Journal of Computational Mathematics
Vol.xx, No.x, 2023, 1–30.

http://www.global-sci.org/jcm
doi:10.4208/jcm.2210-m2021-0067

A NEW GLOBAL OPTIMIZATION ALGORITHM FOR MIXED-INTEGER QUADRATICALLY CONSTRAINED QUADRATIC FRACTIONAL PROGRAMMING PROBLEM*

Bo Zhang
School of Civil and Hydraulic Engineering, Ningxia University, Yinchuan 750021, China
School of Mathematics and Statistics, Ningxia University, Yinchuan 750021, China
Email: zbsdx121@163.com
Yuelin Gao[1)]
School of Mathematics and Information Science, North Minzu University, Yinchuan 750021, China
Ningxia Province Key Laboratory of Intelligent Information and Data Processing, Yinchuan 750021, China
Email: gaoyuelin@263.net
Xia Liu
School of Mathematics and Statistics, Ningxia University, Yinchuan 750021, China
Email: lingxiaoyu911@163.com
Xiaoli Huang
School of Mathematics and Statistics, Ningxia University, Yinchuan 750021, China
Email: hxl1569501@163.com

Abstract

The mixed-integer quadratically constrained quadratic fractional programming (MIQCQFP) problem often appears in various fields such as engineering practice, management science and network communication. However, most of the solutions to such problems are often designed for their unique circumstances. This paper puts forward a new global optimization algorithm for solving the problem MIQCQFP. We first convert the MIQCQFP into an equivalent generalized bilinear fractional programming (EIGBFP) problem with integer variables. Secondly, we linearly underestimate and linearly overestimate the quadratic functions in the numerator and the denominator respectively, and then give a linear fractional relaxation technique for EIGBFP on the basis of non-negative numerator. After that, combining rectangular adjustment-segmentation technique and midpoint-sampling strategy with the branch-and-bound procedure, an efficient algorithm for solving MIQCQFP globally is proposed. Finally, a series of test problems are given to illustrate the effectiveness, feasibility and other performance of this algorithm.

Mathematics subject classification: 90C57, 90C26.
Key words: Global optimization, Branch and bound, Quadratic fractional programming, Mixed integer programming.

1. Introduction

Consider the following class of mixed integer quadratically constrained quadratic fractional programming (MIQCQFP) problems:

* Received March 2, 2021 / Revised version received July 6, 2022 / Accepted October 24, 2022 / Published online May 29, 2023 /
1) Corresponding author

$$\text{(MIQCQFP)} \quad \begin{cases} \min f(x) = \dfrac{f_0(x)}{g(x)}, & \\ \text{s.t. } f_i(x) \le 0, & i = 1, 2, \ldots, N, \\ x \in X = \{x \in \mathbb{R}^n | Ax \le b\}, & \\ x_j \in \mathbb{Z}, & j \in N_I \subseteq \{1, 2, \ldots, n\}, \end{cases}$$

where $f_i(x) = x^T Q_i x + c_i^T x + d_i, i = 0, 1, 2, \ldots, N$ with

$$\begin{aligned} &Q_i \in \mathbb{R}^{n\times n}, \quad i = 0, 1, 2, \ldots, N, \\ &c_i \in \mathbb{R}^n, \qquad i = 0, 1, 2, \ldots, N, \\ &d_i \in \mathbb{R}, \qquad i = 0, 1, 2, \ldots, N, \end{aligned}$$

$g(x) = x^T Hx + h^T x + q$ with $H \in \mathbb{R}^{n\times n}$, $h \in \mathbb{R}^n$, $q \in \mathbb{R}$ and X is a nonempty bounded set with $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$; T denotes the transpose of a vector (or matrix) (for example, c_i^T is the transpose of a vector c_i); there is no convex (or concave) assumption for all functions; note that all forms of inequality constraints can contain all forms of constraints, because any equality constraints can be replaced by two inequality constraints; $\mathbb{Z}$ denotes a set of all integers; N_I is an integer index set. There will always be $g(x) > 0$ or $g(x) < 0$ for the denominator $g(x)$ in the objective function according to its continuity and nonzero property. If $g(x) < 0$ for some $x \in X$, we can make

$$\frac{f_0(x)}{g(x)} = \frac{-f_0(x)}{-g(x)}.$$

Then the original problem keeps unaltered, and the denominator becomes positive. In addition, if there is an $x \in X$ such that the numerator $f_0(x) < 0$, we can construct $f_0(x) + Mg(x) \ge 0$ with a sufficiently large positive number M, then

$$\frac{f_0(x)}{g(x)} = \frac{f_0(x) + Mg(x)}{g(x)} - M,$$

obviously, these two problems

$$\min_{x\in X} \frac{f_0(x)}{g(x)} \quad \text{and} \quad \min_{x\in X} \frac{f_0(x) + Mg(x)}{g(x)}$$

share the same solution. Therefore, throughout this paper, without loss of generality, we assume $f_0(x) \ge 0$ for any $x \in X$.

Problem MIQCQFP and its special cases are ubiquitous in the real world. From a computational point of view, the MIQCQFP may be difficult to solve because it has three kinds of non-convexity, one is the possible integer variable, one is the non-convex quadratic term in the objective function and/or constraint function, and the other is the fractional form of the objective function. Numerous variants of this problem are also divided into several categories according to the form of their objective functions or constraints. For example, when the constraints of MIQCQFP are only linear, Bomze and Amaral [3] pointed out that such problems arise from many application problems, such as optimizing communication or social networks, or studying game theory problems caused by genetics; they include several APX-hard subclasses: the maximum cut problem, k-densest subgraph problem and several variants thereof, or ternary fractional quadratic optimization problem (TFQP); also, they add rich evidence of

common positive optimization methods and reveal possible new approximation strategies combining continuous optimization and discrete optimization techniques in the field of (fractional) polynomial optimization. Generally speaking, the MIQCQFP can be transformed into a typical quadratically constrained quadratic programming (QCQP) problem in the case of $g(x) = 1$ and without requiring partial variables to be integers, which often occurs in many applications, such as the Celis-Dennis-Tapia problem [5], box constrained quadratic programming problem [10], generalized trust-region problem [17], standard quadratic programming problem [4], are subclasses of MIQCQFP problems. The problem QCQP is NP-Hard in general, it usually has multiple local optimal solutions, and finding its local optimal solutions is also NP-Hard [29], and only a few special subclass problems with QCQP can be determined as polynomial solvable [21]. Additionally, both the sum-of-linear fractions programming (SLFP) problem [19, 22, 23, 35] and the min-max (max-min) linear fractional programming(MLFP) problem [18, 30, 47] can be converted into variations of QCQP respectively. Of course, we should also point out that a class of generalized linear multiplicative programming (LMP) problems can also be rewritten as QCQP problems (for details, see [34, 36]). Furthermore, if some (or all) of the variables of the QCQP are required to be integers, the mixed integer quadratically constrained quadratic programming (MIQCQP) problem can be formed, which also belongs to a subclass of MIQCQFP. Also, the QCQP is a subclass of MIQCQP. For solving the problem MIQCQP, Zhao and Liu [46] and Gao and Wei [11] proposed a series of branch-and-bound algorithms, all of which adopted equivalent transformation to construct the relaxation programming subproblem to obtain the optimal solution. Pai and Alberto [7] developed an approximate algorithm, which can find ϵ-approximate solutions to the MIQCQP problem with concave quadratic objective function, and the running time of this algorithm is polynomial in the size of the problem and in $1/\epsilon$, as long as the number of integer variables and negative eigenvalues of the objective function are fixed. More detailed understanding of problem MIQCQP (or QCQP) and their solutions, we encourage readers to refer to [14, 42]. When the elements of all the matrices in the problem MIQCQFP are zero and the integer constraints are thrown away, MIQCQFP will be reduced to a simple linear fractional programming (LFP) problem. Charnes and Cooper [6], Swarup [39] and Ozkok [27] respectively gave different methods for solving LFP. In the case that some variables in LFP must be integers, we call it the mixed integer linear fractional programming (MILFP) problem. The global solution of problem MILFP may be computationally intractable due to the existence of discrete variables and pseudo-convex (non-convex) objective functions. Yue *et al.* [44] combined the Glover linearization scheme [13] with the Charnes-Cooper transformation, and thus proposed a new and effective reconstruction computation-linearization method. Seerengasamy and Jeyaraman [33] proposed a simple optimization method to solve the MILFP by using θ-matrix. If all integer constraints in MIQCQFP are removed, the problem is called quadratically constrained quadratic fractional programming (QCQFP) problem. The main difficulty in solving the problem QCQFP is its non-convexity [45]. At present, QCQFP has been investigated and solved by many scholars mainly because of its widespread application, including signal processing, communication, financial analysis, position theory, portfolio selection and random decision [24, 26, 41]. Such problems are usually difficult to deal with, but several methods based on semi-definite programming relaxation (SDPR) [1, 9, 25] have been successfully used to address some special variants of QCQFP. Moreover, to solve the QCQFP problem with linear constraints, Sivri *et al.* [37] proposed a method based on Taylor series, which can transform the quadratic fractional programming problem, involving factorization or non-factor in the objective function, into a linear programming problem. For a quadratic fractional pro-

