2n} := [A, -A]$*. Let* $i := 1$*,* $q := q_0$*,* $c := c_0$*,* $\delta = \delta_0$ *and go to Step 2; otherwise, stop and* x_0^* *is the obtained global minimizer or approximate global minimizer of the problem* (GPP)*.*

5.4. Numerical examples

In this section, we apply our two Algorithms: strongly local optimization method (SLOM) and global optimization method (GOM) to fifteen test problems. Table 5.1 shows summary information of the fifteen test problems. These test problems include Problems 5.1,5.6-5.9

and 5.14 from the book [17], 5.10-5.12 form the paper [77] and 5.2-5.5, 5.13, 5.15 from the website below:

$http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files$
$/TestGO_files/Page422.htm.$

For the detailed information of these problems, see the appendix in the end.

Table 5.1.: Test problems for (GPP)

Number of problems	Global minimizer x^*	Optimal value $f(x^*)$
5.1	$(0.5, 0, 3)$	-4
5.2	$(1, \cdots, 1, 3, 3, 3, 1)$	-15
5.3	$(2.171996, 2.363683, 8.773926, 5.095984, 0.9906548,$ $1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$	24.3062091
5.4	$(14.095, 0.84296)$	-6961.81388
5.5	$(2.330499, 1.951372, -0.4775414, 4.365726,$ $-0.6244870, 1.038131, 1.594227)$	680.6300573
5.6	$(5, 1, 5, 0, 5, 10)$	-310
5.7	$(78, 33, 29.9953, 45, 36.7758)$	-30665.5387
5.8	$(2.3295, 3.1783)$	-5.5079
5.9	$(579.3167, 1359.943, 5110.071, 182.0174,$ $295.5985, 217.9799, 286.4162, 395.5979)$	7049.3307
5.10	†1	-575.5928
5.11	†2	-1.0178
5.12	†3	-153.6180
5.13	$\pm(1/2^{0.5}, 1/2)$	0.75
5.14	$(40.71751, 1.470)$	-16.73889
5.15	$(1/n^{0.5}, \cdots, 1/n^{0.5})$	-1

†1 = −(0.4034, 0.4274, 0.4486, 0.4674, 0.4839, 0.4983, 0.5107, 0.5211, 0.5296, 0.5363, 0.5410, 0.5437, 0.5444, 0.5430, 0.5393);
†2 = −(0.2418, 0.2208, 0.2085, 0.2000, 0.1934, 0.1882, 0.1838, 0.1800, 0.1767, 0.1738, 0.1712, 0.1688, 0.1667, 0.1647, 0.1629, 0.1612);
†3 = −(-0.3642, 0.3955, 0.5042, 0.5589, 0.5892, 0.6049, 0.6109, 0.6104, 0.6057, 0.5991, 0.5828, 0.5173, 0.5193, 0.5306, 0.5459, 0.5619, 0.5763, 0.5869, 0.5919, 0.5896).

There are equalities involved in Problem 5.13-5.15. We can use our algorithms to solve them by converting equalities $h_s(x) = 0$, $s = 1, \cdots, l$ into equivalent inequalities $h_s(x) \leq 0$, $s = 1, \cdots, l$ and $-h_s(x) \leq 0$, $s = 1, \cdots, l$.

For our experiments, we use the optimality gap mentioned in [97] is:

$$GAP = |f(x) - f(x^*)|$$

where x is a heuristic solution obtained by our method and x^* is the optimal solution. We then say that a heuristic solution x is optimal if:

$$GAP \leq \begin{cases} \varepsilon & f(x^*) = 0 \\ \varepsilon \times |f(x^*)| & f(x^*) \neq 0 \end{cases}$$

In our experimentation we set $\varepsilon = 0.001$ as the same of that in [97].

In the table below, some common statistics are included. We randomly select 30 initial points for every problem. The *suc.rate*(success rate) means the success times out of 30. The *best* is the minimum of the results, the *worst* indicates the maximum of the results, and then it follows the *mean*, *median* and *st.dev.*(standard deviation). In some way, these statistics are able to evaluate the search ability and solution accuracy, reliability and convergence as well as stability.

Table 5.2.: Results of algorithms SLOM and GOM for (GPP)

Problem	statistic	SLOM	GOM
5.1	suc.rate	$30/30$	$30/30$
	best	-4.0000	-4.0000
	worst	-4.0000	-4.0000
	mean	-4.0000	-4.0000
	median	-4.0000	-4.0000
	st.dev	$2.7262e-006$	$2.7262e-006$

continue goes here…

Problem	statistic	SLOM	GOM
5.2	suc.rate	$30/30$	$30/30$
	best	-15.0000	-15.0000
	worst	-15.0000	-15.0000
	mean	-15.0000	-15.0000
	median	-15.0000	-15.0000
	st.dev	$8.9121e-006$	$8.9121e-006$
5.3	suc.rate	$30/30$	$30/30$
	best	24.3062	24.3062
	worst	24.3062	24.3062
	mean	24.3062	24.3062
	median	24.3062	24.3062
	st.dev	$4.9274e-006$	$4.9274e-006$
5.4	suc.rate	$30/30$	$30/30$
	best	$-6.9618e+003$	$-6.9618e+003$
	worst	$-6.9618e+003$	$-6.9618e+003$
	mean	$-6.9618e+003$	$-6.9618e+003$
	median	$-6.9618e+003$	$-6.9618e+003$
	st.dev	$8.0994e-004$	$8.0994e-004$
5.5	suc.rate	$30/30$	$30/30$
	best	680.6301	680.6301
	worst	680.6301	680.6301
	mean	680.6301	680.6301
	median	680.6301	680.6301

continue goes here…

Problem	statistic	SLOM	GOM
	st.dev	$5.3698e-006$	$5.3698e-006$
5.6	suc.rate	$26/30$	$30/30$
	best	-310.0000	-310.0000
	worst	-184.0000	-310.0000
	mean	-293.2000	-310.0000
	median	-310.0000	-310.0000
	st.dev	43.5640	$5.9702e-006$
5.7	suc.rate	$30/30$	$30/30$
	best	$-3.0666e+004$	$-3.0666e+004$
	worst	$-3.0666e+004$	$-3.0666e+004$
	mean	$-3.0666e+004$	$-3.0666e+004$
	median	$-3.0666e+004$	$-3.0666e+004$
	st.dev	$4.4270e-004$	$4.4270e-004$
5.8	suc.rate	$30/30$	$30/30$
	best	-5.5080	-5.5080
	worst	-5.5080	-5.5080
	mean	-5.5080	-5.5080
	median	-5.5080	-5.5080
	st.dev	$9.9335e-007$	$9.9335e-007$
5.9	suc.rate	$29/30$	$30/30$
	best	$7.0492e+003$	$7.0492e+003$
	worst	$8.7331e+003$	$7.0492e+003$
	mean	$7.1054e+003$	$7.0492e+003$
	median	$7.0492e+003$	$7.0492e+003$

continue goes here…

Problem	statistic	SLOM	GOM
	st.dev	307.4294	$1.0895e-006$
5.10	suc.rate	$30/30$	$30/30$
	best	-575.5925	-575.5925
	worst	-575.5925	-575.5925
	mean	-575.5925	-575.5925
	median	-575.5925	-575.5925
	st.dev	$1.9967e-006$	$1.9967e-006$
5.11	suc.rate	$7/30$	$30/30$
	best	-1.1078	-1.1078
	worst	-0.0108	-1.1078
	mean	-0.2692	-1.1078
	median	-0.0144	-1.1078
	st.dev	0.4706	$1.6607e-014$
5.12	suc.rate	$30/30$	$30/30$
	best	-153.6180	-153.6180
	worst	-153.6180	-153.6180
	mean	-153.6180	-153.6180
	median	-153.6180	-153.6180
	st.dev	$7.5214e-007$	$7.5214e-007$
5.13	suc.rate	$30/30$	$30/30$
	best	0.7500	0.7500
	worst	0.7500	0.7500
	mean	0.7500	0.7500
	median	0.7500	0.7500

continue goes here…

Problem	statistic	SLOM	GOM
	st.dev	$6.2234e-009$	$6.2234e-009$
5.14	suc.rate	$30/30$	$30/30$
	best	-16.7389	-16.7389
	worst	-16.7389	-16.7389
	mean	-16.7389	-16.7389
	median	-16.7389	-16.7389
	st.dev	$5.8438e-007$	$5.8438e-007$
5.15	suc.rate	$30/30$	$30/30$
	best	-1.0000	-1.0000
	worst	-1.0000	-1.0000
	mean	-1.0000	-1.0000
	median	-1.0000	-1.0000
	st.dev	$8.2074e-007$	$8.2074e-007$

It is shown from table 5.2 that GOM successfully solves all number of test problems and is very efficient and stable. As a local optimization method, SLOM can also be considered as a competitive algorithm with producing impressive results.

Next, we try to compare our GOM method with the solver GloptiPoly 3 which is a Matlab/SeDuMi add-on for SDP-relaxations of minimization problems over multivariable polynomial functions subject to polynomial or integer constraints [31, 32].

Table 5.3.: Comparisons between GOM and Gloptipoly 3 for (GPP)

Problem	statistic	GOM	GloptiPoly 3
5.1	suc.rate	$30/30$	$30/30$
			order$=4$
	best	-4.0000	-4.0000
	worst	-4.0000	-4.0000
5.2	suc.rate	$30/30$	$30/30$
			order$=2$
	best	-15.0000	-15.0000
	worst	-15.0000	-15.0000
5.3	suc.rate	$30/30$	$30/30$
	best	24.3062	24.3062
	worst	24.3062	24.3062
5.4	suc.rate	$30/30$	$30/30$
	best	$-6.9618e+003$	$-6.9618e+003$
	worst	$-6.9618e+003$	$-6.9618e+003$
5.5	suc.rate	$30/30$	$30/30$
			order$=3$
	best	680.6301	680.6301
	worst	680.6301	680.6301
5.6	suc.rate	$30/30$	$30/30$
			order$=2$
	best	-310.0000	-309.9998

continue goes here...

Problem	statistic	GOM	GloptiPoly 3
	worst	-310.0000	-309.9998
5.7	suc.rate	$30/30$	$0/30$
	best	$-3.0666e+004$	-
	worst	$-3.0666e+004$	-
5.8	suc.rate	$30/30$	$30/30$
			order$=4$
	best	-5.5080	-5.5079
	worst	-5.5080	-5.5079
5.9	suc.rate	$30/30$	$0/30$
	best	$7.0492e+003$	-
	worst	$7.0492e+003$	-
5.10	suc.rate	$30/30$	$0/30$
	best	-575.5925	-
	worst	-575.5925	-
5.11	suc.rate	$30/30$	$0/30$
	best	-1.1078	-
	worst	-1.1078	-
5.12	suc.rate	$30/30$	$0/30$
	best	-153.6180	-
	worst	-153.6180	-
5.13	suc.rate	$30/30$	$30/30$
			order$=3$
	best	0.7500	0.7500

continue goes here...

Problem	statistic	GOM	GloptiPoly 3
	worst	0.7500	0.7500
5.14	suc.rate	$30/30$	$30/30$
	best	-16.7389	-16.7389
	worst	-16.7389	-16.7389
5.15	suc.rate	$30/30$	$0/30$
	best	-1.0000	-
	worst	-1.0000	-

In the table 5.3, we use the solver GloptiPoly 3 to solve Problem 5.1-5.15. We run Gloptipoly 3 30 times for each problem with fixed relaxation order. First, we use the default order to calculate it. If it fails, we increase the order so that the problem may be solved. For example, for Problem 5.1, GloptiPoly 3 fails to solve it until the order equals to 4. If a problem cannot be solved by the solver GloptiPoly 3 with increasing orders from default order to the order making it out of memory, then the success rate is 0/30. From the above table, we can see GloptiPoly 3 solves Problem 5.1-5.6, 5.8, 5.13-5.14 successfully. For the rest problems, GloptiPoly 3 fails, and returns 'out of memory'.

For the large scale Problem 5.10-5.12, the regularization methods for SOS relaxations in large scale polynomial optimization provided in [77] gave global or approximate global optimal values. By our method GOM, we got the same results with those obtained in [77].

Note, all computations in the paper were implemented on a Microsoft Windows XP Desktop of 3.46GB memory and 2.99GHz CPU frequency.

5.5. Conclusion

We study a necessary and sufficient condition for a point being a global minimizer for a constrained univariate polynomial programming problem. Necessary global optimality conditions for the problem (GPP) are provided based on this necessary and sufficient condition. A new local optimization method is designed according to these necessary global conditions which improve the traditional local optimization method (based on KKT conditions). A new global optimization method is designed by combining the new local optimization method and an auxiliary function. The numerical examples illustrate that our methods are efficient and stable.

Chapter 6.

Applications

In this chapter, we will discuss some applications for solving sensor network localization problems and systems of polynomial equations. In particular, we will apply the idea and the results for polynomial programming problems presented in chapter 2, 3, 4 and 5 to nonlinear programming problems (NLP).

6.1. Sensor network localization problems

6.1.1. Introduction

Sensor network localization which is an important problem in communication and information theory has drawn much attention recently. The basic description of this problem is as follows. There is a sequence of unknown vectors (also called sensors) $x_1, \cdots, x_n$ in Euclidean space R^d for a given dimension d. The goal is to place these vectors such that the Euclidean distances between these sensors and the distances to other fixed sensors $a_1, \cdots, a_m$ (also called anchors) are equal to the prescribed numbers [7, 78]. To be more specific, let $A = \{(i,j) \in [n] \times [n] : \|x_i - x_j\|_2 = d_{ij}\}$ and $B = \{(i,k) \in [n] \times [m] : \|x_i - a_k\|_2 = e_{ik}\}$, where d_{ij}, e_{ik} are prescribed distances and $[n] = \{1, \cdots, n\}$. Then the problem of sensor network localization is to place the vectors $\{x_1, \cdots, x_n\}$ such that $\|x_i - x_j\|_2 = d_{ij}$ for every $(i, j) \in A$ and $\|x_i - a_k\|_2 = e_{ik}$ for every $(i, k) \in B$ [78].

In [78], the author formulated the sensor network localization problem as finding the global minimizer of a quartic polynomial.

$$\min_{X \in R^{d\times n}} \ f(X) := \sum_{(i,j)\in A} (\|x_i - x_j\|_2^2 - d_{ij}^2)^2 + \sum_{(i,k)\in B} (\|x_i - a_k\|_2^2 - e_{ik}^2)^2$$

where d_{ij},e_{ik} are given distances and $\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$. Therefore, we will solve some sensor network localization problems by using our optimization methods: *Algorithm* 6 (SLOM) and *Algorithm* 7 (GOM) provided in chapter 3.

6.1.2. Numerical examples

Example 11. *Consider a simple example studied in [78] and [108], with $n = 1$, $d = 2$, $m = 2$. $A = \emptyset$, $B = \{(1, 1), (1, 2)\}$, $d_{11} = d_{12} = 2$. The anchors are $(\pm 1, 0)$. In [78], this problem becomes to a quartic polynomial problem:*

$$\min \quad p(x_{11}, x_{21}) := \quad ((x_{11} + 1)^2 + x_{21}^2 - 4)^2 + ((x_{11} - 1)^2 + x_{21}^2 - 4)^2$$

By our Algorithm *6 with $Q = I$, we can get the global solution are $(0.0000, \pm 1.7321)$ which are the same as the solutions given in [78] and [108].*

Example 12. *Consider another example studied in [78], with four sensors and four anchors*

$$a_1 = (1, 1)^T, \ a_2 = (1, -1)^T, \ a_3 = (-1, -1)^T, \ a_4 = (-1, 1)^T.$$

The network is as follows

$$A = \{(1, 2), (1, 4), (2, 3), (3, 4)\}, \ B = \{(1, 1), (2, 2), (3, 3), (4, 4)\}.$$

The distances are given by

$$d_{12} = d_{14} = d_{23} = d_{34} = s = 2 - \sqrt{2},\ e_{11} = e_{22} = e_{33} = e_{44} = 1.$$

Let $X = [x_1, x_2, x_3, x_4]$. *This problem becomes to a quartic polynomial problem:*

$$\begin{aligned} \min \quad f(X) := \quad & (\|x_1 - x_2\|^2 - s^2)^2 + (\|x_1 - x_4\|^2 - s^2)^2(\|x_2 - x_3\|^2 - s^2)^2 \\ & +(\|x_3 - x_4\|^2 - s^2)^2 + (\|x_1 - a_1\|^2 - 1)^2 + (\|x_2 - a_2\|^2 - 1)^2 \\ & +(\|x_3 - a_3\|^2 - 1)^2 + (\|x_4 - a_4\|^2 - 1)^2 \end{aligned}$$

By our Algorithm *6 with* $Q = I$,*, we can get the global solution are*

$$x_1 = (0.2929, 0.2929)^T,\ x_2 = (0.2929, -0.2929)^T,$$
$$x_3 = (-0.2929, -0.2929)^T,\ x_4 = (-0.2929, 0.2929)^T.$$

which is the same as the solution given in [78].

Example 13. *We consider the example 5.1 in [78]. Randomly generate 500 sensors* $x_1^*, \cdots,$ x_{500}^* *from the unit square* $[-0.5, 0.5] \times [-0.5, 0.5]$. *The edge set* A *is chosen as follows. Initially set* $A = \emptyset$. *Then for each* i *from 1 to 500, compute the set* $I_i = \{j \in [500] : \|x_i^* - x_j^*\|_2 \le 0.3, j \ge i\}$*; if* $|I_i| \ge 10$*, let* A_i *be the subset of* I_i *consisting of the 10 smallest integers; otherwise, let* $A_i = I_i$*; the let* $A = A \bigcup \{(i, j) : j \in A_i\}$. *The edge set* B *is chosen such that* $B = \{(i, k) \in [n] \times [m] : \|x_i^* - a_k\|_2 \le 0.3\}$*, i.e., every anchor is connected to all the sensors that are within distance 0.3. For every* $(i, j) \in A$ *and* $(i, k) \in B$*, let the distances be*

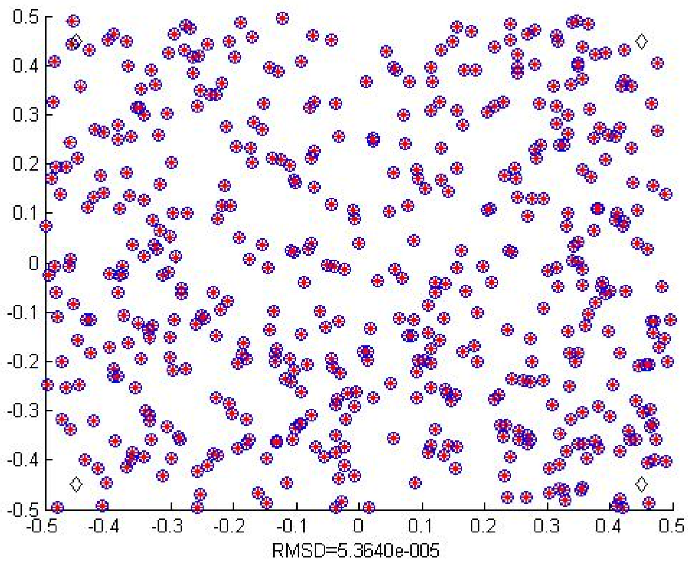
$$d_{ij} = \|x_i^* - x_j^*\|_2,\ e_{ij} = \|x_i^* - a_k\|_2.$$

Four anchors are placed at the positions $(\pm 0.45, \pm 0.45)$. *There are no errors in the distances. The computed results obtained by* Algorithm *7 are plotted in Fig. 6.1. The true* *sensor locations (denoted by circles) and the computed locations (denoted by stars) are connected by solid lines. The computed locations are denoted by* $\hat{x}_1, \cdots, \hat{x}_{500}$. *The accuracy of the computed locations is measured by the Root Mean Square Distance (*$RMSD$*) which is defined as*

$$RMSD = (\frac{1}{n}\sum_{i=1}^{n} \|\hat{x}_i - x_i^*\|_2^2)^{\frac{1}{2}}$$

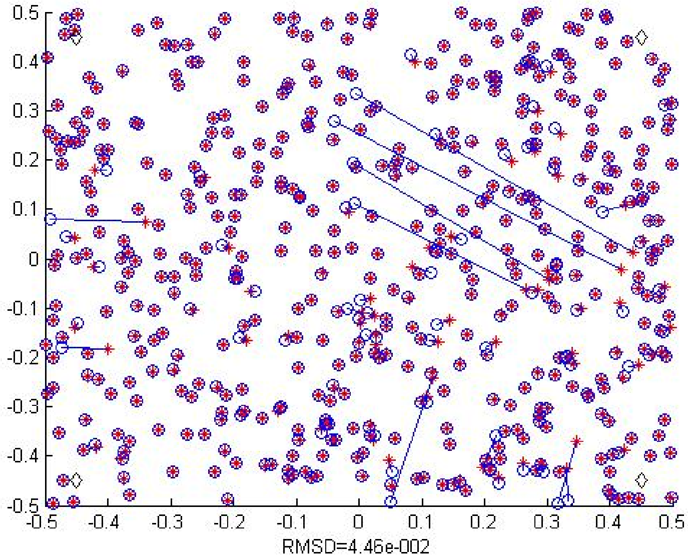
and $RMSD$ *is* $2.9e-006$ *in [78]. By our method,* $RMSD$ *is* $5.3640e-005$.

Figure 6.1.: 500 sensors, sufficient edges



Example 14. *We consider the example 5.2 in [78]. We generate random test problems almost in the same way as in the Example 10, except the following: if* $|I_i| \geq 3$, *let* A_i *be the subset of* I_i *consisting of the 3 smallest integers; otherwise, let* $A_i = I_i$. *Then the number of edges might not be sufficient to determine the sensor locations. Assume there are no distance errors. The computed results obtained by* Algorithm *7 are plotted in Fig. 6.2. The true sensor locations (denoted by circles) and the computed locations (denoted by stars) are connected by solid lines.* $RMSD$ *is* $1.1e-002$ *in [78]. By our method,* $RMSD$ *is* $4.46e-002$.

Figure 6.2.: 500 sensors, insufficient edges



6.1.3. Conclusion

In this section, we applied our local and global methods for solving quartic programming problems to sensor network localization problems. The results of numerical examples show that we can solve these kinds of large scale problems successfully. However, we must admit these methods to solve such large scale problems, say sensor network localization problems with more than 500 sensors, are time-consuming. Hence, we do not recommend to use these methods to solve such large scale problems. Since these methods are not designed for solving very large scale problems, especially, the auxiliary function we applied is not suitable to solve very large scale problems, practical methods to solve very large scale problems are our further study.

6.2. Systems of polynomial equations (SPE)

6.2.1. Introduction

Solving a system of polynomial equations is a classical and fundamental problem in many fields of science and engineering [34, 101, 123].

The general formulation of these problems is given below.

$$
\begin{aligned}
(SPE) \qquad & h_i(x) = 0,\ i = 1, 2, \cdots, m \\
& x \in X
\end{aligned}
$$

$h_i(x)$, $i = 1, 2, \cdots, m$ are polynomial equations and X is a box.

This problem (SPE) is NP-hard even if all the equations are quadratic [101]. Current methods to solve the problem (SPE) can be mainly classified into symbolic and numeric [34]. Symbolic methods based on resultants and Grobner bases [51, 59] order the monomials and eliminate variables, thereby reducing the problem to finding the roots of univariate polynomials. However these methods are efficient only for no more than three or four polynomials [34]. Numeric methods are based on either iterative or homotopy methods [123]. However these methods either depend on a good initial guess for each solution or are computationally demanding, which limits the practical applications of these methods [34]. Numeric methods based on interval arithmetic [10] have slow convergence [34].

The problem (SPE) can be transformed into an optimization problem of the form:

$$
\begin{aligned}
(OPSPE) \quad \min \quad & f(x) := \frac{1}{2}\sum_{i=1}^{m} h_i^2(x) \\
s.t. \quad & x \in X
\end{aligned}
$$

We can use our optimization methods: *Algorithm* 9 (SLOM) and *Algorithm* 10 (GOM) provided in chapter 4 to solve this problem.

6.2.2. Optimization methods for (SPE)

Actually, the problem (OPSPE) has a particular property which is that x^* is a global minimizer of the problem (OPSPE) if and only if $f(x^*) = 0$. So we can use it as a termination condition in our *Algorithm* 9 (SLOM) and *Algorithm* 10 (GOM).
The following strongly or $\varepsilon-$strongly local optimization method is designed for the problem (OPSPE).

Algorithm 13. *Strongly or $\varepsilon-$strongly local optimization method for $(OPSPE)$:(SLOM).*

Step 0. *Take an initial point $x_0 \in X$. Let $Q_1 = I$, $Q_2, \cdots, Q_d, \cdots, Q_N$ be any invertible matrices given randomly, where I is the identity matrix. Let ε be a small positive number. Let $d := 1$, $Q := Q_d$ and $i := 1$. Let $x^* = (x_1^*, \cdots, x_n^*)^T$ be a local minimizer or KKT point of $f(x)$ on $\prod_{i=1}^{n}[u_i, v_i]$ starting from x_0. Let $\bar{x} := x^*$, and go to Step 1.*

Step 1. *If $f(\bar{x}) \leq \varepsilon$, then stop and $\bar{x}$ is a global minimizer of the problem (OPSPE); otherwise, go to Step 2.*

Step 2. *Let $p := G_i(y_i)$, $a := l_i$ and $b := r_i$. Check whether the condition $[NC]_i$ holds: $p(l_i) > 0$ and the following equations hold:*

$$V_{p^{2k}}(a) - V_{p^{2k}}(b) = V_{p^{2k+1}}(a) - V_{p^{2k+1}}(b),$$
$$k = 0, 1, 2, \cdots, [\frac{K_i - 1}{2}]$$

by using the Algorithm 8. If this condition holds, go to Step 4; otherwise, go to Step 3.

Step 3. *Let $\bar{y} = (\bar{y}_1, \ldots, \bar{y}_i, \ldots, \bar{y}_n)^T = Q^{-1}\bar{x}$ and $y = (\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T$. Let $\bar{y}_i^* := argmin\{f(Qy)|y \in N_i\}$, where N_i is defined by (4.5). Let $\bar{y}^* = (\bar{y}_1, \cdots, \bar{y}_{i-1}, \bar{y}_i^*, \bar{y}_{i+1} \cdots, \bar{y}_n)$ and $\bar{x}^* := Q\bar{y}^*$. Let $x^* = (x_1^*, \cdots, x_n^*)$ be a local minimizer or KKT point of*

$f(x)$ on $\prod_{i=1}^{n}[u_i, v_i]$ starting from $\bar{x}^*$. If $f(x^*) < f(\bar{x}) - \varepsilon$, let $\bar{x} := x^*$, $i := 1$, $d := 1$ and $Q := Q_d$ go to Step 1; otherwise, go to Step 4.

Step 4. *If $i := n$, go to Step 5; otherwise, let $i := i + 1$ and go to Step 2.*

Step 5. *Let $d = d + 1$. If $d > N$, go to Step 6; otherwise, let $Q := Q_d$ and $i := 1$, go to Step 2.*

Step 6. *Stop. $\bar{x}$ is a strongly or $\varepsilon-$strongly local minimizer with respect to Q_d, $d = 1, \cdots, N$.*

Since we introduced a filled function method for nonlinear system of equations in the Chapter 1, we could change the global optimization method, i.e. *Algorithm* 10 (GOM) provided in chapter 4, to a tailor-made global optimization method for the problem (OPSPE) by using the filled function defined by (1.9) in Chapter 1. Next, we describe the new global optimization method for the problem (OPSPE).

Algorithm 14. *Global optimization method for (OPSPE):(GOM).*

Step 0. *Choose a small positive number μ and a large positive number M (in the examples of next Section, we take $\mu = 10^{-10}$ and $M = 10^5$). Choose a positive integer number K and directions $e_1, \ldots, e_K$ (in the numerical examples of next Section, we just take $K = 1$ and $e_1 = (1, \ldots, 1)^T$). Choose an initial small positive number q_0 for the parameter q (in the examples of next Section, we take $q_0 = 10^{-2}$). Take an initial point $x_0 \in X$. Let $k = 0$. If $f(x_0) \leq \mu$, then let $x_k^* := x_0$ and go to Step 5; otherwise, go to Step 1.*

Step 1. *Solve the problem (OPSPE) starting from x_k by using* Algorithm *13 $(SLOM)$. Let x_k^* be the obtained strongly or $\varepsilon-$strongly local minimizer of the problem (OPSPE). If $f(x_k^*) \leq \mu$, go to Step 5; otherwise, let $q := q_0$ and $l := 1$, then go to Step 2.*

Step 2. *If $l \leq K$, let $\lambda := 1$, go to (a); otherwise go to Step 5.*
(a). Let $y_k^l := x_k^* + \lambda e_l$. If $f(y_k^l) < f(x_k^*)$, then set $x_{k+1} := y_k^l, k := k + 1$, go to Step 1; otherwise go to (b).

(b). If $f(x_k^*) \leq f(y_k^l) \leq \frac{5f(x_k^*)}{4}$, go to Step 3; otherwise set $\lambda := \frac{\lambda}{2}$, go to (a).

Step 3. *Construct the following function*

$$G_{q,x_k^*}(x) = \exp\left(-\|x - x_k^*\|^2\right) g_{\frac{f(x_k^*)}{4}}\left(f(x) - \frac{f(x_k^*)}{2}\right) + q h_{\frac{f(x_k^*)}{4}, f(x_k^*)}\left(f(x) - \frac{f(x_k^*)}{2}\right), \quad (6.1)$$

where $g_r(t)$ and $h_{r,c}(t)$ are defined by (1.10) and (1.11), respectively. Find a local minimizer of the following problem (6.2) starting from y_k^l:

$$\min_{x \in X} \quad G_{q,x_k^*}(x), \quad (6.2)$$

Let $\bar{y}_k^l$ be a local minimizer of the problem (6.2). If $f(\bar{y}_k^l) < f(x_k^*)$, then let $x_{k+1} := \bar{y}_k^l$ and $k := k+1$, go to Step 1. Otherwise, let $q := 10q$, go to Step 4.

Step 4. *If $q \leq M$, go to Step 3; otherwise, let $q := q_0$ and $l := l+1$, go to Step 2.*

Step 5. *Let $\bar{k} := k$ and $\bar{x} := x_k^*$ and stop.*

6.2.3. Numerical examples

In this section, we try to solve all problems of polynomial equations (Test problem 1, 2, 5 and 6) presented in reference [18] by the optimization methods mentioned in last section. For the detailed information of these problems, see the appendix in the end.

Table 6.1 records the numerical results.

Table 6.1.: Results of algorithms SLOM and GOM for (SPE)

Problem number	statistic	SLOM	GOM
$EQ6.1$	suc.rate	$30/30$	$30/30$
	best	$1.0581e-013$	$1.0581e-013$

continue goes here...

Problem number	statistic	SLOM	GOM
	worst	$9.4318e-012$	$9.4318e-012$
	mean	$3.1337e-012$	$3.1337e-012$
	median	$2.6314e-012$	$2.6314e-012$
	st.dev	$3.2761e-012$	$3.2761e-012$
$EQ6.2$	suc.rate	$30/30$	$30/30$
	best	$4.0732e-015$	$4.0732e-015$
	worst	$3.0047e-014$	$3.0047e-014$
	mean	$1.0685e-014$	$1.0685e-014$
	median	$1.0302e-014$	$1.0302e-014$
	st.dev	$3.8452e-015$	$3.8452e-015$
$EQ6.3$	suc.rate	$30/30$	$30/30$
	best	$9.1023e-012$	$9.1023e-012$
	worst	$4.3756e-007$	$9.9981e-011$
	mean	$1.5583e-008$	$4.5824e-011$
	median	$4.7720e-010$	$4.6579e-011$
	st.dev	$7.9755e-008$	$3.2965e-011$
$EQ6.4$	suc.rate	$30/30$	$30/30$
	best	$2.7859e-014$	$2.7859e-014$
	worst	$1.6228e-009$	$9.9478e-011$
	mean	$5.5002e-010$	$1.2268e-011$
	median	$1.1426e-011$	$1.0949e-011$
	st.dev	$6.7377e-010$	$2.0231e-011$

6.2.4. Conclusion

In this section, we designed a tailor-made strongly or $\varepsilon-$strongly local optimization method and a global optimization method for the problem (SPE) by using the particular property of the problem (SPE) and a new auxiliary function defined by (1.9) in Chapter 1. The results of numerical examples illustrate that the methods presented in this section are efficient and stable.

6.3. Optimality condition and optimization methods for nonlinear programming problems (NLP)

The nonlinear programming problem (NLP) which appears in applied mathematical, physical, chemical, biological, environmental, engineering and economic studies is considered in this section. First, an optimality condition for the problem (NLP) is given by using linear transportations and Lagrange interpolating polynomial. Based on this condition, we design two new local optimization methods. The points obtained by the new local optimization methods may generally improve some KKT points. Finally, two global optimization methods are designed by combining the new local optimization methods and an auxiliary function. Numerical examples show that our methods are efficient and stable.

6.3.1. Introduction

Consider the following nonlinear optimization problem with box constraints:

$$
\begin{aligned}
(NLP) \qquad & \min \quad f(x) \\
& s.t. \quad x_i \in \prod_{i=1}^{n} [u_i, v_i],
\end{aligned}
\tag{6.3}
$$

where $f(x) \in C^r$, r is a given positive integer number, C^r is the set of r times continuously differential functions, $u_i < v_i, i = 1, \ldots, n$. Throughout of this chapter, we let $X := \{x = (x_1, \ldots, x_n)^T \mid x_i \in [u_i, v_i], i = 1, \ldots, n\}$.

Needless to say, a large number of real problems can be formulated as nonlinear programming problems, including in the following areas: optimal control, structural design, mechanical design, electrical networks, water resources management, stochastic resource allocation and location of facilities. See the survey book [100] and references therein. For global optimization, a great deal of attention has been focused on two areas: one is global optimization methods to solve these problems; the other is global optimality conditions. For solving this problem, many methods have been put forward and many algorithms have been designed, including exact methods (adaptive stochastic search methods [2, 138], bayesian search algorithms [75, 98], branch and bound algorithms [53, 85], enumerative strategies [113], homotopy and trajectory methods [42, 55], integral methods [74, 109], 'naive'(passive) approaches [2, 71] and relaxation (out approximation) strategies [52, 113]) and heuristic methods (approximate convex underestimation [84], continuation methods [73], genetic algorithms, evolution strategies [56, 72], 'globalized' extensions of local search methods [2, 71], sequential improvement of local optima [13, 149], simulated annealing [12, 56], and tabu search (TS) [41, 56]). For more details in the idea and applications, see [105].

For optimality conditions of nonlinear programming problems, most literature focuses on special models, such as generalized convex programming problems [60, 115, 121], nonconvex problems involving directionally differentiable functions [114], quadratic programming problems [45], cubic programming problems [144] and quatic programming problems [150]. Since KKT optimality conditions are also sufficient for optimality if the functions involved in the mathematical programming problem are convex, generalized convex functions received more attention later [60]. Researchers tried to solve this question: under what assumptions, are the KKT conditions also sufficient for the various generalizations of convex problems? [115] defined semilocally quasiconvex and semilocally pseudoconvex functions and obtained sufficient optimality conditions for a class of nonlinear programming problems involving such functions. [60] considered a nonlinear programming problem where the functions involved are $\eta-$semidifferentiable and presented KKT necessary optimality conditions and sufficient optimality conditions. [121] introduced a new class nonconvex functions called G-invex functions and provided some necessary conditions and sufficient conditions. [114] studied optimality conditions for nonconvex problems involving a class of directionally differentiable functions and generalized the necessary and sufficient optimality conditions by using the weak subgradient notion. Instead of local optimality conditions, [45], [144] and [150] tried to provide global optimality conditions for some polynomial programming problems. [45] proposed a necessary global optimality condition and a sufficient global optimality condition for mixed integer quadratic programming problems (MIQP). [144] (see Chapter 2) and [150] (see Chapter 3) provided necessary global optimality conditions for cubic polynomial optimization problems with mixed variables (MCP) and quartic polynomial optimization problems with box constraints (QPOP), respectively. Then, we provide necessary global optimality conditions for general unconstrained (GP) and constrained (GPP) polynomial programming problems in Chapter 4 and Chapter 5, respectively. More generally, although [126] developed necessary global optimality conditions for nonlinear programming problems with polynomial constraints, as it mentioned, the conditions are difficult to check for general large dimensional problems since the conditions involve in solving a sequence of semi-definite programs. [127] presented global optimality conditions for polynomial optimization over box or bivalent constraints by using separable polynomial relaxations. However, We notice that it is not easy to decompose a polynomial function to the sum of a separable polynomial function and an SOS-convex polynomial function.

It is well-known that traditional local optimization methods are designed based on KKT conditions. Motivated by this, [45] designed a new local optimization method according to the presented necessary global optimality condition for (MIQP) and also designed a global optimization method by combining the sufficient global optimality condition, an special auxiliary function and the obtained local optimization method. Furthermore, [144] (see Chapter 2) and [150] (see Chapter 3) designed new local optimization methods according to provided necessary global optimality conditions and gave global optimization methods by combining the local methods and some auxiliary functions for the problem (MCP) and the problem (QPOP), respectively. We established strongly local optimization methods and global optimization methods for the problem (GP) and the problem (GPP) in Chapter 4 and Chapter 5, respectively. number of numerical examples are also presented to indicate the significance of the necessary global optimality conditions and show the efficiency of the optimization methods. Particularly, we want to mention that the new local optimization methods produce impressive results.

In this chapter, we try to extend the same idea proposed for polynomial programming problems in Chapter 2, Chapter 3, Chapter 4 and Chapter 5 to nonlinear programming problems. We propose an optimality condition according to the following points. (i) Some specific lines can be obtained by using linear transformations. (ii) On these special directions, the objective function can be simplified into univariate nonlinear functions. (iii) we transform the univariate functions to Lagrange interpolation polynomials by using the technique proposed in [38]. (iv) we try to find a condition which is a necessary and sufficient condition to a point being global minimizers for these univariate polynomial functions along these lines. Then we design new local optimization methods by using this condition which may improve traditional local optimization methods. Finally we design global optimization methods by combining the new local optimization methods and an auxiliary function. Numerical examples illustrate the efficiency of the optimization methods proposed in the chapter.

6.3.2. Preliminary

Consider the following univariate nonlinear function:

$$g(y),\ y \in [a,b],$$

Definition 26. *[38] The unique polynomial given by:*

$$L_N(g)(y) := \sum_{k=0}^{N} g(y_k) l_{N,k}(y), \tag{6.4}$$

where

$$l_{N,k}(y) := \prod_{j \neq k} \frac{y - y_j}{y_k - y_j} \tag{6.5}$$

is called the Lagrange interpolation polynomial of degree N for function $g(y)$ with respect to $y_k, k = 0, 1, \ldots, N$.

If $g(y)$ is $N+1$ times continuously differentiable, the the interpolation error is given by the following proposition.

Proposition 4. *[38] Suppose that $g \in C^{N+1}[a,b]$ and let $L_N(y)$ be given as (6.4), then for any $y \in [a,b]$ one has*

$$g(y) - L_N(g)(y) = \prod_{k=0}^{N} (y - y_k) \frac{g^{N+1}(\zeta)}{(N+1)!}\ for\ some\ \zeta \in [a,b] \tag{6.6}$$

If $[a,b] = [-1,1]$, then it is well-known that the uniform norm of the right-hand side is minimized if we choose the y_k's as the roots of the Chebyshev polynomial (of the first kind) of degree $N+1$. Recall that the Chebyshev polynomial (of the first kind) are defined as:

$$T_j(y) := \cos(j \arccos(y))\ (j = 0, 1, \cdots). \tag{6.7}$$

The roots of T_{N+1} are therefore given by

$$y_k = \cos(\frac{(2k+1)\pi}{2(N+1)}), k = 0, 1, \ldots, N. \tag{6.8}$$

If $[a, b] \neq [-1, 1]$, the one simply does a linear transformation to obtain the Chebyshev nodes on $[a, b]$:

$$\frac{b-a}{2} y_k + \frac{a+b}{2}, k = 0, 1, \ldots, N..$$

where y_k is given in (6.8).