gramming problem with a single homogeneous quadratic constraint, Beck and Teboulle [1], under a certain mild assumption on the problem's data, determined an accurate SDPR for this problem and designed a simple iterative process, which was proved to be able to obtain the global solution of the problem with superlinear convergence. Besides, Fallahi and Salahi [9] considered the quadratic fractional programming problem with two quadratic constraints, and utilized the generalized Charnes-Cooper transformation to convert the problem into a homogeneous quadratic problem, and then, under certain assumptions, it is proved that this problem can be solved globally by a SDPR method. Nguyen *et al.* [25] proposed a SDPR method for a quadratic fractional programming problem with two-sided quadratic constraints. For quadratic fractional programming problems with several homogeneous quadratic constraints, Khurana and Arora [20] constructed the transformation matrix $\mathcal{T}$ through these homogeneous constraints and transformed the given problem into another quadratic fractional programming with fewer constraints, thus establishing the relationship between the original problem and the transformed problem. Beck and Teboulle [2] also investigated the quadratic fractional programming problem with a finite number of non-convex quadratic inequality constraints based on the homogenization technique, and established a sufficient condition to guarantee the realization of the optimal solution. Ibaraki *et al.* [15] presented two algorithms for solving the quadratic fractional programming problem, one is the parameter programming technique based on quadratic programming, the other is Dinkelbach-based method [8]. Suleimann and Nawkhass [38] improved the simplex method by using the idea of Wolfe methods, which were utilized to solve the problem QCQFP. Tantawy [40] studied the feasible direction method, which also solved the QCQFP. Yamamoto and Konno [43] combined the integer programming method for solving non-convex quadratic programming problems in classical Dinkelbach-method with standard nonlinear programming algorithms, and then proposed a new method that can solve a variant problem (i.e., convex-convex quadratic fractional programming problem) of the QCQFP. To solve a class of non-convex fractional minimization problems whose feasible region is the intersection of the unit sphere and a linear inequality constraint, Salahi and Fallahi [32] combined Dinkelbach's idea with diagonalization of matrix, which provided an effective parameterization method for solving the original problem with generalized Newton iterative algorithm. In short, under some technical conditions, the Dinkelbach iterative method can always be used to solve QCQFP, which has led to the development of quadratic fractional programming problems for nearly half a century. In many practical situations, people are only interested in integer solutions of QCQFP, and sometimes they need an optimal integer solution that satisfies integer requirements in addition to the master constraint, which requires that all variables in MIQCQFP are integers, thus forming the so-called integer quadratic fractional programming problem. Until now, only Jain *et al.* [16] had studied quadratic fractional programming problems with linear constraints, and have given a sorting and scanning method that introduces various cut sets in the intermediate steps; however, all integer solutions derived from the base-feasible solutions in this way may not be optimal. It can be seen that most of the above literature can only solve the subclass problems of MIQCQFP, or only give the related algorithm theories of some subclass problems. As far as we know, although some significant progress has been achieved in the theoretical development of solving continuous quadratic fractional programming in addition to the above introduction, there is little work on directly solving the broader problem MIQCQFP globally.

In this paper, we develop a new branch-and-bound algorithm to solve the problem MIQCQFP globally. Firstly, we reconstruct MIQCQFP as the EIGBFP problem. Secondly, in order

to obtain the lower bound of EIGBFP, a unique linear fractional relaxation method is proposed, and a linear fractional relaxation programming subproblem is established for EIGBFP. Thirdly, we add the boundary adjustment strategy and feasibility-based detection method to the traditional branch-and-bound algorithm to better adapt to the implementation of the algorithm. Fourthly, to demonstrate the effectiveness of our algorithm, we did not only solve a total of 18 examples in the existing literature for the four variants (SLFP, MLFP, LMP and MIQCQP) of MIQCQFP, but also constructed and solved 14 additional test problems for MIQCQFP. Fifthly, when solving a series of random LMP problems, the proposed new linear fractional relaxation technique for computing lower bounds is more tighter than the linear relaxation technique of [36] at some cases. Numerical results in Table 4.8 demonstrate that our algorithm may be more efficient than that of [36] when solving LMP problems with p (the number of linear products) greater than or close to n (the number of variables). Besides, the proposed algorithm performs better than software packages SCIP [12] and BARON [31] in solving LMP problems with certain specific cases (see the last three sets of numerical results in Table 4.8). Finally, we have carried out some small and medium-sized numerical experiments on a class of integer-quadratic fractional programming problem 16 with several linear constraints and two quadratic constraints. Experimental results in Table 4.9 illustrate that our algorithm can effectively address this problem within 700 dimensions in less than 1 hour, and has a higher computational efficiency than SCIP and BARON when solving some specific medium-scale problems.

This paper is arranged as follows. In Section 2, the problem MIQCQFP is firstly reformulated into its equivalent generalized bilinear fractional programming (EIGBFP) problem with integer variables, and then a linear fractional relaxation technique for constructing a new relaxation subproblem is proposed. Furthermore, the linear fractional relaxation subproblem is converted into a linear program, which results in the fact that the global optimal solution of MIQCQP can be approximated by addressing a series of linear programming subproblems. In Section 3, the boundary adjustment-segmentation rule of rectangles and the feasibility-based detection method of solutions are given, and then the concrete steps of the branch-and-bound algorithm are described in detail and its convergence is proved. Section 4 consists of a series of numerical experiments and their corresponding numerical results analysis, which are used to explain the performance of the proposed algorithm. Section 5 is the conclusion.

2. Reconstructed Problem and Its Linear Fractional Relaxation Technique

In this section, we will reconstruct the MIQCQFP problem and give its linear fractional relaxation technique.

2.1. Reconstructed problem

First, the vector $H_j \in \mathbb{R}^{1\times n}, j = 1, 2, \ldots, n$, is denoted as the j-th row of the matrix H and for each $i = 0, 1, 2, \ldots, N$, let the j-th, $j = 1, 2, \ldots, n$, row of the matrix Q_i be $Q_{ij} \in \mathbb{R}^{1\times n}$. Then, we have

$$x^T Hx = (x_1, x_2, \cdots, x_n)(H_1x, H_2x, \cdots, H_nx)^T = \sum_{j=1}^{n} x_j H_j x,$$

$$x^T Q_i x = (x_1, x_2, \cdots, x_n)(Q_{i1}x, Q_{i2}x, \cdots, Q_{in}x)^T = \sum_{j=1}^{n} x_j Q_{ij} x.$$

Thus,

$$f_i(x) = x^T Q_i x + c_i^T x + d_i = \sum_{j=1}^{n} x_j Q_{ij} x + c_i^T x + d_i,$$
$$g(x) = x^T H x + h^T x + q = \sum_{j=1}^{n} x_j H_j x + h^T x + q.$$

Then, MIQCQFP is further expressed as the following optimization problem:

$$\text{(EIGBFP)} \quad \begin{cases} \min f(x) = \dfrac{\sum_{j=1}^{n} x_j Q_{0j} x + c_0^T x + d_0}{\sum_{j=1}^{n} x_j H_j x + h^T x + q}, \\ \text{s.t. } f_i(x) = \sum_{j=1}^{n} x_j Q_{ij} x + c_i^T x + d_i \le 0, \quad i = 1, 2, \ldots, N, \\ x \in X = \{x \in \mathbb{R}^n | Ax \le b\}, \quad x_j \in \mathbb{Z}, \qquad j \in N_I \subseteq \{1, 2, \ldots, n\}. \end{cases}$$

Obviously, problems EIGBFP and MIQCQFP are equivalent and have the same global optimal solution and optimal value. Thus, in order to obtain the global optimal solution of MIQCQFP, we will focus on the EIGBFP problem.

2.2. Linear fractional relaxation problem

To draw up a branch-and-bound algorithm that can solve the EIGBFP problem globally, a linear fractional relaxation problem needs to be established, which provides an effective lower bound for the optimal value of EIGBFP. For this reason, a two-stage relaxation technique will be utilized here. The first stage is called continuous relaxation, that is, all integer constraints in the EIGBFP problem are directly removed, while the specific operation of the second stage is as follows.