The Lagrange interpolation polynomial $L_N(g)(y)$ has the following properties:

Proposition 5. *[38] Assume that $L_N(g)(y)$ is the lagrange polynomial that is based on the $N+1$ Chebyshev nodes on $[a, b]$. If $g \in C^{N+1}[a, b]$, then*

$$\|g - L_N(g)\|_{\infty,[a,b]} \leq \frac{2(b-a)^{N+1}}{4^{N+1}(N+1)!} \|g^{N+1}\|_{\infty,[a,b]}. \tag{6.9}$$

where $\|g\|_{\infty,[a,b]} := \underset{x\in[a,b]}{sup} |g(x)|$.

Next, we will introduce the interpolation error when g only has a fixed degree of smoothness.

Definition 27. *(Lebesgue constant) [38] The Lebesgue constant at a set of nodes $\{y_0, \cdots, y_N\}$ is defined as*

$$\Lambda_N(y_0, \cdots, y_N) = \max_{y\in[a,b]} \sum_{k=0}^{N} |l_{N,k}(y)|,$$

where $l_{N,k}(y) := \prod_{j\neq k} \frac{y-y_j}{y_k-y_j}$ as before.

Lemma 3. *[38] Let $g \in C[a, b]$ and $L_N(g)$ be the Lagrange interpolating polynomial at the* *set of nodes* $y_0, \cdots, y_N$. *Then*

$$\|g - L_N(g)\|_{\infty,[a,b]} \leq (1 + \Lambda_N(y_0, \cdots, y_N))E_N.$$

where $E_N := \inf_{p\in R[y], degree(p)\leq N} \|f - p\|_{\infty,[a,b]}$.

Lemma 4. *If* $\{y_0, \cdots, y_N\}$ *is the set of Chebyshev nodes on the interval* $[a, b]$*, then*

$$\Lambda_N(y_0, \cdots, y_N) < \frac{2}{\pi} ln(1 + N) + 1.$$

Proof: From Lemma 2.1 in [38], we know that if $[a, b] = [-1, 1]$, then

$$\Lambda_N(y_0, \cdots, y_N) < \frac{2}{\pi} ln(1 + N) + 1.$$

Indeed, for $y \in [a, b]$, we have the same result.

Let $y = \frac{b-a}{2}x + \frac{a+b}{2}$, then $x \in [-1, 1]$ if and only if $y \in [a, b]$. So we have

$$\begin{aligned} \Lambda_N(y_0, \cdots, y_N) &= \max_{y\in[a,b]} \sum_{k=0}^{N} |l_{N,k}(y)| \\ &= \max_{x\in[-1,1]} \sum_{k=0}^{N} |l_{N,k}(x)|. \end{aligned}$$

□

Lemma 5. *If* $g \in C^r[a, b]$ *and* $N > r \geq 0$*, then*

$$E_N \leq 6^{r+1} e^r (1 + r)^{-1} \left(\frac{b-a}{2N}\right)^r \omega_r \left(\frac{b-a}{2(N-r)}\right).$$

where ω_r *is the modulus of continuity of* $g^{(r)}$ *(*$r = 0$ *corresponds to* g*):*

$$\omega_r(\delta) = \sup_{x,y\in[a,b]} (|g^{(r)}(x) - g^{(r)}(y)| : |x - y| \leq \delta).$$

Proof: From Corollary 1.4.4 in [122], we know that if $g'(y) \in C[-1,1]$, then $E_N(g;[-1,1]) \leq 6E_{N-1}(g';[-1,1])N^{-1}$. Similarly, we can prove that if $g'(y) \in C[a,b]$, then

$$E_N(g;[a,b]) \leq 6E_{N-1}(g';[a,b])\frac{b-a}{2N}.$$

By repeated application of the above inequality, we obtain

$$E_N(g;[a,b]) \leq 6^r E_{N-r}(g^{(r)};[a,b]) \left(\frac{b-a}{2}\right)^r \frac{1}{N(N-1)\cdots(N-r+1)}.$$

From Corollary 1.4.1 in [122], we know $E_{N-r}(g^{(r)};[a,b]) \leq 6\omega_r\left(\frac{b-a}{2(N-r)}\right)$. Then,

$$E_N(g;[a,b]) \leq 6^{r+1}\left(\frac{b-a}{2}\right)^r \frac{1}{N(N-1)\cdots(N-r+1)}\omega_r\left(\frac{b-a}{2(N-r)}\right).$$

From the proof of Theorem 1.5 in [122], we know $\frac{1}{N(N-1)\cdots(N-r+1)} \leq \frac{e^r}{N^r(1+r)}$. Hence,

$$E_N \leq 6^{r+1}e^r(1+r)^{-1}\left(\frac{b-a}{2N}\right)^r \omega_r\left(\frac{b-a}{2(N-r)}\right).$$

□

Using Lemma 3-5, we have the following theorem:

Theorem 14. *If $g \in C^r[a,b]$, $L_N(g)(y)$ is given as (6.4), $y_t, t = 0,1,\ldots,N$ are the roots of the Chebyshev and if $N > r \geq 0$, then the interpolation error using Chebyshev nodes satisfies:*

$$\|g - L_N(g)\|_{\infty,[a,b]} \leq 2K_r\left(\frac{b-a}{2N}\right)^r\left(\frac{1}{\pi}\ln(1+N)+1\right)\omega_r\left(\frac{b-a}{2(N-r)}\right), \quad (6.10)$$

where $K_r = 6^{r+1}e^r(1+r)^{-1}$.

Remark 21. *In order to reduce the interpolation error, we have two ways.*

1. *We can increase N. Indeed, for any given $g \in C^r[a,b]$ and $\varepsilon > 0$, under the mild*

assumption $ln(n)\omega_0(\frac{1}{n}) = o(1)$*, where* ω_0 *is the modulus of continuity of* g*, we have that there exists* $N_g > 0$ *such that* $\|g - L_N(g)\|_{\infty,[a,b]} \leq \varepsilon$ *when* $N \geq N_g$*, see [38]. But in practice,* N *cannot be very big.*

2. *We can decrease the length of interval* $[a, b]$*. We notice from Theorem 14 that the smaller* $b - a$ *is, the smaller the interpolation error is. So, we can partition the interval* $[a, b]$ *into several equally spaced subintervals and then construct the Lagrange interpolation polynomials in each small subinterval by using parallel algorithm.*

Definition 28. *[100] Consider the problem of minimizing* $f(x)$ *over feasible set* X*, and let* $\bar{x} \in X$*. Let* $B_\delta(\bar{x}) = \{x | \|x - \bar{x}\| < \delta\}$ *and* $N_\delta(\bar{x}) = B_\delta(\bar{x}) \bigcap X$*. If* $f(\bar{x}) \leq f(x)$ *for all* $x \in X$*,* $\bar{x}$ *is said to be a global minimizer of* $f(x)$ *over* X*. If there exists an* $\delta > 0$ *such that* $f(\bar{x}) \leq f(x)$ *for each* $x \in N_\delta(\bar{x})$*,* $\bar{x}$ *is said to be a local minimizer of* $f(x)$ *over* X*.*

Definition 29. *[89] Let* $\varepsilon > 0$*. We say* x^* *is an* $\varepsilon-$*global minimizer of nonconvex function* $f : R^n \to R$ *if for all* $x \in X$*,*

$$f(x) \geq f(x^*) - \varepsilon$$

.

Lemma 6. *If* $\bar{y}$ *is a global minimizer of* $L_N(g)(y)$ *on* $[a, b]$*, then there exists a number* $N_0 > 0$ *such that when* $N \geq N_0$*,* $\bar{y}$ *is an* $\varepsilon-$*global minimizer of* $g(y)$ *on* $[a, b]$*.*

Proof: If $\bar{y}$ is a global minimizer of $L_N(g)(y)$ on $[a, b]$, then we have $L_N(g)(y) \geq L_N(g)(\bar{y})$ for any $y \in [a, b]$. For any $\varepsilon > 0$, there exists a number $N_0 > 0$ such that when $N \geq N_0$, $\|g - L_N(g)\|_{\infty,[a,b]} \leq \varepsilon/2$. Hence, for any $y \in [a, b]$,

$$\begin{aligned} & g(y) - g(\bar{y}) \\ = \; & (g(y) - L_N(g)(y)) + (L_N(g)(y) - L_N(g)(\bar{y})) + (L_N(g)(\bar{y}) - g(\bar{y})) \\ \geq \; & -\varepsilon/2 - \varepsilon/2 \\ = \; & -\varepsilon \end{aligned}$$

Then $\bar{y}$ is an ε-global minimizer of $g(y)$ on $[a, b]$. $\square$

6.3.3. Optimality condition for (NLP)

In this section, we will derive an optimality condition for the problem (NLP).

Let $\bar{x} \in X$, $f \in C^r$, N be a given big number such that $N > r$, ε be a given small number. Let $\bar{x} \in X$, Q be an invertible matrix, let

$$x := Qy, \;\; g(y) := f(Qy) = f(x), \;\; \bar{y} := Q^{-1}\bar{x},$$

and let $(Q)_i$ represent the ith row of Q, $(Q)_{ij}$ represent the entry of Q in the ith row and the jth column.

Let $Y = \{y = Q^{-1}x | x \in X\}$. For $\bar{y} = (\bar{y}_1, \ldots, \bar{y}_n)^T = Q^{-1}\bar{x}$, let $y = (\bar{y}_1, \cdots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \cdots, \bar{y}_n)^T$. Let $\triangle_k = \sum\limits_{\substack{j=1 \\ j \neq i}}^{n} (Q)_{kj}\bar{y}_j = \bar{x}_k - (Q)_{ki}\bar{y}_i = \bar{x}_k - (Q)_{ki}(Q^{-1})_i\bar{x}$, $k = 1, \cdots, n$, and let

$$\begin{aligned} l_i &= \max\left\{\min\left\{\frac{u_1 - \triangle_1}{(Q)_{1i}}, \frac{v_1 - \triangle_1}{(Q)_{1i}}\right\}, \cdots, \min\left\{\frac{u_n - \triangle_n}{(Q)_{ni}}, \frac{v_n - \triangle_n}{(Q)_{ni}}\right\}\right\}, \\ r_i &= \min\left\{\max\left\{\frac{u_1 - \triangle_1}{(Q)_{1i}}, \frac{v_1 - \triangle_1}{(Q)_{1i}}\right\}, \cdots, \max\left\{\frac{u_n - \triangle_n}{(Q)_{ni}}, \frac{v_n - \triangle_n}{(Q)_{ni}}\right\}\right\}. \end{aligned}$$

Then we can obtain the following results:

$$\begin{aligned} &(1) \quad l_i \leq r_i \\ &(2) \quad [l_i, r_i] = \{y_i \mid (\bar{y}_1, \cdots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \cdots, \bar{y}_n)^T \in Y\}. \end{aligned}$$

Let

$$f_i(y_i) := f(x), \text{ where } x = Q(\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T, y_i \in [l_i, r_i], \quad (6.11)$$

$$G_i(y_i) := L_N(f_i)(y_i) - L_N(f_i)(\bar{y}_i), \quad (6.12)$$

where $L_N(f_i)$ is defined by (6.4).

Definition 30. *Let Q be an invertible matrix. For any $i = 1, \cdots, n$, let $x = Q(\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T$, $y_i \in [l_i, r_i]$ and $f_i(y_i) := f(x)$. If $f_i(y_i) \geq f_i(\bar{y}_i) - \varepsilon$, for any $i = 1, \cdots, n$, then $\bar{x}$ is called an $\varepsilon-$strongly local minimizer of the problem (NLP) with respect to Q.*

Note: For any invertible matrix Q, if $\bar{x}$ is an $\varepsilon-$global minimizer of the problem (NLP), then $\bar{x}$ is an $\varepsilon-$strongly local minimizer of the problem (NLP) with respect to Q.
We notice that $G_i(y_i)$ is a univariate polynomial. Then we present the main result of this section: the optimality condition for the problem (NLP) by recalling some properties of univariate polynomial functions presented in Chapter 4. . Let

$$\bar{G}_i(y_i) : \; = \begin{cases} G_i(y_i), & if\ G_i(l_i)G_i(r_i) \neq 0 \\ G_i(y_i)/[(y_i - l_i)^{s(i)}(r_i - y_i)^{t(i)}], & if G_i(l_i)G_i(r_i) = 0 \end{cases}, \tag{6.13}$$

where $s(i)$ and $t(i)$ are multiplicities of roots l_i and r_i respectively ($s(i) = 0$ or $t(i) = 0$ means l_i or r_i is not root). $G_i(y_i)$ is defined by (6.12).

Theorem 15. *Let $\bar{x} \in X$ and Q be any given invertible matrix. Let $\bar{y} = Q^{-1}\bar{x}$. $f \in C^r$ ($r \geq 2$), ε be a given small number. For any $i = 1, \ldots, n$, if condition $[LC]_i$ holds: $\bar{G}(l_i) > 0$ and the following equations hold:*

$$V_{\bar{G}^{2k}}(l_i) - V_{\bar{G}^{2k}}(r_i) = V_{\bar{G}^{2k+1}}(l_i) - V_{\bar{G}^{2k+1}}(r_i),\ k = 0, 1, 2, \cdots, [\frac{K_i - 1}{2}],$$

then there exists a number $N_0 > 0$ such that when $N \geq N_0$, $\bar{x}$ is an $\varepsilon-$strongly local minimizer of the problem (NLP) with respect to Q, where K_i is defined in (4.1).

Proof. By Proposition 2 in Chapter 4, For any $i = 1, \ldots, n$, $\bar{y}_i = (Q^{-1})_i\bar{x}$ is a global minimizer of $L_N(f_i)(y_i)$ on $[l_i, r_i]$ if and only if condition $[LC]_i$ holds. By Lemma 6 and Definition 30, we can easily to obtain the results. □

Remark 22. *(1) When $N \leq 4$, i.e., $N = 2, 3, 4$, the condition $[LC]_i$ presented in Theorem 15 is equivalent to*

$$[GNC]_i \quad \widetilde{\bar{x}}_i d_i \leq \min\{0, \alpha_i\}, \tag{6.14}$$

which is easy to be checkable. For the notations therein when $N = 4$, see Theorem 7 in Chapter 3. For the notations therein when $N = 3$ and $N = 2$, see Remark 7 (1) and (2), respectively in Chapter 3.

(2) When $N > 4$, to check the condition $[LC]_i$ means to check if the univariate polynomial function $L_N(f_i)(y_i) - L_N(f_i)(\bar{y}_i) \geq 0$ for any $y_i \in [l_i, r_i]$. We recall the algorithm 8 designed in Chapter 4 which can be used to check whether a univariate polynomial function $p(x) \geq 0$ for any $x \in [a, b]$.

6.3.4. Optimization methods for (NLP)

$\varepsilon-$strongly local optimization method for (NLP)

In this section, we will introduce an $\varepsilon-$strongly local optimization method for the problem (NLP) according to Theorem 15.

Algorithm 15. *$\varepsilon-$Strongly Local Optimization Method for (NLP):(SLOM)*

Step 0. *Take an initial point $x_0 \in X$. Let $Q_1 = I$, $Q_2, \cdots, Q_t, \cdots, Q_T$ be any invertible matrices given randomly, where I is the identity matrix. Let ε be a small positive number. $\bar{N}$, $\underline{N}$, M and S are fixed integers. Set $t := 1$, $i := 1$, $s = 1$ and $N := \underline{N}$. Let $\bar{x} := (x_1^*, \cdots, x_n^*)$ be a local minimizer or KKT point of $f(x)$ on $\prod_{i=1}^{n} [u_i, v_i]$ starting from x_0 and go to Step 1.*

Step 1. *Let $Q := Q_t$, $\bar{y} = Q^{-1}\bar{x} = (\bar{y}_1, \ldots, \bar{y}_i, \ldots, \bar{y}_n)^T$ and $y = (\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T$. Let $f_i(y_i) := f(Qy)$. Let $a = l_i$, $b = r_i$. If $s > S$, go to Step 8; otherwise, partition $[a, b]$ into s equally spaced subintervals. Let $[a, b]_1 = [a, a + \frac{b-a}{s}]$,*

$[a,b]_2 = [a + \frac{b-a}{s}, a + \frac{2(b-a)}{s}], \cdots, [a,b]_s = [b - \frac{b-a}{s}, b]$. *Let* $w = 1$ *and* $[a,b] := [a,b]_w$. *Go to Step 2.*

Step 2. *Let* $L_{i,N}(f_i(y_i)) := \sum_{d=0}^{N} f_i(z_d) \prod_{j \neq d} \frac{y_i - z_j}{z_d - z_j}$, *where* $z_d = \frac{b-a}{2} cos\left(\frac{(2d+1)\pi}{2(N+1)}\right) + \frac{a+b}{2}$, *for* $d = 0, \cdots, N$, *are Chebyshev nodes, go to Step 3.*

Step 3. *Let* $p := L_{i,N}(f_i)(y_i) - L_{i,N}(f_i)(\bar{y}_i)$ *and* $K = K_i$ *defined in (4.1). Check whether the condition holds:* $p(a) > 0$ *and the following equations hold:*

$$V_{p^{2k}}(a) - V_{p^{2k}}(b) = V_{p^{2k+1}}(a) - V_{p^{2k+1}}(b),$$
$$k = 0, 1, 2, \cdots, [\frac{K-1}{2}]$$

by using the Algorithm 8. If the condition holds, go to Step 5, otherwise go to Step 4.

Step 4. *Let* $\bar{y}_i^* := argmin\{L_{i,N}(f_i)(y_i) | y_i \in [a,b]\}$ *and* $\bar{y}^* = (\bar{y}_1, \cdots, \bar{y}_{i-1}, \bar{y}_i^*, \bar{y}_{i+1} \cdots, \bar{y}_n)$ *and* $\bar{x}^* = Q\bar{y}^*$. *Let* $x^* = (x_1^*, \cdots, x_n^*)$ *be a local minimizer or* KKT *point of* $f(x)$ *on* $\prod_{i=1}^{n} [u_i, v_i]$ *starting from* $\bar{x}^*$. *If* $f(x^*) < f(\bar{x}) - \varepsilon$, *let* $\bar{x} := x^*$, $i := 1$, $N := \underline{N}$, $s := 1$ *and* $t := 1$, *go to Step 1, otherwise, go to Step 5.*

Step 5. *Let* $w := w + 1$. *If* $w > s$, *let* $i := i + 1$ *and go to Step 6, otherwise let* $[a,b] := [a,b]_w$ *and go to Step 2.*

Step 6. *If* $i \leq n$, *go to Step 1; else go to Step 7.*

Step 7. *Let* $N := N + M$. *If* $N > \bar{N}$, *go to Step 8; otherwise, let* $i := 1$ *and go to Step 1.*

Step 8. *Let* $s := 2s$. *If* $s > S$, *go to Step 9; otherwise, let* $i := 1$, $N := \underline{N}$ *and go to Step 1.*

Step 9. *Let* $t := t + 1$. *If* $t > T$, *go to Step 10; otherwise, let* $Q := Q_t$, $i := 1$, $N := \underline{N}$ *and* $s := 1$, *go to Step 1.*

Step 10. *Stop.* $\bar{x}$ *is an* $\varepsilon-$*strongly local minimizer with respect to all the chosen* Q_t, $t = 1, \cdots, T$.

Note that, by Theorem 14, for a given $\varepsilon > 0$, when N and S are large enough, for any $i = 1, \dots, n$ and $Q = Q_t, t = 1, \dots, T$,

$$|f_i(y_i) - L_{i,N}(f_i)(y_i)| \le \frac{\varepsilon}{2} \text{ for any } y_i \in [a,b]_\omega, \omega = 1, \dots, S. \tag{6.15}$$

Theorem 16. *For a given $\varepsilon > 0$, suppose that N and S are large enough, such that (6.15) is true. For a given initial point $x_0 \in X$, we can obtain an $\varepsilon-$strongly local minimizer $\bar{x}$ of the problem (NLP) in finite iteration times by the given strongly local optimization method* **SLOM**.

Proof: First, we can prove that this algorithm must stop in finite iteration times.
Let $W := max\{f(x) \mid x \in X\}$ and $m := min\{f(x) \mid x \in X\}$. For the given Q_t, given $[a,b]_\omega$ and given N, there are at most $n[\frac{W-m}{\varepsilon}]$ iteration times from *step 1* to *step 5*. In fact, for the given Q_t, given $[a,b]_\omega$, given N and given i, if the condition in *step 3* holds or if $f(x^*) \ge f(\bar{x}) - \varepsilon$, then we will change the i into $i+1$; only when the condition in *step 3* does not hold and $f(x^*) < f(\bar{x}) - \varepsilon$, we will change i to 1 in *step 5* and go to *step 1*. For the same Q_t, same $[a,b]_\omega$ and same N, when we change i to 1, the objection function value will decrease at least ε. Hence, there are at most $[\frac{W-m}{\varepsilon}]$ times to change i to 1 in *step 5*, where $[a]$ is the largest integer number which is less and equal to a. The total iteration time from *step 1* to *step 5* is at most $n[\frac{W-m}{\varepsilon}]$. For the given Q_t and given $[a,b]_\omega$, since we have $N = [\frac{\bar{N}-N}{M}]+1$ numbers Lagrange interpolation polynomials, the total iteration time from *step 1* to *step 6* is at most $([\frac{\bar{N}-N}{M}]+1)n[\frac{W-m}{\varepsilon}]$. Let $\tilde{S} = \{1, 2, 4, \cdots, S\}$. For the given Q_t and given $s \in \tilde{S}$, we have s intervals, and the total intervals for all $s \in \tilde{S}$ are $\sum\limits_{s \in \tilde{S}} s$. Hence, for the given Q_t, the total iteration time from *step 1* to *step 7* is at most $(\sum\limits_{s \in \tilde{S}} s)([\frac{\bar{N}-N}{M}]+1)n[\frac{W-m}{\varepsilon}]$. Since we have T numbers of Q_t , this algorithm must stop at most $T(\sum\limits_{s \in \tilde{S}} s)([\frac{\bar{N}-N}{M}]+1)n[\frac{W-m}{\varepsilon}]$ iteration times.
Second, we can prove that we can obtain an $\varepsilon-$strongly local minimizer in finite iteration

times. Since this algorithm must stop in finite steps, we will stop at point $\bar{x}$, such that

(i) for any $i = 1, \ldots, n$, $\bar{y}_i = (Q^{-1})_i \bar{x}$ satisfies the condition $[LC]_i$, then $\bar{x}$ is an $\varepsilon-$strongly local minimizer of the problem (NLP) since condition $[LC]_i$ implies that $L_{i,N}(f_i)(y_i) \geq L_{i,N}(f_i)\ (\bar{y}_i)$ for any $y_i \in [l_i, r_i]$ and since $|f_i(y_i) - L_{i,N}(f_i)(y_i)| \leq \frac{\varepsilon}{2}$ for any $y_i \in [l_i, r_i]$, which implies that $f_i(\bar{y}_i) \leq L_{i,N}(f_i)(\bar{y}_i) + \frac{\varepsilon}{2} \leq L_{i,N}(f_i)(y_i) + \frac{\varepsilon}{2} \leq f_i(y_i) + \varepsilon$ for any $y_i \in [l_i, r_i]$.

(ii) for any $i = 1, \cdots, n$, either $\bar{y}_i = (Q^{-1})_i \bar{x}$ satisfies the condition $[LC]_i$ which implies that $f_i(\bar{y}_i) \leq f_i(y_i) + \varepsilon$ for any $y_i \in [l_i, r_i]$; or there exists $y_i^* \in [l_i, r_i]$, such that y_i^* satisfies the condition $[LC]_i$ at y_i^* which implies that $L_{i,N}(f_i)(y_i) \geq L_{i,N}(f_i)(y_i^*)$ for any $y_i \in [l_i, r_i]$, and $f_i(\bar{y}_i) - f_i(y_i^*) \leq \varepsilon$. Hence, $f_i(\bar{y}_i) \leq f_i(y_i^*) + \varepsilon \leq L_{i,N}(f_i)(y_i^*) + \frac{3\varepsilon}{2} \leq L_{i,N}(f_i)(y_i) + \frac{3\varepsilon}{2} \leq f_i(y_i) + 2\varepsilon$ for any $y_i \in [l_i, r_i]$. Therefore, $\bar{x}$ is a $2\varepsilon-$ strongly local minimizer of the problem (NLP).

$\square$

Remark 23. *In Algorithm* **SLOM***, from step 2 to step 5, we can also apply Parallel Algorithm to check the necessary global optimality condition and calculate the global minimizer in s subintervals.*

Algorithm 16. *Parallel Algorithm*

Step 1. *Let* $[a, b] := [a, b]_w$. *Let* $L_{i,N}(f_i(y_i)) := \sum_{d=0}^{N} f_i(z_d) \prod_{j \neq d} \frac{y_i - z_j}{z_d - z_j}$, *where* $z_d = \frac{b-a}{2} \cos\left(\frac{(2d+1)\pi}{2(N+1)}\right) + \frac{a+b}{2}$, *for* $d = 0, \cdots, N$, *are Chebyshev nodes, go to Step 2.*

Step 2. *Let* $p := L_{i,N}(f_i)(y_i) - L_{i,N}(f_i)(\bar{y}_i)$ *and* $K := K_i$, *where* K_i *is defined in (4.1). Check whether the condition holds:* $p(a) > 0$ *and the following equations hold:*

$$V_{p^{2k}}(a) - V_{p^{2k}}(b) = V_{p^{2k+1}}(a) - V_{p^{2k+1}}(b),$$
$$k = 0, 1, 2, \cdots, [\frac{K-1}{2}]$$

by using the Algorithm 8 in Chapter 4. If the condition holds, go to Step 3, otherwise, go to

Step 4.

Step 3. *let* $\bar{y}^*_{i,w} := \bar{y}_i$, $\bar{y}^*_w = (\bar{y}_1, \cdots, \bar{y}_{i-1}, \bar{y}^*_{i,w}, \bar{y}_{i+1} \cdots, \bar{y}_n)$ *and* $\bar{x}^*_w = Q\bar{y}^*_w$. *Then stop.*

Step 4. *Let* $\bar{y}^*_{i,w} := argmin\{L_{i,N}(f_i)(y_i)|y_i \in [a, b]\}$, $\bar{y}^*_w = (\bar{y}_1, \cdots, \bar{y}_{i-1}, \bar{y}^*_{i,w}, \bar{y}_{i+1} \cdots, \bar{y}_n)$ *and* $\bar{x}^*_w = Q\bar{y}^*_w$. *Then stop.*

Algorithm 17. *Applying Parallel Algorithm*

Step 0. *Take an initial point* $x_0 \in X$. *Let* $Q_1 = I$, Q_2, $\cdots$, Q_t, $\cdots$, Q_T *be any invertible matrices given randomly, where* I *is the identity matrix. Let* ε *be a small positive number.* $\bar{N}$, $\underline{N}$, M *and* S *are fixed integers. Set* $t := 1$, $Q := Q_t$, $i := 1$, $w = 1$, $s = 1$ *and* $N := \underline{N}$. *Let* $\bar{x} := (x^*_1, \cdots, x^*_n)$ *be a local minimizer or* KKT *point of* $f(x)$ *on* $\prod_{i=1}^{n}[u_i, v_i]$ *starting from* x_0 *and go to Step 1.*

Step 1. *Let* $\bar{y} = Q^{-1}\bar{x} = (\bar{y}_1, \ldots, \bar{y}_i, \ldots, \bar{y}_n)^T$ *and* $y = (\bar{y}_1, \ldots, \bar{y}_{i-1}, y_i, \bar{y}_{i+1}, \ldots, \bar{y}_n)^T$. *Let* $f_i(y_i) := f(Qy)$. *Let* $a = l_i$, $b = r_i$. *Partition* $[a, b]$ *into* s *equally spaced subintervals. Let* $[a, b]_1 = [a, a + \frac{b-a}{s}]$, $[a, b]_2 = [a + \frac{b-a}{s}, a + \frac{2(b-a)}{s}]$, $\cdots$, $[a, b]_s = [b - \frac{b-a}{s}, b]$. *Go to Step 2.*

Step 2. *Call Parallel Algorithm.*

Step 3. *Let* $\bar{x}^* := argmin\{f(\bar{x}^*_w)|w = 1, \cdots, s\}$. *Let* $x^* = (x^*_1, \cdots, x^*_n)^T$ *be a local minimizer or* KKT *point of* $f(x)$ *on* $\prod_{i=1}^{n}[u_i, v_i]$ *starting from* $\bar{x}^*$. *If* $f(x^*) < f(\bar{x}) - \varepsilon$, *let* $\bar{x} := x^*$, $i := 1$, $N := \underline{N}$, $s := 1$ *and* $t := 1$, *go to Step 1. Otherwise, if* $i := n$, *go to Step 4, else let* $i := i + 1$ *and go to Step 1.*

Step 4. *Let* $N := N + M$. *If* $N > \bar{N}$, *go to Step 5; otherwise, let* $i := 1$ *and go to Step 1.*

Step 5. *Let* $s := 2s$. *If* $s > S$, *go to Step 6; otherwise, let* $i := 1$, $N := \underline{N}$ *and go to Step 1.*

Step 6. *Let* $t := t + 1$. *If* $t > T$, *go to Step 7; otherwise, let* $Q := Q_t$, $i := 1$, $N := \underline{N}$ *and* $s := 1$, *go to Step 1.*

Step 7. *Stop.* $\bar{x}$ *is an* $\varepsilon-$*strongly local minimizer with respect to all the chosen* Q_t, $t = 1, \cdots, T$.

Remark 24. *In Algorithm 15, in the Step 3, we need to check the following optimality condition:*

$p(a) > 0$ *and the following equations hold:*

$$V_{p^{2k}}(a) - V_{p^{2k}}(b) = V_{p^{2k+1}}(a) - V_{p^{2k+1}}(b), \\ k = 0, 1, 2, \cdots, [\frac{K-1}{2}].$$

From Remark 22, we know that when $N = 2, 3, 4$, respectively, the condition above is equivalent to the condition $[GNC]_i$ defined in (6.14) which is easy to check.

If the optimality condition is not satisfied, then in Step 4, we need to calculate

$$\bar{y}_i^* := argmin\{L_{i,N}(f_i)(y_i)|y_i \in [a, b]\}. \tag{6.16}$$

From (6.16), we can see $\bar{y}_i^*$ is a global minimizer of the univariate polynomial function $L_{i,N}(f_i)(y_i)$ over $[a, b]$. Actually, it is not necessarily to obtain a global minimizer here. We just want a point which can improve the current local minimizer $\bar{y}_i$ for the function $L_{i,N}(f_i)(y_i)$, $\forall y_i \in [a, b]$. When $N = 2, 3, 4$, the literature [45], Chapter 2 and Chapter 3 did just this procedure.

When $N = 2, 3, 4$, $\bar{y}_i^*$ is calculated according to the following formulas to improve $\bar{y}_i$:

When $N = 2$,

$$\bar{y}_i^* := argmin\{L_{i,N}(f_i)(y_i)|y_i \in \{a, b\}\}. \tag{6.17}$$

When $N = 3$,

$$\bar{y}_i^* := argmin\{L_{i,N}(f_i)(y_i)|y_i \in \{a, b\} \bigcup Z_i\}. \tag{6.18}$$

where $Z_i = \{y_{i,\bar{x}}\} \bigcap (a, b)$ and $y_{i,\bar{x}}$ is defined by (2.8) in Chapter 2.

When $N = 4$,

$$\bar{y}_i^* := argmin\{L_{i,N}(f_i)(y_i) | y_i \in \{a, b\} \bigcup Z_i\}. \tag{6.19}$$

where $Z_i = P_i \bigcap (a, b)$ and P_i is defined by (3.12) in Chapter 3.

We need to notice the $\bar{y}_i^*$ in (6.17)-(6.19), respectively, is not a global minimizer of $L_{i,N}(f_i)(y_i)$ over $[a, b]$, but we know that $\bar{y}_i^*$ can improve the current local minimizer $\bar{y}_i$ through the analysis in [45], Chapter 2 and Chapter 3. $\bar{y}_i^*$ in (6.17)-(6.19) , respectively, is easy to calculate.

When $N > 4$, please refer to the methods mentioned by Remark 16 in Chapter 4.

Remark 25. *From Remark 21, we know if N is big enough or $b-a$ is small enough, then there must exist a polynomial which is arbitrarily close to original nonlinear function. However, in practice, N is not necessary to be very big and $b - a$ is not necessary to be very small.*

The Weierstrass Theorem states that if the function $f(x)$ is continuous on $[a, b]$ and $\varepsilon > 0$, then there exists a polynomial $p(x)$ such that $\|f(x) - p(x)\| < \varepsilon$, where $\|\cdot\|$ is the uniform norm over the interval $[a, b]$, that is, where $\|g\| := \max_{a \le x \le b} |g(x)|$ [122]. This mean 'a polynomial which is a good fit always exists, but how do we find it, and just how big does N have to be? For practical reasons, however, it is often better to constraint N to remain small (or modest)' [24].

*From the analysis in Remark 24, we know when $N = 2, 3, 4$, respectively, the optimality condition in Step 3 is easy to check and $\bar{y}_i^*$ in Step 4 is easy to calculate in the Algorithm 15 (*__SLOM__*). Hence, we have the following two algorithms in which we mainly keep $N = 2, 3, 4$.*

In Algorithm SLOM, if we take the values $\underline{N} = 2$, $\bar{N} = 4$, $M = 1$ and $S = 4$, then the algorithm is denoted as Algorithm SLOM1; if we take the values $\underline{N} = 2$, $\bar{N} = 10$, $N = 2, 3, 4, 10$, M is not fixed here, and $S = 1$, then the algorithm is denoted as Algorithm SLOM2.

In SLOM1, we take the values $N = 2, 3, 4$ first. If the results are not good enough, we will *partition interval into two, even further into four equally spaced subintervals in order to reduce the interpolation error. We can apply parallel algorithm to achieve this.*

In SLOM2, we also take the values $N = 2, 3, 4$ first. If the results are not good enough, we will increase the degree of Lagrange interpolation polynomial to $N = 10$.

The numerical results in section 6.3.5 show that both SLOM1 and SLOM2 are efficient.

Although the numerical results in section 6.3.5 show that both SLOM1 and SLOM2 are efficient, they are still $\varepsilon-$strongly local optimization methods. It is necessary to design a global optimization method in the next section.

Global optimization method for (NLP)

In this section, we will design a global optimization method for the problem (NLP) by combining the $\varepsilon-$strongly local optimization method and an auxiliary function. In this chapter, we still use the auxiliary function which was presented by (1.2) in Chapter 1. For the properties of this auxiliary function, see Chapter 1.

Algorithm 18. *Global optimization method for (NLP):(GOM)*

Step 0. *Set $M := 10^{10}$, $\mu := 10^{-10}$ and $k_0 := 2n$. Set $A_{n\times n} := I_{n\times n}$ and $B_{n\times 2n} := [A, -A]$. Let $r_0 := 1$, $c_0 := 1$, $q_0 := 10^5$ and $\delta_0 := \frac{1}{2}$. Let $k := 1$, $i := 1$ and $r := r_0$. Let x_1^0 be an initial point and $x_0^* := x_1^0$, then go to Step 1.*

Step 1. *Use the $\varepsilon-$strongly local optimization method $(SLOM)$ to solve the problem (NLP) starting from x_k^0. Let x_k^* be the obtained $\varepsilon-$strongly local minimizer of the problem (NLP). If $f(x_k^*) \geq f(x_0^*)$, then go to Step 5; otherwise let $q := q_0$, $c := c_0$, $r := r_0$, $\delta := \delta_0$, $i := 1$ and $x_0^* = x_k^*$, $k := k+1$, then go to Step 2.*

Step 2. *Let B_i indicate the ith column of B and $\bar{x}_k^* := x_0^* + \delta B_i$. If $\bar{x}_k^* \notin S$, go to Step 3. Otherwise, if $f(\bar{x}_k^*) < f(x_0^*)$, then set $x_{k+1}^0 = \bar{x}_k^*$ and $x_0^* := \bar{x}_k^*$, $k := k+1$ and go to Step 1; otherwise go to Step 4.*

Step 3. *If* $\delta < \mu$*, go to Step 8; otherwise, let* $\delta = \frac{\delta}{2}$ *and go to Step 2.*

Step 4. *If* $f(x_0^*) \leq f(\bar{x}_k^*) \leq f(x_0^*) + 1$*, then go to Step 5; otherwise let* $\delta = \frac{\delta}{2}$ *go to Step 2.*

Step 5. *Let*

$$F_{q,r,c,x_0^*}(x) = q\left(exp(-\frac{\|x - x_0^*\|^2}{q})g_{r,c}\Big(f(x) - f(x_0^*)\Big) + h_{r,c}\Big(f(x) - f(x_0^*)\Big)\right).$$

Solve the problem:

$$\begin{aligned} \min \quad & F_{q,r,c,x_0^*}(x) \\ s.t. \quad & x \in X. \end{aligned} \tag{6.20}$$

by a local search method starting from the initial point $\bar{x}_k^*$*. Let* $\bar{x}_{q,r,c,x_k^*}$ *be the local minimizer obtained. Then set* $x_{k+1}^0 = \bar{x}_{q,r,c,x_k^*}$*,* $k := k+1$ *and go to Step 1.*

Step 6. *If* $q < M$*, then increase* q *(in the following examples, let* $q := 10q$*), then go to Step 5; otherwise go to Step 7.*

Step 7. *If* $c < M$*, then increase* c *(in the following examples, let* $c := 10c$*), and let* $q := q_0$*, then go to Step 5; otherwise go to Step 8.*

Step 8. *If* $i < k_0$*, then let* $i := i+1$*,* $q := q_0$*,* $c := c_0$*,* $\delta = \delta_0$*, go to Step 2; otherwise go to Step 9.*

Step 9. *If* $r > \mu$*, then decrease* r *(in the following examples, let* $r := \frac{r}{10}$*). Randomly select an orthogonal matrix* $A_{n\times n}$ *and set* $B_{n\times 2n} := [A, -A]$*. Let* $i := 1$*,* $q := q_0$*,* $c := c_0$*,* $\delta = \delta_0$ *and go to Step 2; otherwise, stop and* x_0^* *is the obtained global minimizer or approximate global minimizer of the problem* (NLP)*.*

Here, if SLOM is replaced by SLMO1 and SLMO2, then we denote the corresponding global optimization methods as GOM1 and GOM2, respectively.

6.3.5. Numerical examples

In order to test the performance of our algorithms: strongly local optimization methods (SLOM1 and SLOM2) and global optimization methods (GOM1 and GOM2), twenty common benchmark functions from [97] are selected for the experiment. Table 6.2 shows summary information of these test problems. Although we can apply parallel algorithm in the SLOM1, we did not use it. The computation was implemented on a Microsoft Windows XP Desktop of 3.46GB memory and 2.99GHz CPU frequency in our chapter.

Table 6.2.: Test problems for (NLP)

Problem number	Name and parameter values	Global minimizer x^*	Optimal value $f(x^*)$
6.1	Branin	$(9.42478,\ 2.475)$†	0.397887
6.2	Bohachevsky1	$(0,\ 0)$	0
6.3	Bohachevsky2	$(0,\ 0)$	0
6.4	Bohachevsky3	$(0,\ 0)$	0
6.5	Easom	$(\pi,\ \pi)$	-1
6.6	Michalewics(2)	$(2.2029,\ 1.5708)$	-1.8013
6.7	Shubert	$(0.0217, -0.9527)$†	-186.7309
6.8	Schwefel(2)	$(420.9687,\ 420.9687)$	0
6.9	Hartmann(3,4)	$(0.114614,\ 0.555649,\ 0.852547)$	-3.8600
6.10	Shekel(5)	$(4,\ 4,\ 4,\ 4)$	-10.1532
6.11	Shekel(10)	$(4,\ 4,\ 4,\ 4)$	-10.5364
6.12	Hartmann(6,4)	$(0.20169,\ 0.150011,\ 0.47687,$ $0.275332,\ 0.311652,\ 0.6573)$	-3.3224

continue goes here…

Problem number	Name and parameter values	Global minimizer x^*	Optimal value $f(x^*)$
6.13	Schwefel(6)	$(420.9687, \cdots, 420.9687)$	0
6.14	Michalewics(10)	$(2.2029,\ 1.5708,\ 1.2850,\ 1.9231,\ 1.7205$ $1.5708,\ 1.4544,\ 1.7561,\ 1.6557,\ 1.5708)$	-9.66015
6.15	Rastrigin(10)	$(0, \cdots, 0)$	0
6.16	Griewank(10)	$(0, \cdots, 0)$	0
6.17	Rastrigin(20)	$(0, \cdots, 0)$	0
6.18	Griewank(20)	$(0, \cdots, 0)$	0
6.19	Levy(30)	$(1, \cdots, 1)$	0
6.20	Ackley(30)	$(0, \cdots, 0)$	0

† This is one of several multiple optimal solutions.