First, we must solve the following linear programming problems:

$$\underline{x}_j^0 = \min_{x \in X} x_j, \quad \overline{x}_j^0 = \max_{x \in X} x_j, \quad j = 1, 2, \ldots, n.$$

Then, the initial hyper-rectangle

$$D^0 = \left\{ x \in \mathbb{R}^n \mid \underline{x}_j^0 \le x_j \le \overline{x}_j^0, j = 1, 2, \ldots, n \right\},$$

which contains the feasible domain of problem EIGBFP (MIQCQFP), is naturally established.

Suppose $D = [\underline{x}, \overline{x}]$ denotes the initial rectangle D^0 or a sub-rectangle of D^0 improved by the algorithm. For each $i = 0, 1, 2, \ldots, N$, $j = 1, 2, \ldots, n$, let $z_j = H_j x$, $y_{ij} = Q_{ij} x$, then we further do the following over D. Let

$$\underline{y}_{ij} = \sum_{k=1}^{n} \min\{Q_{ijk}\underline{x}_k, Q_{ijk}\overline{x}_k\}, \quad \overline{y}_{ij} = \sum_{k=1}^{n} \max\{Q_{ijk}\underline{x}_k, Q_{ijk}\overline{x}_k\}. \tag{2.1}$$

Similarly, for each $j = 1, 2, \ldots, n$, we also have

$$\underline{z}_j = \sum_{k=1}^{n} \min\{H_{jk}\underline{x}_k, H_{jk}\overline{x}_k\}, \quad \overline{z}_j = \sum_{k=1}^{n} \max\{H_{jk}\underline{x}_k, H_{jk}\overline{x}_k\}. \tag{2.2}$$

Then, by using formulas (2.1) and (2.2), the following two hyper-rectangles:

$$H_y = [\underline{y}, \overline{y}] = \left\{ y \in \mathbb{R}^{n(N+1)} \mid \underline{y}_{ij} \le y_{ij} \le \overline{y}_{ij},\, i = 0, 1, 2, \ldots, N,\, j = 1, 2, \ldots, n \right\},$$

and

$$H_z = [\underline{z}, \overline{z}] = \left\{ z \in \mathbb{R}^n \mid \underline{z}_j \le z_j \le \overline{z}_j,\, j = 1, 2, \ldots, n \right\}$$

are constructed respectively. Through the above definition of H_y and H_z, let

$$\Omega = \{(x, z) : \underline{x} \le x \le \overline{x}, \underline{z} \le z \le \overline{z}\} = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_n,$$

of which

$$\Omega_j = \left\{(x_j, z_j) : \underline{x}_j \le x_j \le \overline{x}_j, \underline{z}_j \le z_j \le \overline{z}_j\right\}, \quad j = 1, 2, \ldots, n.$$

Similarly, let

$$\Theta = \{(x, y) : \underline{x} \le x \le \overline{x}, \underline{y} \le y \le \overline{y}\} = \prod_{j=1}^{n} \Theta_{0j} \times \prod_{j=1}^{n} \Theta_{1j} \times \prod_{j=1}^{n} \Theta_{2j} \times \cdots \times \prod_{j=1}^{n} \Theta_{Nj},$$

where

$$\Theta_{ij} = \left\{(x_j, y_{ij}) : \underline{x}_j \le x_j \le \overline{x}_j, \underline{y}_{ij} \le y_{ij} \le \overline{y}_{ij}\right\}, \quad i = 0, 1, 2, \ldots, N, \quad j = 1, 2, \ldots, n.$$

For any $(x_j, z_j) \in \Omega_j$ with $j = 1, 2, \ldots, n$, we have $\overline{x}_j - x_j \ge 0$, $\underline{z}_j - z_j \le 0$. Let

$$w_j = \overline{x}_j - x_j, \quad v_j = \underline{z}_j - z_j.$$

As a result of

$$w_j - (\overline{x}_j - \underline{x}_j) \le 0, \quad v_j - (\underline{z}_j - \overline{z}_j) \ge 0,$$

there are the following inequalities:

$$\left[w_j - (\overline{x}_j - \underline{x}_j)\right]\left[v_j - (\underline{z}_j - \overline{z}_j)\right] \le 0, \quad j = 1, 2, \ldots, n.$$

Thus, we have

$$w_j v_j \le (\underline{z}_j - \overline{z}_j) w_j + (\overline{x}_j - \underline{x}_j) v_j - (\underline{z}_j - \overline{z}_j)(\overline{x}_j - \underline{x}_j) \triangleq \bar{\vartheta}_j^1(w_j, v_j). \tag{2.3}$$

Upon $w_j \ge 0$ and $v_j \le 0$, it follows that

$$w_j v_j \le 0 \triangleq \bar{\vartheta}_j^2(w_j, v_j). \tag{2.4}$$

Substituting equations $w_j = \overline{x}_j - x_j, v_j = \underline{z}_j - z_j$ into the left end of inequalities (2.3)-(2.4) respectively, we have

$$x_j z_j \le \bar{\vartheta}_j^1(w_j, v_j) + \overline{x}_j z_j + \underline{z}_j x_j - \overline{x}_j \underline{z}_j = \underline{x}_j z_j + \overline{z}_j x_j - \overline{z}_j \underline{x}_j, \tag{2.5}$$

$$x_j z_j \le \bar{\vartheta}_j^2(w_j, v_j) + \overline{x}_j z_j + \underline{z}_j x_j - \overline{x}_j \underline{z}_j = \overline{x}_j z_j + \underline{z}_j x_j - \overline{x}_j \underline{z}_j. \tag{2.6}$$

To continue, we substitute $z_j = H_j x$ into both ends of (2.5)-(2.6) respectively, and obtain

$$x_j H_j x \le \overline{z}_j x_j + \underline{x}_j H_j x - \overline{z}_j \underline{x}_j \triangleq \vartheta_j^1(x), \quad \forall\, x \in D, \tag{2.7}$$

$$x_j H_j x \le \underline{z}_j x_j + \overline{x}_j H_j x - \overline{x}_j \underline{z}_j \triangleq \vartheta_j^2(x), \quad \forall\, x \in D. \tag{2.8}$$

Based on inequalities (2.7) and (2.8), it easily deduced that

$$\begin{aligned} g(x) &= \sum_{j=1}^{n} x_j H_j x + h^T x + q \\ &\le \sum_{j=1}^{n} \min\left\{\vartheta_j^1(x), \vartheta_j^2(x)\right\} + h^T x + q \\ &\le \min\left\{\sum_{j=1}^{n} \vartheta_j^1(x), \sum_{j=1}^{n} \vartheta_j^2(x)\right\} + h^T x + q \\ &= \min\left\{(\alpha_1 + h)^T x + \delta_1 + q, (\alpha_2 + h)^T x + \delta_2 + q\right\} \triangleq \overline{g}(x), \quad \forall x \in D, \end{aligned} \tag{2.9}$$

where

$$\begin{aligned} \alpha_1 &= \sum_{j=1}^{n} \left(\overline{z}_j e_j + \underline{x}_j H_j^T\right), \quad \delta_1 = -\sum_{j=1}^{n} \overline{z}_j \underline{x}_j, \\ \alpha_2 &= \sum_{j=1}^{n} \left(\underline{z}_j e_j + \overline{x}_j H_j^T\right), \quad \delta_2 = -\sum_{j=1}^{n} \underline{z}_j \overline{x}_j. \end{aligned} \tag{2.10}$$

In the above two formulas, the $e_j, j = 1, 2, \ldots, n$, represents a n-dimensional vector with the j-th element 1 and the other elements 0 (the same below).

In addition, for any $(x_i, y_{ij}) \in \Theta_{ij}$, we know $x_j - \overline{x}_j \le 0, y_{ij} - \overline{y}_{ij} \le 0$. Let $s_j = x_j - \overline{x}_j, t_{ij} = y_{ij} - \overline{y}_{ij}$. Upon inequalities $s_j - (\underline{x}_j - \overline{x}_j) \ge 0$ and $t_{ij} - (\underline{y}_{ij} - \overline{y}_{ij}) \ge 0$, it follows that

$$\left[s_j - (\underline{x}_j - \overline{x}_j)\right]\left[t_{ij} - (\underline{y}_{ij} - \overline{y}_{ij})\right] \ge 0, \quad i = 0, 1, 2, \ldots, N, \quad j = 1, 2, \ldots, n.$$

Therefore, there will certainly be

$$s_j t_{ij} \ge (\underline{y}_{ij} - \overline{y}_{ij}) s_j + (\underline{x}_j - \overline{x}_j) t_{ij} - (\underline{x}_j - \overline{x}_j)(\underline{y}_{ij} - \overline{y}_{ij}) \triangleq \bar{\theta}_{ij}^1(s_j, t_{ij}). \tag{2.11}$$