For our experiments, we use the optimality gap mentioned in [97] is:

$$GAP = |f(x) - f(x^*)|$$

where x is a heuristic solution obtained by our method and x^* is the optimal solution. We then say that a heuristic solution x is optimal if:

$$GAP \leq \begin{cases} \varepsilon & f(x^*) = 0 \\ \varepsilon \times |f(x^*)| & f(x^*) \neq 0 \end{cases}$$

In our experimentation we set $\varepsilon = 0.001$ as the same of that in [97].

For comparison, some common statistics are included. We randomly select 30 initial points for every problem. The *suc.rate*(success rate) means the success times out of 30. The *best* is the minimum of the results, the *worst* indicates the maximum of the results, and then it follows the *mean*, *median* and *st.dev.*(standard deviation). In some way, these statistics are able to evaluate the search ability and solution accuracy, reliability and convergence as well as stability.

In the below table, we record the results of algorithms SLOMs and GOMs.

Table 6.3.: Results of algorithms SLOMs and GOMs for (NLP)

Problem	statistic	SLOM1	GOM1	SLOM2	GOM2
6.1	suc.rate	$30/30$	$30/30$	$30/30$	$30/30$
	best	0.3979	0.3979	0.3979	0.3979
	worst	0.3979	0.3979	0.3979	0.3979
	mean	0.3979	0.3979	0.3979	0.3979
	median	0.3979	0.3979	0.3979	0.3979
	st.dev	$7.8971e-014$	$7.8971e-014$	$4.7145e-014$	$4.7145e-014$
6.2	suc.rate	$30/30$	$30/30$	$30/30$	$30/30$
	best	0	0	0	0
	worst	$2.7756e-015$	$2.7756e-015$	$3.7748e-015$	$3.7748e-015$
	mean	$6.5873e-016$	$6.5873e-016$	$7.9566e-016$	$7.9566e-016$
	median	0	0	0	0
	st.dev	$8.7949e-016$	$8.7949e-016$	$1.0705e-015$	$1.0705e-015$
6.3	suc.rate	$30/30$	$30/30$	$30/30$	$30/30$
	best	0	0	0	0
	worst	$4.3854e-015$	$4.3854e-015$	$7.8826e-015$	$7.8826e-015$
	mean	$1.9725e-015$	$1.9725e-015$	$1.7967e-015$	$1.7967e-015$
	median	$2.2760e-015$	$2.2760e-015$	$1.7208e-015$	$1.7208e-015$

continue goes here...

Problem	statistic	SLOM1	GOM1	SLOM2	GOM2
	st.dev	$8.6536e-016$	$8.6536e-016$	$1.5724e-015$	$1.5724e-015$
6.4	suc.rate	$30/30$	$30/30$	$30/30$	$30/30$
	best	$5.5511e-017$	$5.5511e-017$	0	0
	worst	$4.0190e-014$	$4.0190e-014$	$5.5456e-014$	$5.5456e-014$
	mean	$2.5152e-014$	$2.5152e-014$	$1.3961e-014$	$1.3961e-014$
	median	$3.0836e-014$	$3.0836e-014$	$1.6875e-014$	$1.6875e-014$
	st.dev	$1.0495e-014$	$1.0495e-014$	$1.4312e-014$	$1.4312e-014$
6.5	suc.rate	$10/30$	$30/30$	$8/30$	$27/30$
	best	-1.0000	-1	-1.0000	-1.0000
	worst	0	-1.0000	0	0
	mean	-0.2273	-1.0000	-0.2667	-0.9000
	median	$-4.9193e-009$	-1.0000	0	-1.0000
	st.dev	0.4289	$2.4737e-014$	0.4498	0.3051
6.6	suc.rate	$30/30$	$30/30$	$30/30$	$30/30$
	best	-1.8013	-1.8013	-1.8013	-1.8013
	worst	-1.8013	-1.8013	-1.8013	-1.8013
	mean	-1.8013	-1.8013	-1.8013	-1.8013
	median	-1.8013	-1.8013	-1.8013	-1.8013
	st.dev	$2.8223e-015$	$2.8223e-015$	$4.4042e-015$	$4.4042e-015$
6.7	suc.rate	$30/30$	$30/30$	$30/30$	$30/30$
	best	-186.7309	-186.7309	-186.7309	-186.7309
	worst	-79.4109	-186.7309	-186.7309	-186.7309
	mean	-183.1536	-186.7309	-186.7309	-186.7309
	median	-186.7309	-186.7309	-186.7309	-186.7309

continue goes here…

Problem	statistic	SLOM1	GOM1	SLOM2	GOM2
	st.dev	$1.4530e-012$	$1.4530e-012$	$1.4427e-012$	$1.4427e-012$
6.8	suc.rate	$30/30$	$30/30$	$30/30$	$30/30$
	best	$2.5455e-005$	$2.5455e-005$	$2.5455e-005$	$2.5455e-005$
	worst	$2.5455e-005$	$2.5455e-005$	$2.5455e-005$	$2.5455e-005$
	mean	$2.5455e-005$	$2.5455e-005$	$2.5455e-005$	$2.5455e-005$
	median	$2.5455e-005$	$2.5455e-005$	$2.5455e-005$	$2.5455e-005$
	st.dev	$1.8569e-013$	$1.8569e-013$	$7.7183e-014$	$7.7183e-014$
6.9	suc.rate	$30/30$	$30/30$	$30/30$	$30/30$
	best	-3.8600	-3.8600	-3.8600	-3.8600
	worst	-3.8600	-3.8600	-3.8600	-3.8600
	mean	-3.8600	-3.8600	-3.8600	-3.8600
	median	-3.8600	-3.8600	-3.8600	-3.8600
	st.dev	$1.7965e-012$	$1.7965e-012$	$2.0214e-012$	$2.0214e-012$
6.10	suc.rate	$30/30$	$30/30$	$30/30$	$30/30$
	best	-10.1532	-10.1532	-10.1532	-10.1532
	worst	-10.1532	-10.1532	-10.1532	-10.1532
	mean	-10.1532	-10.1532	-10.1532	-10.1532
	median	-10.1532	-10.1532	-10.1532	-10.1532
	st.dev	$1.3087e-013$	$1.3087e-013$	$1.3098e-013$	$1.3098e-013$
6.11	suc.rate	$30/30$	$30/30$	$30/30$	$30/30$
	best	-10.5321	-10.5321	-10.5321	-10.5321
	worst	-10.5321	-10.5321	-10.5321	-10.5321
	mean	-10.5321	-10.3520	-10.5321	-10.5321
	median	-10.5321	-10.5321	-10.5321	-10.5321

continue goes here…

Problem	statistic	SLOM1	GOM1	SLOM2	GOM2
	st.dev	$1.2280e-013$	$1.2280e-013$	$1.2172e-013$	$1.2172e-013$
6.12	suc.rate	$30/30$	$30/30$	$30/30$	$30/30$
	best	-3.3224	-3.3224	-3.3224	-3.3224
	worst	-3.3224	-3.3224	-3.3224	-3.3224
	mean	-3.3224	-3.3224	-3.3224	-3.3224
	median	-3.3224	-3.3224	-3.3224	-3.3224
	st.dev	$5.6529e-014$	$5.6529e-014$	$6.2499e-014$	$6.2499e-014$
6.13	suc.rate	$30/30$	$30/30$	$30/30$	$30/30$
	best	$7.6365e-005$	$7.6365e-005$	$7.6365e-005$	$7.6365e-005$
	worst	$7.6365e-005$	$7.6365e-005$	$7.6365e-005$	$7.6365e-005$
	mean	$7.6365e-005$	$7.6365e-005$	$7.6365e-005$	$7.6365e-005$
	median	$7.6365e-005$	$7.6365e-005$	$7.6365e-005$	$7.6365e-005$
	st.dev	$4.0204e-013$	$4.0204e-013$	$6.1747e-013$	$6.1747e-013$
6.14	suc.rate	$14/30$	$30/30$	$10/30$	$30/30$
	best	-9.6602	-9.6602	-9.6602	-9.6602
	worst	-9.4974	-9.6602	-9.4684	-9.6602
	mean	-9.6399	-9.6602	-9.6126	-9.6602
	median	-9.6552	-9.6602	-9.6184	-9.6602
	st.dev	0.0332	$6.8798e-015$	0.0490	$2.6735e-014$
6.15	suc.rate	$30/30$	$30/30$	$30/30$	$30/30$
	best	0	0	0	0
	worst	0	0	0	0
	mean	0	0	0	0
	median	0	0	0	0

continue goes here. . .

Problem	statistic	SLOM1	GOM1	SLOM2	GOM2
	st.dev	0	0	0	0
6.16	suc.rate	30/30	30/30	30/30	30/30
	best	$5.7288e-014$	$5.7288e-014$	$2.9943e-013$	$2.9943e-013$
	worst	$9.7844e-012$	$9.7844e-012$	$8.7973e-012$	$8.7973e-012$
	mean	$2.7571e-012$	$2.7571e-012$	$3.0619e-012$	$3.0619e-012$
	median	$2.1166e-012$	$2.1166e-012$	$2.2805e-012$	$2.2805e-012$
	st.dev	$2.6216e-012$	$2.6216e-012$	$2.3935e-012$	$2.3935e-012$
6.17	suc.rate	30/30	30/30	30/30	30/30
	best	0	0	0	0
	worst	0	0	0	0
	mean	0	0	0	0
	median	0	0	0	0
	st.dev	0	0	0	0
6.18	suc.rate	30/30	30/30	30/30	30/30
	best	$1.7958e-012$	$1.7958e-012$	$1.5604e-012$	$1.5604e-012$
	worst	$1.8837e-011$	$1.8837e-011$	$1.9730e-011$	$1.9730e-011$
	mean	$6.4374e-012$	$6.4374e-012$	$6.6825e-012$	$6.6825e-012$
	median	$4.8330e-012$	$4.8330e-012$	$4.5881e-012$	$4.5881e-012$
	st.dev	$4.3478e-012$	$4.3478e-012$	$5.2359e-012$	$5.2359e-012$
6.19	suc.rate	30/30	30/30	30/30	30/30
	best	$1.0660e-015$	$1.0660e-015$	$2.7355e-015$	$2.7355e-015$
	worst	$5.1639e-013$	$5.1639e-013$	$1.8216e-012$	$1.8216e-012$
	mean	$6.9774e-014$	$6.9774e-014$	$3.0685e-013$	$3.0685e-013$
	median	$1.0666e-015$	$1.0666e-015$	$8.4956e-014$	$8.4956e-014$

continue goes here…

Problem	statistic	SLOM1	GOM1	SLOM2	GOM2
	st.dev	$1.7817e-013$	$1.7817e-013$	$4.5055e-013$	$4.5055e-013$
6.20	suc.rate	$30/30$	$30/30$	$30/30$	$30/30$
	best	$1.5253e-010$	$1.5253e-010$	$7.0455e-011$	$7.0455e-011$
	worst	$5.6948e-010$	$5.6948e-010$	$1.6898e-010$	$1.6898e-010$
	mean	$1.3298e-010$	$1.3298e-010$	$1.3298e-010$	$1.3298e-010$
	median	$2.9937e-010$	$2.9937e-010$	$1.5952e-010$	$1.5952e-010$
	st.dev	$1.5074e-010$	$1.5074e-010$	$5.4358e-011$	$5.4358e-011$

From table 6.3, we can see SLOM1 and SLOM2 behave similarly. As local optimization methods, SLOM1 and SLOM2 can also be considered as competitive algorithms with producing impressive results.

Actually, it did not need to partition the interval or increase the degree to 10 for most of the above problems in SLOM1 or SLOM2. For example, for Problem 6.1-6.4, 6.6-6.9, 6.11-6.13, 6.15-6.20, we can obtain the global minimizer by taking $N = 2, 3, 4$ for 30 randomly selected starting points. For Problem 6.10, from some starting points of 30, we can obtain the global minimizer by taking $N = 2, 3, 4$ only and for the rest starting points of 30, we can obtain the global minimizer by taking $S = 2$ or taking $N = 10$. For Problem 6.5 and 6.14, SLOM1 and SLOM2 failed from some starting points even by taking $S = 4$ or taking $N = 10$.

Next, we will compare GOM1 and GOM2 with two other heuristic methods: simulated annealing heuristic pattern search (SAHPS) [12] and quasi-filled function method (QFFM) [149].

Table 6.4.: Comparisons among various algorithms for (NLP)

Problem	statistic	SAHPS	QFFM	GOM1	GOM2
6.1	suc.rate	$30/30$	$30/30$	$30/30$	$30/30$
	best	0.3979	0.3979	0.3979	0.3979
	worst	0.3979	0.3979	0.3979	0.3979
	mean	0.3979	0.3979	0.3979	0.3979
	median	0.3979	0.3979	0.3979	0.3979
	st.dev	$3.7326e-009$	$5.1121e-014$	$7.8971e-014$	$4.7145e-014$
6.2	suc.rate	$21/30$	$30/30$	$30/30$	$30/30$
	best	$8.9346e-011$	0	0	0
	worst	0.4699	$2.5535e-015$	$2.7756e-015$	$3.7748e-015$
	mean	0.1296	$1.0399e-015$	$6.5873e-016$	$7.9566e-016$
	median	$3.6152e-009$	$1.1102e-015$	0	0
	st.dev	0.2019	$9.6368e-016$	$8.7949e-016$	$1.0705e-015$
6.3	suc.rate	$26/30$	$30/30$	$30/30$	$30/30$
	best	$6.4645e-010$	$2.2760e-015$	0	0
	worst	0.2183	$7.8826e-015$	$4.3854e-015$	$7.8826e-015$
	mean	0.0291	$2.6627e-015$	$1.9725e-015$	$1.7967e-015$
	median	$3.7839e-009$	$2.3870e-015$	$2.2760e-015$	$1.7208e-015$
	st.dev	0.0755	$1.0336e-015$	$8.6536e-016$	$1.5724e-015$
6.4	suc.rate	$20/30$	$30/30$	$30/30$	$30/30$
	best	$6.0262e-010$	0	$5.5511e-017$	0
	worst	0.2263	$4.2688e-014$	$4.0190e-014$	$5.5456e-014$

continue goes here…

Problem	statistic	SAHPS	QFFM	GOM1	GOM2
	mean	0.0754	$2.2005e-014$	$2.5152e-014$	$1.3961e-014$
	median	$5.2473e-009$	$2.4092e-014$	$3.0836e-014$	$1.6875e-014$
	st.dev	0.1085	$1.0517e-014$	$1.0495e-014$	$1.4312e-014$
6.5	suc.rate	$0/30$	$9/30$	$30/30$	$27/30$
	best	$-9.9396e-021$	-1.0000	-1	-1.0000
	worst	0	0	-1.0000	-0
	mean	$-3.3132e-022$	-0.3000	-1.0000	-0.9000
	median	0	0	-1.0000	-1.0000
	st.dev	$1.8147e-021$	0.4661	$2.4737e-014$	0.3051
6.6	suc.rate	$20/30$	$30/30$	$30/30$	$30/30$
	best	-1.8013	-1.8013	-1.8013	-1.8013
	worst	-1.0000	-1.8013	-1.8013	-1.8013
	mean	-1.6144	-1.8013	-1.8013	-1.8013
	median	-1.8013	-1.8013	-1.8013	-1.8013
	st.dev	0.3003	$3.4456e-015$	$2.8223e-015$	$4.4042e-015$
6.7	suc.rate	$19/30$	$30/30$	$30/30$	$30/30$
	best	-186.7309	-186.7309	-186.7309	-186.7309
	worst	-54.4049	-186.7309	-186.7309	-186.7309
	mean	-157.4907	-186.7309	-186.7309	-186.7309
	median	-186.7309	-186.7309	-186.7309	-186.7309
	st.dev	42.6679	$1.4074e-012$	$1.4530e-012$	$1.3429e-012$
6.8	suc.rate	-	$30/30$	$30/30$	$30/30$
	best	-	$2.5455e-005$	$2.5455e-005$	$2.5455e-005$
	worst	-	$2.5455e-005$	$2.5455e-005$	$2.5455e-005$

continue goes here…

Problem	statistic	SAHPS	QFFM	GOM1	GOM2
	mean	-	$2.5455e-005$	$2.5455e-005$	$2.5455e-005$
	median	-	$2.5455e-005$	$2.5455e-005$	$2.5455e-005$
	st.dev	-	$6.6425e-013$	$1.8569e-013$	$7.7183e-014$
6.9	suc.rate	$28/30$	$30/30$	$30/30$	$30/30$
	best	-3.8600	-3.8600	-3.8600	-3.8600
	worst	-3.0859	-3.8600	-3.8600	-3.8600
	mean	-3.8084	-3.8600	-3.8600	-3.8600
	median	-3.8600	-3.8600	-3.8600	-3.8600
	st.dev	0.1964	$1.9807e-012$	$1.7965e-012$	$2.0214e-012$
6.10	suc.rate	$9/30$	$30/30$	$30/30$	$30/30$
	best	-10.1532	-10.1532	-10.1532	-10.1532
	worst	-2.6305	-10.1532	-10.1532	-10.1532
	mean	-5.3973	-10.1532	-10.1532	-10.1532
	median	-3.8690	-10.1532	-10.1532	-10.1532
	st.dev	3.3014	$1.3346e-013$	$1.3087e-013$	$1.3098e-013$
6.11	suc.rate	$10/30$	$24/30$	$30/30$	$30/30$
	best	-10.5321	-10.5321	-10.5321	-10.5321
	worst	-1.8535	-4.0790	-10.5321	-10.5321
	mean	-5.6426	-9.2415	-10.5321	-10.3520
	median	-4.0790	-10.5321	-10.5321	-10.5321
	st.dev	3.5815	2.6253	$1.2280e-013$	$1.2172e-013$
6.12	suc.rate	$21/30$	$30/30$	$30/30$	$30/30$
	best	-3.3224	-3.3224	-3.3224	-3.3224
	worst	-3.2032	-3.3224	-3.3224	-3.3224

continue goes here…

Problem	statistic	SAHPS	QFFM	GOM1	GOM2
	mean	-3.2866	-3.3224	-3.3224	-3.3224
	median	-3.3224	-3.3224	-3.3224	-3.3224
	st.dev	0.0556	$5.4112e-014$	$5.6529e-014$	$6.2499e-014$
6.13	suc.rate	-	$4/30$	$30/30$	$30/30$
	best	-	$7.6365e-005$	$7.6365e-005$	$7.6365e-005$
	worst	-	473.7534	$7.6365e-005$	$7.6365e-005$
	mean	-	185.5535	$7.6365e-005$	$7.6365e-005$
	median	-	236.8767	$7.6365e-005$	$7.6365e-005$
	st.dev	-	115.0547	$4.0204e-013$	$6.1747e-013$
6.14	suc.rate	$0/30$	$30/30$	$30/30$	$30/30$
	best	-8.9839	-9.6602	-9.6602	-9.6602
	worst	-3.0081	-9.6602	-9.6602	-9.6602
	mean	-5.2334	-9.6602	-9.6602	-9.6602
	median	-4.9169	-9.6602	-9.6602	-9.6602
	st.dev	1.4494	$2.6645e-014$	$6.8798e-015$	$2.6735e-014$
6.15	suc.rate	$4/30$	$30/30$	$30/30$	$30/30$
	best	$1.1045e-008$	0	0	0
	worst	136.3081	0	0	0
	mean	49.6813	0	0	0
	median	58.2048	0	0	0
	st.dev	38.8843	0	0	0
6.16	suc.rate	$0/30$	$30/30$	$30/30$	$30/30$
	best	0.1895	$5.5511e-016$	$5.7288e-014$	$2.9943e-013$
	worst	6.3315	$9.9787e-012$	$9.7844e-012$	$8.7973e-012$

continue goes here…

Problem	statistic	SAHPS	QFFM	GOM1	GOM2
	mean	2.0503	$2.2377e-012$	$2.7571e-012$	$3.0619e-012$
	median	1.6968	$1.2696e-012$	$2.1166e-012$	$2.2805e-012$
	st.dev	1.4573	$2.6958e-012$	$2.6216e-012$	$2.3935e-012$
6.17	suc.rate	0/30	30/30	30/30	30/30
	best	0.0076	0	0	0
	worst	272.6164	0	0	0
	mean	139.0301	0	0	0
	median	138.7959	0	0	0
	st.dev	64.8933	0	0	0
6.18	suc.rate	0/30	30/30	30/30	30/30
	best	0.0099	$9.9920e-016$	$1.7958e-012$	$1.5604e-012$
	worst	1.5758	$5.2655e-011$	$1.8837e-011$	$1.9730e-011$
	mean	0.3955	$8.9460e-012$	$6.4374e-012$	$6.6825e-012$
	median	0.2882	$6.4375e-012$	$4.8330e-012$	$4.5881e-012$
	st.dev	0.4084	$1.0764e-011$	$4.3478e-012$	$5.2359e-012$
6.19	suc.rate	0/30	30/30	30/30	30/30
	best	46.4558	$8.7599e-016$	$1.0660e-015$	$2.7355e-015$
	worst	205.5195	$2.7737e-012$	$5.1639e-013$	$1.8216e-012$
	mean	86.6739	$3.1436e-013$	$6.9774e-014$	$3.0685e-013$
	median	77.5467	$4.0304e-014$	$1.0666e-015$	$8.4956e-014$
	st.dev	36.0563	$7.1156e-013$	$1.7817e-013$	$4.5055e-013$
6.20	suc.rate	0/30	30/30	30/30	30/30
	best	17.4409	$3.7834e-011$	$1.5253e-010$	$7.0455e-011$
	worst	19.9983	$5.3527e-010$	$5.6948e-010$	$1.6898e-010$

continue goes here…

Problem	statistic	SAHPS	QFFM	GOM1	GOM2
	mean	19.0563	$1.9748e-010$	$1.3298e-010$	$1.3298e-010$
	median	19.0168	$1.9176e-010$	$2.9937e-010$	$1.5952e-010$
	st.dev	0.6045	$9.5080e-011$	$1.5074e-010$	$5.4358e-011$

It is shown from table 6.4 that SAHPS is not successful for many test problems. QFFM exhibits the robustness on most test problems. GOMs can successfully solve almost all the test problems except that GOM2 can only successfully solve Problem 6.5 (Easom function problem) 27 out of 30 times. Hence, GOMs are the most efficient and stable, which combine the new local optimization methods SLOMs and the QFFM.

6.3.6. Conclusion

An optimality condition for the problem (NLP) is provided by using linear transportations and Lagrange interpolating polynomial. Two new local optimization methods SLOM1 and SLOM2 are designed according to this condition. The significance of the new local optimization methods is that instead of solving a complex nonlinear programming problem, we solve some simple univariate polynomial programming problems. Global optimization methods GOM1 and GOM2 are designed by combining the new local optimization methods and an auxiliary function.

We evaluate the performance of the proposed SLOMs and GOMs by using 20 benchmark functions for testing and comparing GOMs with two other heuristic methods: SAHPS and QFFM. The results demonstrate that GOMs are very robust and efficient optimization algorithms. In all cases of numerical experiments, they can almost successfully solve all the test problems except that GOM2 can only successfully solve Problem 6.5 (Easom function problem) 27 out of 30 times. Although SLOMs are local optimization methods, they perform a lot better in terms of computational efficiency compared to the SAHPS method. Since QFFM have fine performance relating to global search ability and convergence accuracy, it confirms the effectiveness of GOMs which combine SLOMs and the QFFM.

6.4. Conclusion

In this chapter, we apply our strongly or $\varepsilon-$strongly local optimization methods and global optimization methods to solve the sensor network localization problems and the systems of polynomial equations. The results illustrate that our methods are very efficient and stable. It is worth mentioning that we apply our idea - presenting optimality conditions, designing new local optimization methods according to these optimality conditions and designing global optimization methods by combining new local methods and some auxiliary functions - to nonlinear programming problems. The numerical results demonstrate that our methodology to solve nonlinear programming problems is comparable and promising.

Conclusions and future work

For global optimization, much attention has been paid on two aspects: one is global optimality conditions; the other is global optimization methods. This thesis focuses on both the global optimality conditions and optimization methods for some polynomial programming problems.

At the early stage, we considered cubic programming problems with mixed variables and quartic programming problems with box constraints, which have a wide range of practical applications as well. For these two problems, we proposed necessary global optimality conditions. Based on these conditions, we designed strongly local minimization methods. Global minimization methods were established by combining the local minimization methods and auxiliary functions.

Then, we developed the global necessary optimality conditions for general unconstrained and constrained polynomial programming problems. We designed strongly local minimization methods according to these necessary conditions and global minimization methods combining the local minimization methods and an auxiliary function.

Finally, we discussed some applications for solving some sensor network localization problems and systems of polynomial equations. The results showed our methods are efficient. It was worth mentioning that we applied the idea and the results for polynomial programming problems to nonlinear programming problems (NLP). We provided an optimality condition and designed new local optimization methods according to the optimality condition and global optimization methods for (NLP). The numerical results demonstrate that our methodology to solve nonlinear programming problems is comparable and promising.

Our contribution

Global optimality conditions are very important topics. Various necessary global optimality conditions and sufficient global optimality conditions for quadratic programming problems and some special polynomial programming problems have been developed recently. To the authors' best knowledge, there are few checkable global optimality conditions for general polynomial programming problems. The significance of the thesis is due to several aspects. First of all, we propose easily checkable necessary global optimality conditions for some polynomial programming problems which are generally stronger than KKT conditions. Secondly, as traditional local optimization methods are designed based on KKT local conditions, we establish strongly local optimization methods based on the necessary global optimality conditions which may improve some KKT points. Thirdly, we provide global optimization methods by combining the strongly local methods and some auxiliary functions. Finally, we extend the similar idea for polynomial programming problems to nonlinear programming problems and give an optimality condition and design $\varepsilon-$strongly local optimization methods and global optimization methods. The numerical results showed the methods are efficient and stable.

Future work

1. Checkable sufficient global optimality conditions

We proposed checkable necessary global optimality conditions for some polynomial programming problems. Our future work will concentrate on checkable sufficient global conditions for these polynomial programming problems.

2. New auxiliary functions

In this thesis, we used different auxiliary functions for different programming problems. We know the behavior of an auxiliary function directly depends on the construction of the auxiliary function. We will try to construct some new auxiliary functions which are tailor-made for polynomial programming problems.

3. Difference of Convex functions (DC) programming problems

In this thesis, we considered some polynomial programming problems and nonlinear programming problems. This study could go further to DC programming problems if we have more time in the future.

4. Large scale problems

We have tested our algorithms on some large scale problems. However the methods presented in this thesis are not designed for very large scale problems and at this stage, they are time-consuming. These methods will aim at developing the solvability of very large scale polynomial programming problems in the future.

Bibliography

[1] A. A. Ahmadi, A. Olshevsky, P.A. Parrilo and J.N. Tsitsiklis, NP-hardness of deciding convexity of quartic polynomials and related problems, *MIT LIDS technical report*. 2010.

[2] A.A. Zhigljavsky, Theory of global random search, *J. Pinter (Ed.). Springer Netherlands*. 1991.

[3] A. Beck and M. Teboulle, Global optimality conditions for quadratic optimization problems with binary constraints, *SIAM Journal on Optimization*. 11(1), 2000, 179–188.

[4] A. Cantoni and M. C. Blasikiewiez, Derivative constrained broadband antenna array processors revisited, *In Proceedings of IASTED, 2nd International Symposium on Signal Processing and Its Applications, Gold Coast, Queensland*. 1990, 319–326.

[5] A. J. Sommese, J. Verschelde and C. W. Wampler, Numerical algebraic geometry. *In: The Mathematics of Numerical Analysis. 32 of Lectures in Applied Mathematics*, 1996.

[6] A. Morgan, Solving Polynomial Systems Using Continuation for Engineering and Scientific Problems, *SIAM*. 57, 2009.

[7] A.M.C. So and Y. Ye, Theory of semidefinite programming for sensor network localization, *Mathematical Programming*. 109(2-3), 2007, 367–384.

[8] A. M. Rubinov, Abstract Convexity and Global Optimization, *Kluwer*. 2000.

[9] A.M Rubinov and Z.Y Wu, Optimality conditions in global optimization and their applications, *Mathematical programming*. 120(1), 2009, 101-123.

[10] A. Neumaier, Interval methods for systems of equations (Vol. 37). *Cambridge university press*. 1990.

[11] A.P. Roberts and M.M. Newmann, Polynomial Optimization of Stochastic Feedback Control for Stable Plants, *IMA Journal of Mathematical Control and Information*. 5(3), 1988, 243–257.

[12] A.R. Hedar and F. Masao, Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization, *Optimization Methods and Software*. 19(3-4), 2004, 291–308.

[13] A.V. Levy and A. Montalvo, The tunneling algorithm for the global minimization of functions, *SIAM Journal on Scientific and Statistical Computing*. 6(1), 1985, 15–29.

[14] B. JIANG, Z. LI and S. ZHANG, Approximation methods for complex polynomial optimization, *Technical report, Department of Industrial and Systems Engineering, University of Minnesota*. 2012.

[15] B. Maricic, Z.Q. Luo and T.N. Davidson, Blind constant modulus equalization via convex optimization, *Signal Processing, IEEE Transactions on*. 51(3), 2003, 805–818.

[16] C.A. Floudas, Deterministic Global Optimization: Theory, Methods and Applications, *Kluwer Academic Publishers*. 1999.

[17] C.A. Floudas and P.M.A. Pardalos, Collection of test problems for constrained global

optimization algorithms, *In: Goos, G., Hartmanis, J.(eds.) Lecture Notes in Computer Science. Springer-Verlag*. 1990.

[18] C.A. Floudas, P.M. Pardalos and C.S. Adjiman, etc., Handbook of Test Problems in Local and Global Optimization, *Kluwer Academic Publishers*. 1999.

[19] C.A. Floudas and V. Visweswaran, A global optimization algorithm (GOP) for certain classes of nonconvex NLPs-I. theorey, *Computers and Chemical Engineering*. 14(12), 1990, 1397–1417.

[20] C.A. Floudas and V. Visweswaran, Quadratic optimization, *in R. HORST, P.M. Pardalos, eds., Handbook of Global Optimization, Kluwer Academic. Dordrecht*. 1995, 217–269.

[21] C. Henin and J. Doutriaux, A specialization of the convex simppex method to cubic programming, *Decisions in Economics and Finance*. 3(2), 1980, 61–72.

[22] C. Kanzow, Global optimization techniques for mixed complementarity problems, *Journal of Global Optimization*. 16, 2000, 1–21.

[23] C. Ling, X. Zhang and L. Qi, Semidefinite relaxation approximation for multivariate bi-quadratic optimization with quadratic constraints, *Numerical Linear Algebra with Applications*, 19(1), 2012, 113–131.

[24] C. Parnell, Numerical analysis, *Lecture Notes*. Retrieved January 20, 2014, from http://www-solar.mcs.st-and.ac.uk/ clare/Lectures/num-analysis.html.

[25] C.R. Bector, Indefinite cubic programming with standard errors in bbjective function, *Unternehmensforschung*. 12(1), 1968, 113–120.

[26] D.A. Bini, L. Gemignani and V.Y. Pan, Inverse power and Durand-Kerner iterations

for univariate polynomial root-finding, *Computers and Mathematics with Applications*. 47(2), 2004, 447–459.

[27] D.A. Cox, J.B. Little and D. O'Shea, Using algebraic geometry, *Graduate Textx in Mathematics, Springer-Verlag*. 185, 1998.

[28] D. Bertsimas, R. Freund and X. Sun, An accelerated first-order method for solving unconstrained SOS polynomial optimization problems, *Optimization Methods and Software*. 2011.

[29] D. Den Hertog, Interior point approach to linear, quadratic and convex programming: algorithms and complexity. 1992.

[30] D. Han, Global optimization with polynomials, *http://hdl.handle.net/1721.1/3883*. 2004. Accessed 20 December 2013.

[31] D. Henrion and J.B. Lasserre, GloptiPoly: Global optimization over polynomials with Matlab and SeDuMi, *ACM Transactions on Mathematical Software (TOMS)*. 29(2), 2003, 165–194.

[32] D. Henrion, J.B. Lasserre and J. Lofberg, GloptiPoly 3: moments, optimization and semidefinite programming, *Optimization Methods and Software*. 24(4-5), 2009, 761–779.

[33] D. Keren, D. Cooper and J. Subrahmonia, Describing complicated objects by implicit polynomials, *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. 16(1), 1994, 38–53.

[34] D. Manocha, Solving systems of polynomial equations, *Computer Graphics and Applications, IEEE*, 14(2), 1994, 46–55.

[35] D.S. Hanif and H.T. Cihan, A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique, *Journal of Global Optimization*. 2(1), 1992, 101–112.

[36] E. Balas, S. Ceria and G. Cornuejols, A lift-and-project cutting plane algorithm for mixed 0C1 programs, *Mathematical programming*. 58(1-3), 1993, 295–324.

[37] E.D. Andersen, C. Roos and T. Terlaky, On implementing a primal-dual interior-point method for conic quadratic optimization. *Mathematical Programming*. 95(2), 2003, 249–277.

[38] E.D. Klerk, G. Elabwabi and D.D. Hertog, Optimization of univariate functions on bounded intervals by interpolation and semidefinite programming, *CentER Discussion Paper Series No. 2006-26. Available at SSRN: http://ssrn.com/abstract=900108 or ttp://dx.doi.org/10.2139/ssrn.900108 (April 11, 2006)*. Accessed 11 November 2013

[39] E.L.S. Wong, Active-set methods for quadratic programming. 2011.

[40] E.R. Hansen, Global optimization using interval analysis: the one-dimensional case, *Journal of Optimization Theory and Applications*. 29(3), 1979, 331-344.

[41] F. Glover and M. Laguna, Tabu Search. *Kluwer Academic Publishers, Dordrecht / Boston / London*. 1997.

[42] F. Jiang, H. Baoyin and J. Li, Practical techniques for low-thrust trajectory optimization with homotopic approach, *Journal of Guidance, Control, and Dynamics*. 35(1), 2012, 245–258.

[43] G. Danninger and I.M. Bomze, Using copositivity for global optimality criteria in concave quadratic programming problems, *Mathematical Programming*. 62(1-3), 1993, 575–580.

[44] G. Hanoch and H. Levy, Efficient portfolio analysis with quadratic and cubic utility, *The Journal of Business*. 43(2), 1970, 181–198.

[45] G.Q. Li, Z.Y. Wu and J. Quan, A new local and global optimization method for mixed integer quadratic programming problems, *Applied Mathematics and Computation*. 217(6), 2010, 2501–2512.

[46] H.D. Sherali, Global optimization of nonconvex polynomial programming problems having rational exponents, *Journal of Global Optimization*. 12(3), 1998, 267–283.

[47] H.D. Sherali and C.H. Tuncbilek, New reformulation linearization/convexification relaxations for univariate and multivariate polynomial programming problems *Operations Research Letters*. 21(1), 1997, 1–9.

[48] H.D. Sherali and C.H. Tuncbilek, Comparison of two reformulation-linearization technique based linear programming relaxations for polynomial programming problems, *Journal of Global Optimization*. 10(4), 1997, 381–390.

[49] H. Levy and M. Sarnat, Investment and Portfolio Analysis, *New york: Wiley*. 1972.

[50] H. M. Markowitz, Portfolio Selection. *The journal of finance*. 7(1), 1952, 79–91.

[51] H.M. Moller and H.J. Stetter, Multivariate Polynomial Equations with Multiple Zeros Solved by Matrix Eigenproblems, *Numerische Mathematike*. 70, 1995, 311–329.

[52] H.P. Benson, Concave minimization: theory, applications, and algorithms, *In Horst, R. and Pardalos, P.M., editors, Handbook of Global Optimization*. 43–148, 1995, Kluwer Academic Publishers, Dordrecht / Boston / London.

[53] H. Ratschek and J. Rokne, New computer methods for global optimization, *Halsted Press*. 1988.

[54] H. Schichl and A. Neumaier, Transposition theorems and qualification-free optimality conditions, *SIAM Journal on Optimization*, 17(4), 2006, 1035–1055.

[55] I. Diener, Trajectory methods in global optimization, *In Handbook of Global optimization, Springer US*. 1995, 649–668.

[56] I.H. Osman and J.P. Kelly, (Eds.). Meta-heuristics: theory and applications, *Springer*. 1996.

[57] I. M. Bomze and G. Danninger, A global optimization algorithm for concave quadratic programming problems, *SIAM Journal on Optimization*. 3(4), 1993, 826–842.

[58] I. M. Bomze and G. Danninger, A finite algorithm for solving general quadratic problems, *Journal of Global Optimization*. 4(1) , 1994, 1–16.

[59] I.M. Gelfand, M.M. Kapranov and A.V. Zelevinsky, A-Discriminants. *Birkhauser Boston*. 1994, 271–296.

[60] I.M., Stancu-Minasian, Optimality and duality in nonlinear programming involving semilocally B-preinvex and related functions, *European journal of operational research*, 173(1), 2006, 47–58.

[61] I. Thng, A. Cantoni and Y. H. Leung, Derivative constrained optimum broad-band antenna arrays, *Signal Processing, IEEE Transactions on*. 41(7), 1993, 2376–2388.

[62] I. Thng, A. Cantoni and Y. H. Leung, Analytical solutions to the optimization of a quadratic cost function subject to linear and quadratic equality constraints, *Applied Mathematics and Optimization*. 34(2), 1996, 161–182.

[63] I.Z. Emiris and J. Verschelde, How to count efficiently all affine roots of a polynomial system, *Discrete Applied Mathematics*. 93(1), 1999, 21–32.

[64] J.B. Hiriart-Urruty, Conditions for global optimality 2, *Journal of Global Optimization*. 13(4), 1998, 349–367.

[65] J.B. Hiriart-Urruty, Global optimality conditions in maximizing a convex quadratic function under convex quadratic constraints, *Journal of Global Optimization*. 21(4), 2001, 443–453.

[66] J.B. Hiriart-Urruty and C. Lemarechal, Testing necessary and sufficient conditions for global optimality in the problem of maximizing a convex quadratic funcion over a convex polyhedron, *Preliminary Report, University of Paul Sabatier, Toulouse*. 1990.

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

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

[69] J.B. Lasserre, Convexity in semialgebraic geometry and polynomial optimization, *SIAM Journal on Optimization*. 19(4), 2009, 1995–2014.

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

[71] J.D. Pinter, Global Optimization in Action (Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications), *Kluwer Academic Publishers, Dordrecht / Boston / London*. 1996.

[72] J.J. Grefenstette, Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, *Psychology Press*. 2013.

[73] J.J. More and Z. Wu, Global continuation for distance geometry problems, *SIAM Journal on Optimization*. 7(3), 1997, 814–836.

[74] J. Hichert, A. Hoffman, and H.X. Phu, Convergence speed of an integral method for computing the essential supremum. *In I.M. Bomze, T. Csendes, R. Horst and P.M. Pardalos, editors, Developments in Global Optimization*. 153–170, 1997, Kluwer Academic Publishers, Dordrecht / Boston / London.

[75] J. Maryak, Bayesian Heuristic Approach to Discrete and Global Optimization, *Technometrics*. 41(4), 1999, 378–379.