Besides, we can derive

$$s_j t_{ij} \ge 0 \triangleq \bar{\theta}_{ij}^2(s_j, t_{ij}) \tag{2.12}$$

from inequalities $s_j \le 0$, $t_{ij} \le 0$. Now, substituting $s_j = x_j - \overline{x}_j$, $t_{ij} = y_{ij} - \overline{y}_{ij}$ into the left end of inequalities (2.11)-(2.12) respectively, we have

$$x_j y_{ij} \ge \bar{\theta}_{ij}^1(s_j, t_{ij}) + \overline{y}_{ij} x_j + \overline{x}_j y_{ij} - \overline{x}_j \overline{y}_{ij} = \underline{y}_{ij} x_j + \underline{x}_j y_{ij} - \underline{x}_j \underline{y}_{ij}, \tag{2.13}$$

$$x_j y_{ij} \ge \bar{\theta}_{ij}^2(s_j, t_{ij}) + \overline{y}_{ij} x_j + \overline{x}_j y_{ij} - \overline{x}_j \overline{y}_{ij} = \overline{y}_{ij} x_j + \overline{x}_j y_{ij} - \overline{x}_j \overline{y}_{ij}. \tag{2.14}$$

Similarly, the equation $y_{ij} = Q_{ij} x$ is substituted into the two ends of inequalities (2.13)-(2.14), respectively, it easily follows that

$$x_j Q_{ij} x \ge \underline{y}_{ij} x_j + \underline{x}_j Q_{ij} x - \underline{x}_j \underline{y}_{ij} \triangleq \theta_{ij}^1(x), \quad \forall\, x \in D, \tag{2.15}$$

$$x_j Q_{ij} x \ge \overline{y}_{ij} x_j + \overline{x}_j Q_{ij} x - \overline{x}_j \overline{y}_{ij} \triangleq \theta_{ij}^2(x), \quad \forall\, x \in D. \tag{2.16}$$

According to formulas (2.15)-(2.16), $x_j Q_{ij} x \ge \max\{\theta_{ij}^1(x), \theta_{ij}^2(x)\}, \forall x \in D$. Thus, for each $i = 0, 1, 2, \ldots, N$, for any $x \in D$, it is not too difficult to deduce that

$$\begin{aligned} \sum_{j=1}^{n} x_j Q_{ij} x &\ge \sum_{j=1}^{n} \max\left\{\theta_{ij}^1(x), \theta_{ij}^2(x)\right\} \ge \max\left\{\sum_{j=1}^{n} \theta_{ij}^1(x), \sum_{j=1}^{n} \theta_{ij}^2(x)\right\} \\ &= \max\left\{(\beta_1^i)^T x + \varrho_1^i, (\beta_2^i)^T x + \varrho_2^i\right\}, \end{aligned}$$

where

$$\beta_1^i = \sum_{j=1}^{n} \underline{y}_{ij} e_j^T + \underline{x}_j Q_{ij}^T, \quad \varrho_1^i = -\sum_{j=1}^{n} \underline{x}_j \underline{y}_{ij},$$
$$\beta_2^i = \sum_{j=1}^{n} \overline{y}_{ij} e_j^T + \overline{x}_j Q_{ij}^T, \quad \varrho_2^i = -\sum_{j=1}^{n} \overline{x}_j \overline{y}_{ij}.$$

Then,

$$\begin{aligned} f_i(x) &= \sum_{j=1}^{n} x_j Q_{ij} x + c_i^T x + d_i \\ &\geq \max\left\{(\beta_1^i)^T x + \varrho_1^i, (\beta_2^i)^T x + \varrho_2^i\right\} + c_i^T x + d_i \\ &= \max\left\{(\beta_1^i + c_i)^T x + \varrho_1^i + d_i, (\beta_2^i + c_i)^T x + \varrho_2^i + d_i\right\} \\ &\triangleq \underline{f}_i(x), \quad \forall x \in D. \end{aligned} \tag{2.17}$$

Notice that, when $i = 0$, there may also be $x \in X \cap D$ such that $\underline{f}_0(x) < 0$, so we employ the hypothesis $f_0(x) \geq 0$, there will be

$$f_0(x) \geq \max\left\{\underline{f}_0(x), 0\right\} \triangleq f_0^l(x), \quad \forall x \in D. \tag{2.18}$$

Then, the above inequality (2.18) ensures that the underestimation of the numerator function is not less than zero, which reduces some unnecessary search operations in the execution process of the algorithm, and thus improves the computational efficiency. Therefore, it follows from inequalities (2.9) and (2.18) that

$$f(x) = \frac{f_0(x)}{g(x)} \geq \frac{f_0(x)}{\overline{g}(x)} \geq \frac{f_0^l(x)}{\overline{g}(x)} \triangleq \underline{f}(x), \quad \forall x \in X \cap D. \tag{2.19}$$

Next, by inequalities (2.17) and (2.19), we will obtain a linear fractional relaxation programming subproblem for EIGBFP(MIQCQFP) over D

$$(\mathrm{LFRP}_D) \quad \begin{cases} \min \ \dfrac{s}{v}, \\ \text{s.t. } (\beta_1^0 + c_0)^T x + \varrho_1^0 + d_0 \leq s, \\ \quad (\beta_2^0 + c_0)^T x + \varrho_2^0 + d_0 \leq s, \\ \quad (\alpha_1 + h)^T x + \delta_1 + q \geq v, \\ \quad (\alpha_2 + h)^T x + \delta_2 + q \geq v, \\ \quad (\beta_1^i + c_i)^T x + \varrho_1^i + d_i \leq 0, \qquad i = 1, 2, \ldots, N, \\ \quad (\beta_2^i + c_i)^T x + \varrho_2^i + d_i \leq 0, \qquad i = 1, 2, \ldots, N, \\ \quad s \geq 0, \quad v > 0, \quad x \in X \cap D. \end{cases}$$

Theorem 2.1. *The functions $g(x)$, $\overline{g}(x)$, $f_i(x), i = 0, 1, 2, \ldots, N$, $\underline{f}_i(x), i = 0, 1, 2, \ldots, N$, $f(x)$ and $\underline{f}(x)$ defined over a rectangle $D \subseteq D^0$ satisfy the following conditions:*

(i) for any $x \in X \cap D$, $g(x) \leq \overline{g}(x)$, when $\|D\| \to 0$, there will be $|g(x) - \overline{g}(x)| \to 0$;

(ii) for any $x \in X \cap D$, $f_i(x) \geq \underline{f}_i(x)$, when $\|D\| \to 0$, there will be $|f_i(x) - \underline{f}_i(x)| \to 0$;

(iii) for any $x \in X \cap D$, $f(x) \geq \underline{f}(x)$, when $\|D\| \to 0$, there will be $|f(x) - \underline{f}(x)| \to 0$.

Proof.

(i) For any $x \in X \cap D$, according to formulas (2.9) and (2.10), we can easily know that $g(x) \le \overline{g}(x)$ and

$$\begin{aligned}
\big|g(x)-\overline{g}(x)\big| &= \left|\sum_{j=1}^{n} x_j H_j x - \min\left\{\alpha_1^T x+\delta_1, \alpha_2^T x+\delta_2\right\}\right| \\
&= \left|\max\left\{\sum_{j=1}^{n} x_j H_j x - \left[\alpha_1^T x+\delta_1\right], \sum_{j=1}^{n} x_j H_j x - \left[\alpha_2^T x+\delta_2\right]\right\}\right| \\
&= \left|\max\left\{\sum_{j=1}^{n} x_j z_j - \left[\alpha_1^T x+\delta_1\right], \sum_{j=1}^{n} x_j z_j - \left[\alpha_2^T x+\delta_2\right]\right\}\right| \\
&= \left|\max\left\{\sum_{j=1}^{n} (x_j-\underline{x}_j)(z_j-\overline{z}_j), \sum_{j=1}^{n} (x_j-\overline{x}_j)(z_j-\underline{z}_j)\right\}\right| \\
&\le \left|\max\left\{\sum_{j=1}^{n} |\overline{x}_j-\underline{x}_j||\overline{z}_j-\underline{z}_j|, \sum_{j=1}^{n} |\overline{x}_j-\underline{x}_j||\overline{z}_j-\underline{z}_j|\right\}\right| \\
&= \sum_{j=1}^{n} |\overline{x}_j-\underline{x}_j||\overline{z}_j-\underline{z}_j|. \qquad (2.20)
\end{aligned}$$