[76] J.M Peng and Y. Yuan, Optimality conditions for the minimization of a quadratic with two quadratic constraints, *SIAM Journal on Optimization*. 7(3), 1997, 579–594.

[77] J. Nie, Regularization methods for sum of squares relaxations in large scale polynomial optimization, *No. arXiv:0909.3551*. 2009.

[78] J. Nie, Sum of squares method for sensor network localization, *Computational Optimization and Applications*. 43(2), 2009, 151–179.

[79] J. Nie and J. Demmel, Minimum ellipsoid bounds for solutions of polynomial systems via sum of squares, *Journal of Global Optimization*. 33(4), 2005, 511–525.

[80] J. Nie, J. Demmel and B. Sturmfels, Minimizing polynomials via sum of squares over the gradient, *Mathematical programming*. 106(3), 2006, 587–606.

[81] J. Nie and L. Wang, Regularization methods for SDP relaxations in large-scale polynomial optimization. *SIAM Journal on Optimization*. 22(2), 2012, 408–428.

[82] J. Quan , Z.Y. Wu and G.Q. Li, Global optimality conditions for some classes of polynomial integer programming problems, *Journal of Industrial and Management Optimization*. 7(1), 2011, 67–78.

[83] J. Verschelde, Polynomial homotopies for dense, sparse and determinantal systems, *arXiv preprint math/9907060*. 1999.

[84] K.A. Dill, A.T. Phillips and J.B. Rosen, Molecular structure prediction by global optimization, *In Developments in Global Optimization, Springer US*. 1997, 217–234.

[85] K. Kianfar, Branch-and-Bound Algorithms, *Wiley Encyclopedia of Operations Research and Management Science*. 2010.

[86] L.C.W. Dixon and R.C. Price, The Truncated Newton Method for Sparse Unconstrained Optimisation Using Automatic Differentiation, *Journal of optimization theory and applications*. 60(2), 1989, 261–275.

[87] L.J. Xie, A Note on Sturm Theorem, *Mathematisc in Practice and Theory*. 37(1), 2007, 121–125.(in Chinese)

[88] L. Qi and K.L. Teo, Multivariate polynomial minimization and its application in signal processing, *Journal of Global Optimization*. 26(4), 2003, 419–433.

[89] L. Qi, Z. Wan, and Y.F. Yang, Global miminization of normal quartic polynomial based on global descent directions, *SIAM Journal on Optimization*. 15(1), 2004, 275–302.

[90] L.S. Zhang, C.K. NG, D. Li and W.W. Tian, A new filled function method for global optimization, *Journal of Global Optimization*. 28(1), 2004, 17–43.

[91] M.C. Pinar, Sufficient global optimality conditions for bivalent quadratic optimization, *Journal of optimization theory and applications*. 122(2), 2004, 433–440.

[92] M.H. Er and A. Cantoni, Techniques in robust broadband beamforming, *Control and Dynamic Systems V53: High Performance Systems Techniques and Applications: Advances in Theory and Applications*. 1992.

[93] M. Kojima, S. Kim, and H. Waki, A general framework for convex relaxation of polynomial optimization problems over cones, *Journal of Operations Research Society of Japan*. 46(2), 2003, 125–144.

[94] M. Kojima and L. Tuncel, Cones of matrices and successive convex relaxations of nonconvex sets, *SIAM Journal on Optimization*. 10, 2000, 750–778.

[95] M. Kojima and L. Tuncel, Discretization and localization in successive convex relaxation methods for nonconvex quadratic optimization problems, *Mathematical Programming*. 89, 2000, 79–111.

[96] M.L. Hazelton, Some comments on origin-destination matrix estimation, *A Transportation Research Part A*. 37, 2003, 811–822.

[97] M. Laguna and R. Marti, Experimental testing of advanced scatter search designs for global optimization of multimodal functions, *Journal of Global Optimization* . 33(2), 2005, 235–255.

[98] M. Pelikan, Bayesian optimization algorithm. *In Hierarchical Bayesian Optimization Algorithm. Springer Berlin Heidelberg*. 2005, 31–48.

[99] M.R. Garey and D.S. Johnson, Computers and intractability: a guide to the theory of NP-completeness, *A Series of Books in the Mathematical Sciences*. 1979.

[100] M.S. Bazaraa, H.D. Sherali and C.M. Shetty, Nonlinear Programming: Theory and Algorithms, 3rd edition, *John Wiley and Sons, Inc., Hoboken, New Jersey*. 2006.

[101] N. Courtois, A. Klimov, J. Patarin and A. Shamir, Efficient algorithms for solving overdefined systems of multivariate polynomial equations, *In Advances in Cryptology-WUROCRYPT. Springer Berlin Heidelberg*. 2000, 392–407

[102] P. Biswas, T.C. Liang, T.C. Wang and Y. Ye, Semidefinite programming based algorithms for sensor network localization, *ACM Transactions on Sensor Networks (TOSN)*. 2(2), 2006, 188–220.

[103] P.D. Angelis, P. Pardalos and G. Toraldo, Quadratic programming with box constraints, *Developments in global optimization. Springer US*. 1997, 73–93.

[104] P.E. Gill and E. Wong, Methods for convex and general quadratic programming, *Numerical Analysis Report*. 10-1, 2013.

[105] P.M. Pardalos and H.E. Romeijin, Handbook of Global Optimization Volume 2, *Kluwer Academic Publishers*. 2002.

[106] P.M. Pardalos, H.E. Romeijin and T. Hoang, Recent developments and trends in global optimization, *Journal of Computational and Applied Mathematics*. 124(1), 209-228, 2000.

[107] P. Parrilo and B. Sturmfels, Minimizing polynomial functions, *Proceedings of the DIMACSWorkshop on Algorithmic and Quantitative Aspects of Real Algebraic Geometry in Mathematics and Computer Science (March 2001), In: Basu, S., Gonzalez-Vega, L.(eds.). American Mathematical Society*. 2003, 83–100.

[108] P. Tseng, Second-order cone programming relaxation of sensor network localization, *SIAM Journal on Optimization*. 18(1), 2007, 156–185.

[109] Q. Zheng and D. Zhuang, Integral global minimization: algorithms, implementations and numerical tests, *Journal of Global Optimization*, 7(4), 1995, 421–454.

[110] R.D. Monteiro, First-and second-order methods for semidefinite programming, *Mathematical Programming*. 97(1-2), 2003, 209–244.

[111] R. Ge, A filled function method for finding a global minimizer of a function of several variables, *Mathamatical Programming*. 46(1-3), 1990, 191–204.

[112] R. Ge, The globally convexized filled functions for global optimization, *Applied Mathematics and Computation*. 35, 1990, 131–158.

[113] R. Horst and H. Tuy, Global optimization: Deterministic approaches, *Springer*. 1996.

[114] R. Kasimbeyli and M. Mammadov, Optimality conditions in nonconvex optimization via weak subdifferentials. *Nonlinear Analysis: Theory, Methods and Applications*, 74(7), 2011, 2534–2547.

[115] R.N. Kaul and S. Kaur, (1982). Generalizations of convex and related functions *European Journal of Operational Research*. 9(4),1982, 369–377.

[116] S. Fortune, An iterated eigenvalue algorithm for approximating roots of univariate polynomials, *Journal of Symbolic Computation*. 33(5), 2002, 627–646.

[117] S. He, Z. Li and S. Zhang, Approximation algorithms for homogeneous polynomial optimization with quadratic constraints, *Mathematical programming*. 125(2), 2010, 353–383.

[118] S. Kim and M. Kojima, Second order cone programming relaxation of nonconvex quadratic optimization problems, *Optimization Methods and Software*. 15, 2001, 201–224.

[119] S. Martello, Jeno Egervary: from the origins of the Hungarian algorithm to satellite communication, *Central European Journal of Operations Research*, 18(1), 2010, 47–58.

[120] S. Schaible, Quasi concavity and pseudo concavity of cubic functions, *Mathematical Programming*. 5(1), 1973, 243–247.

[121] T. Antczak, New optimality conditions and duality results of G type in differentiable mathematical programming, *Nonlinear Analysis: Theory, Methods and Applications*, 66(7), 2007, 1617–1632.

[122] T.J. Rivlin, An introduction to the approximation of functions, *Dover, New York*. 1981.

[123] T.Y. Li, T. Sauer and J.A. Yorke, (1989). The cheater's homotopy: an efficient procedure for solving systems of polynomial equations, *SIAM Journal on Numerical Analysis*, 26(5), 1989, 1241–1251.

[124] V. Jeyakumar, A.M. Rubinov and Z.Y. Wu, Sufficient global optimality conditions for non-convex quadratic minimization problems with box constraints, *Journal of Global Optimization*. 36(3), 2006, 471–481.

[125] V. Jeyakumar, A.M. Rubinow and Z.Y. Wu, Non-convex quadratic minimization problems with quadratic constraints: global optimality conditions, *Mathematical Programming*. 110(3), 2007, 521–541.

[126] V. Jeyakumar and G.Y. Li, Necessary global optimality conditions for nonlinear programming problems with polynomial constraints, *Mathematical Programming*. 126(2), 2011, 393–399.

[127] V. Jeyakumar, G. Li and S. Srisatkunarajah, Global optimality principles for polynomial optimization over box or bivalent constraints by separable polynomial approximations, *Journal of Global Optimization*. 2013, 1–20.

[128] V.V. Prasolov, Polynomials, *Springer Berlin Heidelberg New York*. 2004.

[129] V. Visweswaran and C.A. Floudas, Unconstrained and constrained global optimization of polynomial functions in one variable, *Journal of Global Optimization*. 2 (1), 1992, 73–99.

[130] W.X. Zhu, A class of filled functions for box constrained continuous global optimization, *Applied Mathematics and Computation*. 169(1), 2005, 129–145.

[131] W.X. Zhu, Globally concavized filled function method for the box constrained continuous global minimization problem, *Optimisation Methods and Software*. 21(4), 2006, 653–666.

[132] X. Liu, Finding global minima with a computable filled function, *Journal of Global Optimization*. 19(2), 2001, 151-161.

[133] X. Liu, Several filled functions with mitigators, *Applied Mathematics and Computation*. 133(2), 2002, 375–387.

[134] X. Liu, A class of genaralilized filled functions with improved computability, *Journal of Computational and Applied Mathematics*. 137(1), 2001, 62–69.

[135] Y.J. Chang and B.W. Wah, Polynomial programming using groebner bases, *Computer Software and Applications Conference COMPSAC 94. Proceedings., Eighteenth Annual International*. 1994, 236–241.

[136] Y.J. Wang and Z.A. Liang, Global optimality conditions for cubic minimization problem with box or binary constraints, *Journal of Global Optimization*, 47(4), 2010, 583–595.

[137] Y. Nesterov, Squared functional systems and optimization problems, *High Performance Optimization, H. Frenk et al., eds., Kluwer Academic Publishers*. 2000, 405–440.

[138] Z.B. Zabinsky, Stochastic adaptive search for global optimization, *Springer*. 72, 2003.

[139] Z.Q. Luo and S. Zhang, A semidefenite relaxation scheme for multivariate quartic

polynomial optimization with quadratic constraints, *SIAM Journal on Optimization*. 20(4), 2010, 1716–1736.

[140] Z. Wan and C. H. Yang, New approach to global minimizaton of normal multivariate polynomial based on tensor, *Journal of Industrial and Management Optimization*. 4(2), 2008, 271–285.

[141] Z. Xu, H.X. Huang, P.M. Pardalos and C.X. Xu, Filled functions for unconstrained global optimization, *Journal of Global Optimization*. 20(1), 2001, 49–65.

[142] Z.Y Wu, Sufficient global optimality conditions for weakly convex minimization programs, *Journal of Global Optimization*. 39(3), 2007, 427–440.

[143] Z.Y Wu and A.M Rubinov, Global optimality conditions for some classes of optimization problems, *Journal of optimization theory and applications*. 145(1), 2010, 164–185.

[144] Z.Y. Wu, J. Quan, G.Q. Li and J. Tian, Necessary optimality conditions and new optimization methods for cubic polynomial optimization problems with mixed variables, *Journal of Optimization Theory and Applications*. 153(2), 2012, 408–435.

[145] Z.Y. Wu and F.S. Bai, Global optimality conditions for mixed nonconvex quadratic programs, *Optimization*. 58(1), 2009, 39–47.

[146] Z.Y. Wu, F.S. Bai, H.W.J. Lee, Y.J. Yang, A filled function method for constrained global optimization, *Journal of Global Optimization*. 39, 2007, 495–507.

[147] Z.Y. Wu, F.S. Bai, Y.J. Yang, and M. Mammadov, A new auxiliary function method for general constrained global optimization, *Optimization*, 62(2), 2013, 193–210.

[148] Z.Y. Wu, F.S. Bai, G.Q. Li and Y.J. Yang, A new auxiliary function method for

system of nonlinear questions, *Journal of Industrial and Management Optimization* (accepted, Feburary 2014)

[149] Z.Y Wu, H.W.J. Lee, L.S. Zhang and X.M.Yang, A novel filled function method and quasi-filled function method for global optimization, *Computational Optimization and Applications*. 34(2), 2006, 249–272.

[150] Z.Y. Wu, J. Tian, J. Quan and J. Ugon, Optimality conditions and optimization methods for quartic polynomial optimization, *Applied Mathematics and Computation*. 232, 2014, 968–982.

[151] Z.Y. Wu, M. Mammadov, F.S. Bai and Y.J. Yang, A Filled Function Method for Nonlinear Equations, *Applied Mathematics and Computation*. 189, 2007, 1196–1204.

[152] Z.Y. Wu, V. Jeyakumar and A.M Rubinov, Sufficient conditions for globally optimality of bivalent nonconvex quadratic programs, *Journal of optimization theory and applications*. 133(1), 2007, 123–130.

[153] Z.Y. Wu, Y.J. Yang, F.S. Bai and J. Tian, Global optimality conditions and optimization methods for quadratic assignment problems, *Applied Mathematics and Computation*. 218(11), 2012, 6214–6231.

Appendix A.

Test problems for general polynomial programming problems

Problem 4.1: Beale Function

$$\begin{aligned} \min \quad & f(x) := (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2 \\ s.t. \quad & -4.5 \leq x_i \leq 4.5,\ i = 1, 2. \end{aligned}$$

Problem 4.2: Booth Function

$$\begin{aligned} \min \quad & f(x) := (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2 \\ s.t. \quad & -10 \leq x_i \leq 10,\ i = 1, 2. \end{aligned}$$

Problem 4.3: Matyas Function

$$\begin{aligned} \min \quad & f(x) := 0.26(x_1^2 + x_2^2) - 0.48x_1x_2 \\ s.t. \quad & -10 \leq x_i \leq 10,\ i = 1, 2. \end{aligned}$$

Problem 4.4: Goldstein and Price Function

$$\begin{aligned}\min\quad & f(x) := [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \\ & \times[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)] \\ s.t.\quad & -2 \le x_i \le 2,\ i = 1, 2.\end{aligned}$$

Problem 4.5: Six-hump Camelback Function

$$\begin{aligned}\min\quad & f(x) := (4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2 \\ s.t.\quad & -3 \le x_1 \le 3, -2 \le x_2 \le 3.\end{aligned}$$

Problem 4.6: Perm(3, 0.5) Function

$$\begin{aligned}\min\quad & f(x) := \sum_{i=1}^{n}[\sum_{j=1}^{n}(j^i + 0.5)((x_j/j)^i - 1)]^2 \\ s.t.\quad & x_i \in [-n, n],\ i = 1, 2, \cdots, n \\ \text{where}\quad & n = 3.\end{aligned}$$

Problem 4.7: Perm0(3, 10) Function

$$\begin{aligned}\min\quad & f(x) := \sum_{k=1}^{n}[\sum_{i=1}^{n}(i + 10)(x_i^k - (1/i)^k)]^2 \\ s.t.\quad & x_i \in [-n, n],\ i = 1, 2, \cdots, n \\ \text{where}\quad & n = 3.\end{aligned}$$

Problem 4.8: Perm(4, 0.5) Function

$$\begin{aligned} \min \quad & f(x) := \sum_{i=1}^{n}[\sum_{j=1}^{n}(j^i + 0.5)((x_j/j)^i - 1)]^2 \\ s.t. \quad & x_i \in [-n, n],\ i = 1, 2, \cdots, n \end{aligned}$$

$$\text{where} \quad n = 4.$$

Problem 4.9: Perm0(4, 10) Function

$$\begin{aligned} \min \quad & f(x) := \sum_{k=1}^{n}[\sum_{i=1}^{n}(i + 10)(x_i^k - (1/i)^k)]^2 \\ s.t. \quad & x_i \in [-n, n],\ i = 1, 2, \cdots, n \end{aligned}$$

$$\text{where} \quad n = 4.$$

Problem 4.10: Colville Function

$$\begin{aligned} \min \quad & f(x) := 100(x_1^2 - x_2^2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 \\ & +10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1) \\ s.t. \quad & -10 \le x_i \le 10,\ i = 1, 2, 3, 4. \end{aligned}$$

Problem 4.11: Powersum Function

$$\begin{aligned} \min \quad & f(x) := \sum_{i=1}^{4}[(\sum_{i=1}^{4} x_j^i) - b_i]^2 \\ s.t. \quad & 0 \le x_i \le n,\ i = 1, \cdots, 4. \end{aligned}$$

$$\text{where} \quad b = (8, 18, 44, 114).$$

Problem 4.12: Dixon and Price Function

$$\begin{aligned} \min \quad & f(x) := (x_1-1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2 \\ s.t. \quad & x_i \in [-10, 10],\ i = 1, 2, \cdots, n. \end{aligned}$$

where $n = 5.$

Problem 4.13: Dixon and Price Function

$$\begin{aligned} \min \quad & f(x) := (x_1-1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2 \\ s.t. \quad & x_i \in [-10, 10],\ i = 1, 2, \cdots, n. \end{aligned}$$

where $n = 10.$

Problem 4.14: Trid Function

$$\begin{aligned} \min \quad & f(x) := \sum_{i=1}^{n}(x_i-1)^2 - \sum_{i=2}^{n} x_i x_{i-1} \\ s.t. \quad & -n^2 \le x_i \le n^2,\ i = 1, 2, \cdots, n. \end{aligned}$$

where $n = 10.$

Problem 4.15: Rosenbrock Function

$$\begin{aligned} \min \quad & f(x) := \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \\ s.t. \quad & -5 \le x_i \le 10,\ i = 1, 2, \cdots, n. \end{aligned}$$

where $n = 20.$

Problem 4.16: Sum Squares Function

$$
\begin{aligned}
\min \quad & f(x) := \sum_{i=1}^{n} i x_i^2 \\
s.t. \quad & -10 \le x_i \le 10,\ i = 1, 2, \cdots, n.
\end{aligned}
$$

$$\text{where} \quad n = 20.$$

Problem 4.17: Zakharov Function

$$
\begin{aligned}
\min \quad & f(x) := \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5 i x_i)^2 + (\sum_{i=1}^{n} 0.5 i x_i)^4 \\
s.t. \quad & -5 \le x_i \le 10,\ i = 1, 2, \cdots, n.
\end{aligned}
$$

$$\text{where} \quad n = 20.$$

Problem 4.18: Powell Function

$$
\begin{aligned}
\min \quad & f(x) := \sum_{i=1}^{n/4} [(x_{4i-3} + 10 x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 \\
& +(x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4] \\
s.t. \quad & -4 \le x_i \le 5,\ i = 1, 2, \cdots, n.
\end{aligned}
$$

$$\text{where} \quad n = 24.$$

Problem 4.19: Sphere Function

$$
\begin{aligned}
\min \quad & f(x) := \sum_{i=1}^{n} x_i^2 \\
s.t. \quad & -5.12 \le x_i \le 5.12,\ i = 1, 2, \cdots, n.
\end{aligned}
$$

$$\text{where} \quad n = 30.$$

Problem 4.20

$$\begin{array}{lll} \min & & f(x) := \sum_{k=1}^{3}(\sum_{i=1}^{n} x_i^k - 1)^2 + \sum_{i=1}^{n}(x_{i-1}^2 + x_i^2 + x_{i+1}^2 - x_i^3 - 1)^2 \\ s.t. & & x_i \in [-500, 500], i = 1, \cdots, n. \end{array}$$

where $x_0 = x_{n+1} = 0,\ n = 16.$

Problem 4.21

$$\min f(x) := \sum_{i=1}^{m} f_i^2(x) s.t. \qquad x_i \in [-500, 500], i = 1, \cdots, n.$$

$n = 30$ and the polynomials f_i are defined as follows:

$$f_i(x) := \sum_{j=2}^{n}(j-1)x_j t_i^{j-2} - (\sum_{j=1}^{n} x_j t_i^{j-1})^2 - 1,\ t_i = \frac{i}{29},\ 1 \le i \le 29,$$

and $f_{30} = x_1$, $f_{31} = x_2 - x_1^2 - 1$.

Appendix B.

Test problems for general polynomial programming problems with polynomial constraints

Problem 5.1

$$
\begin{aligned}
\min \quad & f(x) := -2x_1 + x_2 - x_3 \\
s.t. \quad & x_1 + x_2 + x_3 \leq 4 \\
& x_1 \leq 2 \\
& x_3 \leq 3 \\
& 3x_2 + x_3 \leq 6 \\
& x_1, x_2, x_3 \geq 0 \\
& x^T B^T B x - 2r^T B x + \|r\|^2 - 0.25\|b - v\|^2 \geq 0
\end{aligned}
$$

where

$$
\begin{aligned}
B &= \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ -2 & 1 & -1 \end{bmatrix} \\
b &= [3, 0, -4] \\
v &= [0, -1, -6] \\
r &= [1.5, -0.5, -5]
\end{aligned}
$$

Problem 5.2

$$
\begin{aligned}
\min \quad & f(x) := 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \\
s.t. \quad & 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \\
& 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0 \\
& 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0 \\
& -8x_1 + x_{10} \leq 0 \\
& -8x_2 + x_{11} \leq 0 \\
& -8x_3 + x_{12} \leq 0 \\
& -2x_4 - x_5 + x_{10} \leq 0 \\
& -2x_6 - x_7 + x_{11} \leq 0 \\
& -2x_8 - x_9 + x_{12} \leq 0 \\
& x_i \geq 0,\ i = 1, \cdots, 13 \\
& x_i \leq 1,\ i = 1, \cdots, 9, 13 \\
& x_i \leq 100,\ i = 10, \cdots, 12.
\end{aligned}
$$

Problem 5.3

$$\begin{aligned}
\min \quad & f(x) := x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + ... \\
& 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + ... \\
& 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45; \\
s.t. \quad & 4x_1 + 5x_2 - 3x_7 + 9x_8 - 105 \leq 0 \\
& 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0 \\
& -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0 \\
& 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0 \\
& 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0 \\
& 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0 \\
& x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0 \\
& -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0 \\
& -10 \leq x_i \leq 10, \ i = 1, \cdots, 10.
\end{aligned}$$

Problem 5.4

$$\begin{aligned}
\min \quad & f(x) := (x_1 - 10)^3 + (x_2 - 20)^3 \\
s.t. \quad & -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0 \\
& (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0 \\
& 13 \leq x_1 \leq 100, 0 \leq x_2 \leq 100.
\end{aligned}$$

Problem 5.5

$$
\begin{aligned}
\min \quad & f(x) := (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + ... \\
& 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7; \\
s.t. \quad & v1 + 3v2^2 + x_3 + 4x_4^2 + 5x_5 - 127 \le 0 \\
& 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 - 282 \le 0 \\
& 23x_1 + v2 + 6x_6^2 - 8x_7 - 196 \le 0 \\
& 2v1 + v2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \le 0 \\
& -10 \le x_i \le 10,\ i = 1, \cdots, 7 \\
\text{where} \quad & v1 = 2x_1^2,\ v2 = x_2^2.
\end{aligned}
$$

Problem 5.6

$$\begin{aligned}
\min \quad & f(x) := -25(x_1-2)^2 - (x_2-2)^2 - (x_3-1)^2 \\
& \qquad -(x_4-4)^2 - (x_5-1)^2 - (x_6-4)^2 \\
s.t. \quad & (x_3-3)^2 + x_4 \geq 4 \\
& (x_5-3)^2 + x_6 \geq 4 \\
& x_1 - 3x_2 \leq 2 \\
& -x_1 + x_2 \leq 2 \\
& x_1 + x_2 \leq 6 \\
& x_1 + x_2 \geq 2 \\
& 0 \leq x_1 \leq 6 \\
& 0 \leq x_2 \leq 8 \\
& 1 \leq x_3 \leq 5 \\
& 0 \leq x_4 \leq 6 \\
& 1 \leq x_5 \leq 5 \\
& 0 \leq x_6 \leq 10.
\end{aligned}$$

Problem 5.7

$$
\begin{aligned}
\min \quad & f(x) := 37.293239x_1 + 0.8356891x_1x_5 + 5.3578547x_3^2 - 40792.141 \\
s.t. \quad & -0.0022053x_3x_5 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 6.665593 \leq 0 \\
& 0.0022053x_3x_5 - 0.0056858x_2x_5 - 0.0006262x_1x_4 - 85.334407 \leq 0 \\
& 0.0071317x_2x_5 + 0.0021813x_3^2 + 0.0029955x_1x_2 - 29.48751 \leq 0 \\
& -0.0071317x_2x_5 - 0.0021813x_3^2 - 0.0029955x_1x_2 + 9.48751 \leq 0 \\
& 0.0047026x_3x_5 + 0.0019085x_3x_4 + 0.0012547x_1x_3 - 15.699039 \leq 0 \\
& -0.0047026x_3x_5 - 0.0019085x_3x_4 - 0.0012547x_1x_3 + 10.699039 \leq 0 \\
& 78 \leq x_1 \leq 102 \\
& 33 \leq x_2 \leq 45 \\
& 27 \leq x_3 \leq 45 \\
& 27 \leq x_4 \leq 45 \\
& 27 \leq x_5 \leq 45.
\end{aligned}
$$

Problem 5.8

$$
\begin{aligned}
\min \quad & f(x) := -x - y \\
s.t. \quad & y \leq 2x^4 - 8x^3 + 8x^2 + 2 \\
& y \leq 4x^4 - 32x^3 + 88x^2 - 96x + 36 \\
& 0 \leq x \leq 3 \\
& 0 \leq y \leq 4.
\end{aligned}
$$

Problem 5.9

$$\begin{array}{rl} \min & f(x) := x_1 + x_2 + x_3 \\ s.t. & -1 + 0.0025(x_4 + x_6) \leq 0 \\ & -1 + 0.0025(-x_4 + x_5 + x_7) \leq 0 \\ & -1 + 0.01(-x_5 + x_8) \leq 0 \\ & 100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 \leq 0 \\ & x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \leq 0 \\ & x_3x_5 - x_3x_8 - 2500x_5 + 1250000 \leq 0 \\ & l_i \leq x_i \leq u_i,\ i = 1, \cdots, 8 \\ \text{where} & l = 10 \times (10, 100, 100, 1, 1, 1, 1, 1) \\ & u = 1000 \times (10, 10, 10, 1, 1, 1, 1, 1). \end{array}$$

Problem 5.10

$$\begin{array}{rl} \min & \sum_{1 \leq i < j < k \leq n} (i + j)x_ix_jx_k + (j + k)x_i^2x_j^2x_k^2 \\ s.t. & x_1^4 + \cdots + x_n^4 \leq 1 \\ \text{where} & n = 15. \end{array}$$

Problem 5.11

$$\begin{array}{rl} \min & \sum_{1 \leq i < j < k \leq n} x_ix_jx_k(1 + x_i + x_j + x_k) + ix_i^6 + jx_j^6 + kx_k^6 \\ s.t. & x_1^4 + \cdots + x_{\frac{n}{2}}^4 \leq 1 \\ s.t. & x_{\frac{n}{2}+1}^4 + \cdots + x_n^4 \leq 1 \\ \text{where} & n = 16. \end{array}$$

Problem 5.12

$$
\begin{aligned}
\min \quad & \sum_{1\le i<j<k\le \frac{n}{2}} i x_i x_j x_k + j x_{\frac{n}{2}+i} x_{\frac{n}{2}+j} x_{\frac{n}{2}+k} + k x_i x_j x_k x_{\frac{n}{2}+i} x_{\frac{n}{2}+j} x_{\frac{n}{2}+k} \\
s.t. \quad & x_1^4 + \cdots + x_{\frac{n}{2}}^4 \le 1 \\
& x_{\frac{n}{2}+1}^4 + \cdots + x_n^4 \le 1 \\
\text{where} \quad & n = 20.
\end{aligned}
$$

Problem 5.13

$$
\begin{aligned}
\min \quad & f(x) := x_1^2 + (x_2 - 1)^2 \\
s.t. \quad & x_2 - x_1^2 = 0 \\
& -1 \le x_i \le 1,\ i = 1, 2.
\end{aligned}
$$

Problem 5.14

$$
\begin{aligned}
\min \quad & f(x) := -12x_1 - 7x_2 + x_2^2 \\
s.t. \quad & -2x_1^4 + 2 - x_2 = 0 \\
& 0 \le x_1 \le 2,\ 0 \le x_2 \le 3.
\end{aligned}
$$

Problem 5.15

$$
\begin{aligned}
\min \quad & f(x) := (\sqrt{n})^n \prod_{i=1}^{n} x_i \\
s.t. \quad & \sum_{i=1}^{n} x_i^2 - 1 = 0 \\
& 0 \le x_i \le 1,\ i = 1, \cdots, n \\
\text{where} \quad & n = 20.
\end{aligned}
$$

Appendix C.

Nonlinear systems of polynomial equations

Problem EQ6.1: Himmelblau function

$$\begin{aligned} 4x_1^3 + 4x_1x_2 + 2x_2^2 - 42x_1 &= 14 \\ 4x_2^3 + 2x_1^2 + 4x_1x_2 - 26x_2 &= 22 \end{aligned}$$

$$-5 \leq x_1, x_2 \leq 5$$

Problem EQ6.2: Equilibrium Combustion

$$\begin{aligned}
x_1x_2 + x_1 - 3x_5 &= 0 \\
2x_1x_2 + x_1 + 3R_{10}x_2^2 + x_2x_3^2 + R_7x_2x_3 + & \\
R_9x_2x_4 + R_8x_2 - Rx_5 &= 0 \\
2x_2x_3^2 + R_7x_2x_3 + 2R_5x_3^2 + R_6x_3 - 8x_5 &= 0 \\
R_9x_2x_4 + 2x_4^2 - 4Rx_5 &= 0 \\
x_1x_2 + x_1 + R_{10}x_2^2 + x_2x_3^2 + R_7x_2x_3 + R_9x_2x_4 + & \\
R_8x_2 + R_5x_3^2 + R_6x_3 + x_4^2 &= 1 \\
0.0001 \le x_i \le 100,\ i = 1, \cdots, 5 &
\end{aligned}$$

Where $R = 10$, $R_5 = 0.193$, $R_6 = 4.10622\ 10^{-4}$, $R_7 = 5.45177\ 10^{-4}$, $R_8 = 4.4975\ 10^{-7}$, $R_9 = 3.40735\ 10^{-5}$, $R_{10} = 9.615\ 10^{-7}$.

Problem EQ6.3

$$\begin{aligned}
2x_1 + x_2 + x_3 + x_4 + x_5 &= 6 \\
x_1 + 2x_2 + x_3 + x_4 + x_5 &= 6 \\
x_1 + x_2 + 2x_3 + x_4 + x_5 &= 6 \\
x_1 + x_2 + x_3 + 2x_4 + x_5 &= 6 \\
x_1x_2x_3x_4x_5 &= 1 \\
-2 \le x_i \le 2,\ i = 1, \cdots, 5 &
\end{aligned}$$

Problem EQ6.4

$$\begin{aligned}
4.731\ 10^{-3}x_1x_3 - 0.3578x_2x_3 - 0.1238x_1 + x_7 - \\
1.637\ 10^{-3}x_2 - 0.9338x_4 - 0.3571 &= 0 \\
0.2238x_1x_3 + 0.7623x_2x_3 + 0.2638x_1 - x_7 - \\
0.07745x_2 - 0.6734x_4 - 0.6022 &= 0 \\
x_6x_8 + 0.3578x_1 + 4.731\ 10^{-3}x_2 &= 0 \\
-0.7623x_1 + 0.2238x_2 + 0.3461 &= 0 \\
x_1^2 + x_2^2 - 1 &= 0 \\
x_3^2 + x_4^2 - 1 &= 0 \\
x_5^2 + x_6^2 - 1 &= 0 \\
x_7^2 + x_8^2 - 1 &= 0 \\
-1 \leq x_i \leq 1,\ i = 1, \cdots, 8
\end{aligned}$$

Appendix D.

Test problems for nonlinear programming problems

Problem 6.1: Branin Function

$$\begin{aligned}
\min \quad & f(x) := (x_2 - \frac{5}{4\pi^2 x_1^2} + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})cos(x_1) + 10 \\
s.t. \quad & -5 \leq x_1 \leq 10, 0 \leq x_2 \leq 15.
\end{aligned}$$

Problem 6.2: Bohachevsky Function 1

$$\begin{aligned}
\min \quad & f(x) := x_1^2 + 2x_2^2 - 0.3cos(3\pi x_1) - 0.4cos(4\pi x_2) + 0.7 \\
s.t. \quad & -100 \leq x_i \leq 100,\ i = 1, 2.
\end{aligned}$$

Problem 6.3: Bohachevsky Function 2

$$\begin{aligned}
\min \quad & f(x) := x_1^2 + 2x_2^2 - 0.3cos(3\pi x_1)cos(4\pi x_2) + 0.3 \\
s.t. \quad & -100 \leq x_i \leq 100,\ i = 1, 2.
\end{aligned}$$

Problem 6.4: Bohachevsky Function 3

$$\begin{aligned} \min \quad & f(x) := x_1^2 + 2x_2^2 - 0.3cos(3\pi x_1 + 4\pi x_2) + 0.3 \\ s.t. \quad & -100 \leq x_i \leq 100,\ i = 1, 2. \end{aligned}$$

Problem 6.5: Easom Function

$$\begin{aligned} \min \quad & f(x) := -cos(x_1)cos(x_2)exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2) \\ s.t. \quad & -100 \leq x_i \leq 100,\ i = 1, 2. \end{aligned}$$

Problem 6.6: Michalewics Function

$$\begin{aligned} \min \quad & f(x) := -\sum_{i=1}^{n} sin(x_i)sin^{2m}(\frac{ix_i^2}{\pi}) \\ s.t. \quad & 0 \leq x_i \leq \pi,\ i = 1, 2. \\ \text{where} \quad & m = 10. \end{aligned}$$

Problem 6.7: Shubert Function

$$\begin{aligned} \min \quad & f(x) := (\sum_{i=1}^{5} icos((i+1)x_1 + i))(\sum_{i=1}^{5} icos((i+1)x_2 + i) \\ s.t. \quad & -5.12 \leq x_i \leq 5.12,\ i = 1, 2. \end{aligned}$$

Problem 6.8: Schwefel Function

$$\begin{aligned} \min \quad & f(x) := 418.9829n - \sum_{i=1}^{n} x_i sin(\sqrt{|x_i|}) \\ s.t. \quad & -500 \leq x_i \leq 500,\ i = 1, 2. \end{aligned}$$

Problem 6.9: Hartmann(3,4) Function

$$\begin{aligned} \min \quad & f(x) := -\sum_{i=1}^{4} \alpha_i exp(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2) \\ s.t. \quad & 0 < x_i < 1,\ i = 1, 2, 3. \\ \text{where} \quad & \alpha = [1.0, 1.2, 3.0, 3.2]^T \end{aligned}$$

$$A = \begin{bmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 36 \end{bmatrix}$$

,

$$P = 10^{-4} \begin{bmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix}$$

Problem 6.10: Shekel Function

$$\begin{aligned} \min \quad & f(x) := -\sum_{i=1}^{m} (\sum_{j=1}^{4} (x_j - C_{ji})^2 + \beta_i)^{-1} \\ s.t. \quad & 0 \le x_i \le 10,\ i = 1, 2, 3, 4. \\ \text{where} \quad & \beta = \frac{1}{10}[1, 2, 2, 4, 4, 6, 3, 7, 5, 5]^T,\ m = 5 \end{aligned}$$

$$C = \begin{bmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.0 \end{bmatrix}$$

Problem 6.11: Shekel Function

$$\begin{aligned} \min \quad & f(x) := -\sum_{i=1}^{m}(\sum_{j=1}^{4}(x_j - C_{ji})^2 + \beta_i)^{-1} \\ s.t. \quad & 0 \leq x_i \leq 10,\ i = 1, 2, 3, 4. \\ \text{where} \quad & \beta = \frac{1}{10}[1, 2, 2, 4, 4, 6, 3, 7, 5, 5]^T,\ m = 10 \end{aligned}$$

$$C = \begin{bmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.0 \end{bmatrix}$$

Problem 6.12: Hartmann(6,4) Function

$$\begin{aligned} \min \quad & f(x) := -\sum_{i=1}^{4} \alpha_i exp(-\sum_{j=1}^{6} A_{ij}(x_j - P_{ij})^2) \\ s.t. \quad & 0 < x_i < 1,\ i = 1, 2, \cdots, 6. \\ \text{where} \quad & \alpha = [1.0, 1.2, 3.0, 3.2]^T \end{aligned}$$

$$A = \begin{bmatrix} 10 & 3 & 17 & 3.50 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$$

$$P = 10^{-4} \begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}$$

Problem 6.13: Schwefel Function

$$\begin{aligned} \min \quad & f(x) := 418.9829n - \sum_{i=1}^{n} x_i sin(\sqrt{|x_i|}) \\ s.t. \quad & -500 \leq x_i \leq 500,\ i = 1, \cdots, 6. \end{aligned}$$

Problem 6.14: Michalewics Function

$$\begin{aligned} \min \quad & f(x) := -\sum_{i=1}^{n} sin(x_i) sin^{2m}(\frac{ix_i^2}{\pi}) \\ s.t. \quad & 0 \leq x_i \leq \pi,\ i = 1, \cdots, 10. \end{aligned}$$

$$\text{where} \quad m = 10.$$

Problem 6.15: Rastrigin Function

$$\begin{aligned} \min \quad & f(x) := 10n + \sum_{i=1}^{n} (x_i^2 - 10cos(2\pi x_i)) \\ s.t. \quad & -5.12 \leq x_i \leq 5.12,\ i = 1, \cdots, 10. \end{aligned}$$

Problem 6.16: Griewank Function

$$\begin{aligned} \min \quad & f(x) := \sum_{i=1}^{n} \frac{x_i^2}{4000} - \sum_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1 \\ s.t. \quad & -600 \leq x_i \leq 600,\ i = 1, 2, \cdots, 10. \end{aligned}$$

Problem 6.17: Rastrigin Function

$$\begin{aligned} \min \quad & f(x) := 10n + \sum_{i=1}^{n} (x_i^2 - 10cos(2\pi x_i)) \\ s.t. \quad & -5.12 \leq x_i \leq 5.12,\ i = 1, \cdots, 20. \end{aligned}$$

Problem 6.18: Griewank Function

$$\begin{aligned} \min \quad & f(x) := \sum_{i=1}^{n} \frac{x_i^2}{4000} - \sum_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1 \\ s.t. \quad & -600 \leq x_i \leq 600,\ i = 1, 2, \cdots, 20. \end{aligned}$$

Problem 6.19: Levy Function

$$\begin{aligned} \min \quad & f(x) := sin^2(\pi y_1) + \sum_{i=1}^{k-1} (y_i - 1)^2 (1 + 10 sin^2(\pi y_i + 1)) \\ & +(y_k - 1)^2 (1 + sin^2(2\pi x_k)) \\ s.t. \quad & y_i = 1 + \frac{x_i - 1}{4},\ i = 1, 2, \cdots, 30, \\ & -10 \leq x_i \leq 10,\ i = 1, 2, \cdots, 30. \end{aligned}$$

Problem 6.20: Ackley Function

$$\begin{aligned} \min \quad & f(x) := 20 + e - 20 e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)} \\ s.t. \quad & -15 \leq x_i \leq 30,\ i = 1, 2, \cdots, 30. \end{aligned}$$