Besides, by (2.2), we have

$$\begin{aligned}
|\overline{z}_j-\underline{z}_j| &= \left|\sum_{k=1}^{n} \max\{H_{jk}\underline{x}_k, H_{jk}\overline{x}_k\} - \sum_{k=1}^{n} \min\{H_{jk}\underline{x}_k, H_{jk}\overline{x}_k\}\right| \\
&\le \sum_{k=1}^{n} \big|\max\{H_{jk}\underline{x}_k, H_{jk}\overline{x}_k\} - \min\{H_{jk}\underline{x}_k, H_{jk}\overline{x}_k\}\big| \\
&= \sum_{k=1}^{n} \Big|\max\left\{H_{jk}\underline{x}_k - \min\{H_{jk}\underline{x}_k, H_{jk}\overline{x}_k\}, H_{jk}\overline{x}_k - \min\{H_{jk}\underline{x}_k, H_{jk}\overline{x}_k\}\right\}\Big| \\
&= \sum_{k=1}^{n} \Big|\max\big\{\max\{H_{jk}\underline{x}_k - H_{jk}\underline{x}_k, H_{jk}\underline{x}_k - H_{jk}\overline{x}_k\}, \\
&\qquad\qquad \max\{H_{jk}\overline{x}_k - H_{jk}\underline{x}_k, H_{jk}\overline{x}_k - H_{jk}\overline{x}_k\}\big\}\Big| \\
&= \sum_{k=1}^{n} \Big|\max\big\{\max\{0, H_{jk}\underline{x}_k - H_{jk}\overline{x}_k\}, \max\{H_{jk}\overline{x}_k - H_{jk}\underline{x}_k, 0\}\big\}\Big| \\
&= \sum_{k=1}^{n} \big|\max\{H_{jk}(\underline{x}_k-\overline{x}_k), H_{jk}(\overline{x}_k-\underline{x}_k), 0\}\big| \\
&\le \sum_{k=1}^{n} |H_{jk}||\overline{x}_k-\underline{x}_k|. \qquad (2.21)
\end{aligned}$$

Upon inequalities (2.20) and (2.21), it easily follows that

$$\big|g(x)-\overline{g}(x)\big| \le \sum_{j=1}^{n} |\overline{x}_j-\underline{x}_j||\overline{z}_j-\underline{z}_j| \le \sum_{j=1}^{n} |\overline{x}_j-\underline{x}_j| \sum_{k=1}^{n} |H_{jk}||\overline{x}_k-\underline{x}_k|. \qquad (2.22)$$

Thus, $|g(x)-\overline{g}(x)| \to 0$ as $\|D\| = \|\overline{x}-\underline{x}\| \to 0$.

(ii) For any $x \in X \cap D$, for each $i = 0, 1, 2, \ldots, N$, it follows from the formula (2.17) that

$$\begin{aligned}
\left|f_i(x) - \underline{f}_i(x)\right| &= \left| \sum_{j=1}^{n} x_j Q_{ij} x - \max\left\{(\beta_1^i)^T x + \varrho_1^i, (\beta_2^i)^T x + \varrho_2^i\right\}\right| \\
&= \left| \min\left\{ \sum_{j=1}^{n} x_j Q_{ij} x - \left[(\beta_1^i)^T x + \varrho_1^i\right], \sum_{j=1}^{n} x_j Q_{ij} x - \left[(\beta_2^i)^T x + \varrho_2^i\right]\right\}\right| \\
&= \left| \min\left\{ \sum_{j=1}^{n} x_j y_{ij} - \left[(\beta_1^i)^T x + \varrho_1^i\right], \sum_{j=1}^{n} x_j y_{ij} - \left[(\beta_2^i)^T x + \varrho_2^i\right]\right\}\right| \\
&= \left| \min\left\{ \sum_{j=1}^{n} (x_j - \underline{x}_j)(y_{ij} - \underline{y}_{ij}), \sum_{j=1}^{n} (\overline{x}_j - x_j)(\overline{y}_{ij} - y_{ij})\right\}\right| \\
&\leq \sum_{j=1}^{n} |\overline{x}_j - \underline{x}_j||\overline{y}_{ij} - \underline{y}_{ij}|. \qquad (2.23)
\end{aligned}$$

Further, we can also obtain

$$\begin{aligned}
|\overline{y}_{ij} - \underline{y}_{ij}| &= \left| \sum_{k=1}^{n} \max\left\{Q_{ijk}\underline{x}_k, Q_{ijk}\overline{x}_k\right\} - \sum_{k=1}^{n} \min\left\{Q_{ijk}\underline{x}_k, Q_{ijk}\overline{x}_k\right\}\right| \\
&\leq \sum_{k=1}^{n} |Q_{ijk}||\overline{x}_k - \underline{x}_k| \qquad (2.24)
\end{aligned}$$

by a derivation similar to formula (2.21). In view of (2.23)-(2.24), it follows that

$$\left|f_i(x) - \underline{f}_i(x)\right| \leq \sum_{j=1}^{n} |\overline{x}_j - \underline{x}_j||\overline{y}_{ij} - \underline{y}_{ij}| \leq \sum_{j=1}^{n} |\overline{x}_j - \underline{x}_j| \sum_{k=1}^{n} |Q_{ijk}||\overline{x}_k - \underline{x}_k|. \qquad (2.25)$$

Hence, when $\|D\| \to 0$, $|f_i(x) - \underline{f}_i(x)| \to 0$.

(iii) First of all, according to the hypothetical condition $g(x) > 0$ for any $x \in X$, and then

$$0 < g(x) \leq \overline{g}(x), \quad \forall x \in X \cap D.$$

From the formula (2.19), we have

$$f(x) \geq \underline{f}(x), \quad \forall x \in X \cap D.$$

Next, let

$$\triangle_1 = \left| \frac{f_0(x)}{g(x)} - \frac{\underline{f}^l(x)}{g(x)} \right|, \quad \triangle_2 = \left| \frac{\underline{f}^l(x)}{g(x)} - \frac{\underline{f}^l(x)}{\overline{g}(x)} \right|.$$

Then

$$\begin{aligned}
\left|f(x) - \underline{f}(x)\right| &= \left| \frac{f_0(x)}{g(x)} - \frac{\underline{f}^l(x)}{\overline{g}(x)} \right| \\
&\leq \left| \frac{f_0(x)}{g(x)} - \frac{\underline{f}^l(x)}{g(x)} \right| + \left| \frac{\underline{f}^l(x)}{g(x)} - \frac{\underline{f}^l(x)}{\overline{g}(x)} \right| \\
&= \triangle_1 + \triangle_2. \qquad (2.26)
\end{aligned}$$

On the basis of the continuity of functions $g(x)$, $\overline{g}(x)$ and $\underline{f}_0(x)$ over a bounded closed set X, we can see that the function $|1/g(x)|$ is bounded, so there must be positive real numbers L_1 and L_2 such that

$$\left|\frac{1}{g(x)}\right| \le L_1, \quad \left|\frac{\underline{f}_0(x)}{g(x)\overline{g}(x)}\right| \le L_2$$

for any $x \in X$, then

$$\begin{aligned}\triangle_1 &= \left|\frac{f_0(x)}{g(x)} - \frac{\underline{f}^l(x)}{\overline{g}(x)}\right| = \left|\frac{1}{g(x)}\right| \left|f_0(x) - \underline{f}^l(x)\right| \\ &\le L_1 \left|f_0(x) - \max\left\{\underline{f}_0(x), 0\right\}\right| = L_1 \left|\min\left\{f_0(x) - \underline{f}_0(x), f_0(x)\right\}\right| \\ &= L_1 \min\left\{\left|f_0(x) - \underline{f}_0(x)\right|, \left|f_0(x)\right|\right\}. \end{aligned} \tag{2.27}$$

The last equation of formula (2.27) is established mainly because $f_0(x) \ge \underline{f}_0(x)$, $f_0(x) \ge 0$ such that

$$f_0(x) - \underline{f}_0(x) = \left|f_0(x) - \underline{f}_0(x)\right| \ge 0, \quad f_0(x) = \left|f_0(x)\right| \ge 0.$$

Besides, we also have

$$\begin{aligned}\triangle_2 &= \left|\frac{\underline{f}^l(x)}{g(x)} - \frac{\underline{f}^l(x)}{\overline{g}(x)}\right| \le \left|\frac{\max\{\underline{f}_0(x), 0\}}{g(x)\overline{g}(x)}\right| \left|\overline{g}(x) - g(x)\right| \\ &\le \left|\max\left\{\frac{\underline{f}_0(x)}{g(x)\overline{g}(x)}, 0\right\}\right| \left|\overline{g}(x) - g(x)\right| \le L_2 \left|\overline{g}(x) - g(x)\right|. \end{aligned} \tag{2.28}$$

Then, it follows from formulas (2.22) and (2.25) that $\triangle_1 \to 0$ and $\triangle_2 \to 0$ as $\|D\| \to 0$. Therefore, by means of formulas (2.26)-(2.28), it can be concluded that $|f(x) - \underline{f}(x)| \to 0$ as $\|D\| \to 0$. The proof is complete. □

From Theorem 2.1, we know that the optimal value of LFRP_D is the lower bound of the global optimal value of EIGBFP over D.

The problem LFRP_D is still a nonlinear programming problem, but it is a quasi-convex problem, so any KKT point is its global optimal solution. Moreover, the problem LFRP_D is also a LFP problem, which can be solved by the methods in [6,27,28,39]. However, we present a method similar to that in [35] for solving problem LFRP_D. Now, we let $t = 1/v > 0$, $w = tx$, $\mu = ts$, then the problem LFRP_D can be reformulated into a linear program

$$(\mathrm{LP}_D) \quad \begin{cases} \min\ \mu, \\ \text{s.t. } (\beta_1^0 + c_0)^T w + (\varrho_1^0 + d_0)t \le \mu, \quad (\beta_2^0 + c_0)^T w + (\varrho_2^0 + d_0)t \le \mu, \\ \qquad (\alpha_1 + h)^T w + (\delta_1 + q)t \ge 1, \qquad (\alpha_2 + h)^T w + (\delta_2 + q)t \ge 1, \\ \qquad (\beta_1^i + c_i)^T w + (\varrho_1^i + d_i)t \le 0, \qquad i = 1, 2, \ldots, N, \\ \qquad (\beta_2^i + c_i)^T w + (\varrho_2^i + d_i)t \le 0, \qquad i = 1, 2, \ldots, N, \\ \qquad t > 0, \quad \mu \ge 0, \quad Aw - bt \le 0, \quad \underline{x}t \le w \le \overline{x}t. \end{cases}$$

The equivalence between problems LFRP_D and LP_D is explained by Theorem 2.2.

Theorem 2.2. *If (x^*, s^*, v^*) is the global optimal solution of LFRP_D if and only if (w^*, μ^*, t^*) is the global optimal solution of LP_D and*

$$x^* = \frac{w^*}{t^*}, \quad v^* = \frac{1}{t^*}, \quad s^* = \frac{\mu^*}{t^*}.$$

Proof. The argumentation method is similar to the [35, Theorem 1], which is not repeated here. □

Based on Theorem 2.2, if the optimal solution (w^*, μ^*, t^*) of LP_D is obtained, the optimal solution

$$(x^*, s^*, v^*) = \left(\frac{w^*}{t^*}, \frac{\mu^*}{t^*}, \frac{1}{t^*}\right)$$

of LFRP_D can also be obtained indirectly. Therefore, Theorem 2.2 indicates that we can address the linear fractional program LFRP_D by solving a linear program LP_D instead.

Remark 2.1. *Note that any convex combination of functions $\bar{\vartheta}_j^1(w_j, v_j)$ and $\bar{\vartheta}_j^2(w_j, v_j)$ on the right of inequalities* (2.3), (2.4) *can still be used as the upper bound of $w_j v_j$, that is*

$$w_j v_j \le \lambda_j \bar{\vartheta}_j^1(w_j, v_j) + (1-\lambda_j)\bar{\vartheta}_j^2(w_j, v_j), \quad \forall\, \lambda_j \in [0,1],$$
$$w_j v_j \le \lambda_j \bar{\vartheta}_j^2(w_j, v_j) + (1-\lambda_j)\bar{\vartheta}_j^1(w_j, v_j), \quad \forall\, \lambda_j \in [0,1].$$

Of course, any convex combination of functions $\bar{\theta}_{ij}^1(s_j, t_{ij})$ and $\bar{\theta}_{ij}^2(s_j, t_{ij})$ on the right of inequalities (2.11), (2.12) *can still be utilized as the lower bound of $s_j t_{ij}$, that is*

$$s_j t_{ij} \ge \eta_{ij} \bar{\theta}_{ij}^1(s_j, t_{ij}) + (1-\eta_{ij})\bar{\theta}_{ij}^2(s_j, t_{ij}), \quad \forall\, \eta_{ij} \in [0,1],$$
$$s_j t_{ij} \ge \eta_{ij} \bar{\theta}_{ij}^2(s_j, t_{ij}) + (1-\eta_{ij})\bar{\theta}_{ij}^1(s_j, t_{ij}), \quad \forall\, \eta_{ij} \in [0,1].$$

As a result, it may be necessary to find a better convex combination to approximate $w_j v_j$ and $s_j t_{ij}$, which can produce a fairly good approximation of the original problem to determine the lower bound.

3. Branch and Bound Algorithm and Its Convergence

In this section, to better construct an algorithm to effectively solve the MIQCQFP problem, it is necessary to give some basic operations that can be incorporated into the branch-and-bound framework in advance, i.e., the boundary adjustment-segmentation rule of rectangles, feasibility-based detection method of solutions and the bounding operation. Next, we will describe in the specific details of these operations.

3.1. Branching operation and boundary adjustment

Selecting an effective branching strategy will bring great convenience to solve the problem EIGBFP (MIQCQFP) in a large way. In this paper, we will combine the standard bisection rule with an integer interval adjusting technique to divide each region of interest. The main reason that this method can guarantee the global convergence of the algorithm is that it only removes the part without the optimal solution and improves the efficiency of the algorithm to the maximum extent.

Taking into account any subproblems defined on the rectangle

$$D = \prod_{j=1}^{n} [\underline{x}_j, \overline{x}_j] = [\underline{x}, \overline{x}] \subseteq D^0,$$

the detailed process of dividing D into two sub-rectangles by the adjustment-segmentation method of rectangles is as follows:

(i) For each $j = 1, 2, \ldots, n$, set

$$\underline{x}_j = \begin{cases} \lceil \underline{x}_j \rceil, & j \in N_I, \\ \underline{x}_j, & j \notin N_I, \end{cases} \quad \overline{x}_j = \begin{cases} \lfloor \overline{x}_j \rfloor, & j \in N_I, \\ \overline{x}_j, & j \notin N_I. \end{cases}$$

Then, let

$$l = \arg\max \big\{ \overline{x}_j - \underline{x}_j \mid j = 1, 2, \ldots, n \big\}, \quad \tilde{x}_l = \frac{\overline{x}_l + \underline{x}_l}{2}.$$

(ii) If $l \in N_I$, set

$$D' = \prod_{j=1}^{l-1} \big[\underline{x}_j, \overline{x}_j\big] \times \big[\underline{x}_l, \lfloor \tilde{x}_l \rfloor\big] \times \prod_{j=l+1}^{n} \big[\underline{x}_j, \overline{x}_j\big],$$

$$D'' = \prod_{j=1}^{l-1} \big[\underline{x}_j, \overline{x}_j\big] \times \big[\lceil \tilde{x}_l \rceil, \overline{x}_l\big] \times \prod_{j=l+1}^{n} \big[\underline{x}_j, \overline{x}_j\big].$$

Otherwise, set

$$D' = \prod_{j=1}^{l-1} \big[\underline{x}_j, \overline{x}_j\big] \times \big[\underline{x}_l, \tilde{x}_l\big] \times \prod_{j=l+1}^{n} \big[\underline{x}_j, \overline{x}_j\big],$$

$$D'' = \prod_{j=1}^{l-1} \big[\underline{x}_j, \overline{x}_j\big] \times \big[\tilde{x}_l, \overline{x}_l\big] \times \prod_{j=l+1}^{n} \big[\underline{x}_j, \overline{x}_j\big].$$

Thus, by taking advantage of the adjustment-segmentation rule of rectangles described above, each rectangle D can be divided into two sub-rectangles D' and D'' respectively.

3.2. Feasibility detecting and bounding operation

For each sub-rectangle D generated after the above branching operation, the bounding operation is mainly focused on the lower bound $LB(D)$ and upper bound $UB(D)$ of the optimal value of EIGBFP problem. Assuming that the UB is the best upper bound known to date, the bounding process can be run on the following ways. By solving the linear fractional relaxation programming subproblem $\mathrm{LFRP}_{D^{ki}}$, $i \in \{1, 2\}$, over the region of all the sub-rectangles we are still interested in the k-th iteration, then the minimum of these optimal values will provide the lower bound for the optimal value of EIGBFP.

As for the upper bound determination method, a so-called feasibility-based detection method is used here. Specifically, if the integer index set $N_I = \emptyset$, then the optimal solution of the linear fractional relaxation programming subproblem $\mathrm{LFRP}_{D^{ki}}$ over any sub-rectangle-region is x_{ki}, $i \in \{1, 2\}$; if there is an $i \in \{1, 2\}$ such that x_{ki} is feasible for problem EIGBFP, then the minimum of all the objective function values corresponding to all such feasible solutions x_{ki}, $i \in \{1, 2\}$, for the problem EIGBFP can be denoted as UB_k, then $\min\{UB, UB_k\}$ will provide a new upper bound for EIGBFP. Otherwise, if $N_I \neq \emptyset$, for the optimal solution x^{ki}, $i \in \{1, 2\}$, over each sub-rectangle that we are still interested in, set $x_j^{ki} = [x_j^{ki}]$, $j \in N_I$, $i \in \{1, 2\}$. Suppose there is an $i \in \{1, 2\}$ such that x_{ki} a feasible solution to the problem EIGBFP, then set

$$UB_k = \min \big\{ f(x_{ki}) \mid i \in \{1, 2\} \big\}.$$

Moreover, for each $i \in \{1, 2\}$, set

$$x_{mid}^{ki} = \left[\frac{\overline{x}^{ki} + \underline{x}^{ki}}{2}\right],$$

if the existence of $i \in \{1, 2\}$ makes x_{mid}^{ki} a feasible solution to EIGBFP, let

$$\overline{UB}_k = \min\left\{f(x_{mid}^{ki}) \middle| \ i \in \{1, 2\}\right\},$$

otherwise set $\overline{UB}_k = +\infty$. Then, $\min\{UB, UB_k, \overline{UB}_k\}$ will provide a new upper bound for EIGBFP.

3.3. Algorithm narrative

Based on the previous discussion, we summarize the steps of the branch-and-bound algorithm as follows.

Algorithm 3.3: Branch-and-bound algorithm.

Step 1. Initialization.

Step 1.1. Select a convergence tolerance $\epsilon > 0$. Set iteration counter $k := 0$, initial upper bound $UB = +\infty$. The set of initial rectangle is set to $\Omega_0 = \{D^0\}$. Transform the problem MIQCQFP into an equivalent form defined by EIGBFP.

Step 1.2. If the initial linear fractional relaxation problem LFRP_{D^0} is infeasible, then the original problem is unsolvable; otherwise, the optimal solution and value of problem LFRP_{D^0} are denoted as x^0 and $LB(D^0)$, respectively. Set the initial lower bound to $LB = LB(D^0)$ and explore the upper bound UB according to the feasibility-based detection rules outlined in Section 3.2. If $UB - LB \leq \epsilon$, terminate the algorithm, UB is the optimal value of EIGBFP, otherwise continue with Step 2.

Step 2. Rectangular adjustment-segmentation rule.

For each sub-rectangle $D \subseteq \Omega_k$, the rectangle D is divided into two new sub-rectangles D' and D'' according to the adjustment-segmentation rules described in Section 3.1. Remove the D from Ω_k and put the two new sub-rectangles into the remaining rectangular set, continuing to denote the set of new sub-rectangles as Ω_k.

Step 3. Bounding and pruning.

For each sub-rectangle $D \subseteq \Omega_k$, solving the linear fractional relaxation programming problem LFRP_D; If $LB(D) > UB - \epsilon$, remove the D from the Ω_k; Otherwise, if possible, we can update the lower and upper bounds according to the feasibility-based detection method and bounding operation mentioned in Sections 3.1-3.2.

Step 4. Termination.

If $UB - LB \leq \epsilon$, the algorithm can be stopped, and UB is the global optimal value of the problem EIGBFP(MIQCQFP). Otherwise, set the LB and UB to the latest lower and upper bounds of the global optimal value of EIGBFP. Then set $k := k + 1$. Choose a rectangle corresponding to the minimum optimal value of the relaxation programming subproblem for the next partition object, and return to Step 2.

3.4. Convergence of the algorithm

Theorem 3.1. *If problem* MIQCQFP *is feasible, the algorithm terminates after a finite number of iterations and returns a global optimal solution for* MIQCQFP.

Proof. Suppose F denotes the feasible region of problem MIQCQFP, and if $|N_I| = n$($|N_I|$ denotes the number of elements in N_I), then the feasible region F is obviously a finite set. Through the rectangular adjustment-segmentation rule, each adjusted sub-rectangle $D = [\underline{x}, \overline{x}] \subseteq D^0$ will satisfy $\underline{x} = [\underline{x}]$, $\overline{x} = [\overline{x}]$. Therefore, there will be a finite number of child nodes in the branch-and-bound tree, that is, the algorithm has a finite number of iterations.

Otherwise, if $|N_I| \neq n$, we assume that the algorithm has an infinite number of iterations. Then, an infinite rectangle-sequence $\{D^k\}$ is generated in the light of the rectangular adjustment-segmentation rule. Since the bisection method is exhausted and the selected rectangles to be partitioned is improved, we will know that

$$\lim_{k\to\infty} \underline{x}^k = \lim_{k\to\infty} \overline{x}^k = x^* \in F, \quad \lim_{k\to\infty} UB^k \;\; \to \;\; f(x^*). \tag{3.1}$$

Moreover, by the pruning and termination rules in Steps 3-4, we have

$$LB(D^k) \leq UB^k - \epsilon. \tag{3.2}$$

Nevertheless, from the method of determining the lower bound in Step 3 and the conclusion (iii) of Theorem 2.1, we know that

$$LB(D^k) \;\; \to \;\; f(x^*). \tag{3.3}$$

Therefore, taking the limit on both sides of inequality (3.2), and using (3.1) and (3.3) at the same time, we have

$$f(x^*) \leq f(x^*) - \epsilon. \tag{3.4}$$

It can be observed that the establishment of inequality (3.4) is contradictory, so the algorithm terminates after a finite number of iterations.

If the algorithm terminates at the k-th iteration, then there is a solution $x^k \in F$ such that

$$UB^k = f(x^k). \tag{3.5}$$

According to the termination rule, we have

$$UB^k - LB^k \leq \epsilon. \tag{3.6}$$

Moreover, from the feasibility-based detection strategy of solutions and the bounding operation described in Section 3.2, we know that

$$LB^k \leq f(x^*). \tag{3.7}$$

Suppose x^* is a global optimal solution, then according to formulas (3.5)-(3.7), we can obtain

$$f(x^*) - \epsilon \leq f(x^k) - \epsilon \leq LB^k \leq f(x^*),$$

that is, x^k becomes a global optimal solution of problem MIQCQFP. The proof is complete. □

4. Numerical Experiments

To verify the feasibility and effectiveness of the proposed algorithm, we will utilise this algorithm to perform some numerical test experiments. In view of this, we have compiled and executed the code for the algorithms in this paper and some other literature at Matlab (2017a). In these algorithms, the linear programming subproblems required to be solved are carried out by using the linprog solver of Matlab. Besides, all the QCQP subproblems needed to be solved in Dinkelbach's algorithm [8] are directly carried out by utilizing academic software package SCIP. All experimental procedures were performed on a desktop computer with Intel(R)Core(TM)i5−2320 3.00 GHz power processor, 4.00 GB memory, and Microsoft Win7 operating system. In addition, the convergence accuracy adopted by all numerical experiments is 10^{-7}.

By utilising the proposed algorithm, we first solve 18 existing test problems, which were given in [18,23,34,46]. Among them, Examples 1-5 are SLFP problems, which correspond to Examples 1, 3-5, and 9 in [23], respectively; Examples 6-9 are MLFP problems, which are Examples $1, 2, 4$ and 8 in [18], respectively; Examples 10-13 are LMP problems, which correspond to Examples $1, 2, 6$, and 10 in [34], respectively; Examples 14-18, of course, are MIQCQP problems, which correspond to Examples 5-9 in [46], respectively. The numerical results for these 18 test examples are listed in Tables 4.1-4.2.

Moreover, we have constructed 14 additional problems, namely Problems 1-14. The constraints adopted to construct these 14 problems are shown in Tables 4.3-4.4, and the properties of the numerators and denominators and feasible regions in these 14 problems are also recorded in Table 4.5. These test problems are indicated below, and their numerical results are also recorded in Tables 4.6-4.7.

As in Tables 4.1 and 4.6-4.9, the symbol for the header line means: Solution: the optimal solution; Optimum: the optimal value; Iter: the number of iterations; Time: the CPU running time in seconds; BBA: the proposed algorithm; OSBBA: the algorithm in [36]; BRA: the algorithm in [11]; Avg.iter: the average number of iterations of the correlation algorithm for solving the 15 test problems; Avg.time: the average CPU running time(s) spent by the relevant algorithm for 15 test problems; Avg.Val: the average optimal value obtained from 15 test problems with relevant algorithm; "–": the algorithm cannot solve the problem in 3600 seconds for all cases.

Problem 1.

$$\begin{cases} \min \dfrac{-x_1^2 - x_2^2 - 3x_3^2 - 4x_4^2 - 2x_5^2 + 8x_1 + 2x_2 + 3x_3 + x_4 + 2x_5 + 60000}{7x_1^2 + 6x_2^2 + 12x_1 + 8x_3^2 + 4x_2x_3 + 6x_4^2 + 2x_3x_4 + 7x_5^2 + 11}, \\ \text{s.t. } x \in X_1. \end{cases}$$

Problem 2.

$$\begin{cases} \min \dfrac{x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2}{-x_1^2 - 2x_2^2 - 8x_3^2 - 4x_2x_3 - 6x_4^2 - 2x_3x_4 - 7x_5^2 - 1}, \\ \text{s.t. } x \in X_1. \end{cases}$$

Problem 3.

$$\begin{cases} \min \dfrac{7x_1^2 + 6x_2^2 + 12x_1 + 8x_3^2 + 4x_2x_3 + 6x_4^2 + 2x_1x_4 + 11}{x_1^2 + x_2^2 + 3x_3^2 + 4x_4^2 - x_1 - x_2 - 3x_3 - 2x_4}, \\ \text{s.t. } x \in X_3. \end{cases}$$

Problem 4.

$$\begin{cases} \min \dfrac{7x_1^2+6x_2^2+12x_1+8x_3^2+4x_2x_3+6x_4^2+2x_1x_4+11}{x_1^2+x_2^2+3x_3^2+4x_4^2-x_1-x_2-3x_3-2x_4}, \\ \text{s.t. } x \in X_3. \end{cases}$$

Problem 5.

$$\begin{cases} \min \dfrac{-x_1^2-x_2^2-3x_3^2+8x_1+2x_2+3x_3+60000}{7x_1^2+6x_2^2-15.8x_1-93.2x_2+8x_3^2-6x_1x_3+4x_2x_3-63x_3+440}, \\ \text{s.t. } x \in X_2. \end{cases}$$

Problem 6.

$$\begin{cases} \min \dfrac{-5x_1^2-4x_2^2-4x_3^2-6x_4^2-4x_5^2-2x_1x_3+2x_1x_4+6x_3x_4-2x_3x_5-4x_4x_5-11}{-x_1^2-2x_2^2-8x_3^2-4x_2x_3-6x_4^2-2x_3x_4-7x_5^2-1}, \\ \text{s.t. } x \in X_7. \end{cases}$$

Problem 7.

$$\begin{cases} \min \dfrac{-5x_1^2-4x_2^2-4x_3^2-6x_4^2-4x_5^2-2x_1x_3+2x_1x_4+6x_3x_4-2x_3x_5-4x_4x_5-11}{-x_1^2-2x_2^2-8x_3^2-4x_2x_3-6x_4^2-2x_3x_4-7x_5^2-1}, \\ \text{s.t. } x \in X_5. \end{cases}$$

Problem 8.

$$\begin{cases} \min \dfrac{7x_1^2+6x_2^2+8x_3^2+4x_2x_3+6x_4^2+7x_5^2+2x_1x_4+11}{x_1^2+2x_2^2+3x_3^2+4x_4^2+5x_5^2-x_1-x_2-2x_3-3x_4-4x_5}, \\ \text{s.t. } x \in X_5. \end{cases}$$

Problem 9.

$$\begin{cases} \min \dfrac{5x_1^2+x_2^2+2x_3^2+10x_4^2+5x_5^2+2x_1x_3-2x_1x_4-2x_2x_4+2x_3x_5+6x_4x_5}{-5x_1^2-4x_2^2-4x_3^2-6x_4^2-4x_5^2-2x_1x_3+2x_1x_4+6x_3x_4-2x_3x_5-4x_4x_5}, \\ \text{s.t. } x \in X_5. \end{cases}$$

Problem 10.

$$\begin{cases} \min \dfrac{-x_1^2-x_2^2-3x_3^2-4x_4^2-2x_5^2+8x_1+2x_2+3x_3+x_4+2x_5}{7x_1^2+6x_2^2+8x_3^2+4x_2x_3+6x_4^2+2x_3x_4+7x_5^2+11}, \\ \text{s.t. } x \in X_5. \end{cases}$$

Problem 11.

$$\begin{cases} \min \dfrac{7x_1^2+6x_2^2+12x_1+8x_3^2+4x_2x_3+6x_4^2+7x_5^2+2x_1x_4+11}{x_1^2+2x_2^2+3x_3^2+4x_4^2+5x_5^2-x_1-x_2-2x_3-3x_4-4x_5}, \\ \text{s.t. } x \in X_4. \end{cases}$$

Problem 12.

$$\begin{cases} \min \dfrac{5x_1^2+x_2^2+7x_3^2+5x_4^2+7x_5^2+2x_1x_2+4x_1x_5+2x_2x_5-4x_3x_4+15}{-3x_1^2-5x_2^2-6x_3^2-6x_4^2-4x_5^2-4x_1x_4+6x_2x_3+2x_2x_4+2x_3x_5+2x_4x_5+2552}, \\ \text{s.t. } x \in X_6. \end{cases}$$

Problem 13.

$$\begin{cases} \min \dfrac{-x_1^2 - x_2^2 - 3x_3^2 - 4x_4^2 - 2x_5^2 + 8x_1 + 2x_2 + 3x_3 + x_4 + 2x_5}{7x_1^2 + 6x_2^2 + 8x_3^2 + 4x_2x_3 + 6x_4^2 + 2x_3x_4 + 7x_5^2 + 11}, \\ \text{s.t. } x \in X_4. \end{cases}$$

Problem 14.

$$\begin{cases} \min \dfrac{-x_1^2 - x_2^2 - 3x_3^2 - 4x_4^2 - 2x_5^2 + 8x_1 + 2x_2 + 3x_3 + x_4 + 2x_5}{-x_1^2 - 2x_2^2 - 8x_3^2 - 4x_2x_3 - 6x_4^2 - 2x_3x_4 - 7x_5^2 - 11}, \\ \text{s.t. } x \in X_6. \end{cases}$$

Firstly, it must be pointed out that the algorithms in [11, 18, 19, 22, 23, 34, 36, 46, 47] are all branch-and-bound algorithms, and the branching operations of the algorithms in [11, 18, 34, 46] occur in the n-dimensional space where the decision variables are located. Moreover, two well-known global optimization solvers, SCIP and BARON, which are employed to solve mixed integer (linear or nonlinear) programming problems, are also mainly integrated on the branch-and-bound algorithm framework that performs branching operations in n-dimensional space. Branching operations of the algorithms in [19, 22, 23, 36] take place in the p-dimensional space where auxiliary variables exist. In contrast, the branching operation of the algorithm in [47] only takes place in the 1-dimensional space where an auxiliary variable is located. Generally speaking, $n > p > 1$, and it can be seen from Table 4.1 that when solving MLFP problems, the algorithm in [47] performs much better than the algorithm in [18] and our algorithm, which due to the implicit enumeration characteristics of the branch-and-bound algorithm. Secondly, as is given in Tables 4.1-4.2, the numerical results show that our algorithm can find the optimal solution of these 18 test examples within a finite iteration, and the optimal value is at least as good as that in other literature. Thirdly, although the performance of our algorithm is not the best, the computing time is not more than three seconds. This demonstrates that our algorithm is robust and efficient. Fourthly, our algorithm can solve at least four other types of problems, namely SLFP, MLFP, LMP and MIQCQP, which indicate that our algorithm has a wide range of application.

For Problems 1-14, we solved them with our algorithm, Dinkelbach's algorithm, SCIP and BARON respectively. Numerical results in Tables 4.6-4.7 show that all optimal solutions and values obtained by our algorithm are basically the same as those obtained by the other three methods. In addition, Dinkelbach's algorithm based on SCIP generally uses few iterations and spends the most time in each iteration, but it takes the least time in total, except for Problems 3-5. When $N_I = \emptyset$, SCIP could not solve Problem 12 in 3600 seconds (in fact, it could not solve the problem in 14 hours), and of course, when $N_I = \{1, 2, 3, 4, 5\}$, SCIP was the least effective. Except for Problems 2 and 12, SCIP takes less time to address the other 12 problems than our algorithm. However, our algorithm solves Problems 2, 5, 8-12 with fewer iterations than SCIP. For Problems 1-5 and 12, our algorithm takes less time than BARON. Also, our algorithm takes less time than BARON to solve Problems 9, 10, 13 and 14 with $N_I = \{1, 2, 3, 4, 5\}$. Nevertheless, our algorithm has fewer iterations than BARON only when solving Problems 9, 12 and 14 with $N_I = \{1, 2, 3, 4, 5\}$ and Problem 2 with $N_I = \{1, 3, 5\}$. Although the computational results of our algorithm are not the best, it is sufficient to demonstrate that the algorithm can effectively solve these 14 problems with different attributes.

In any case, the numerical results of the above 32 test examples or problems are sufficient to illustrate that our algorithm is feasible. To further test the proposed algorithm, we also employ Problems 15 and 16 below for generating a series of random instances.

Problem 15.

$$\begin{cases} \max \sum\limits_{i=1}^{p} \left(\mathbf{c}_i^T \mathbf{x} + c_{0i}\right)\left(\mathbf{d}_i^T \mathbf{x} + d_{0i}\right), \\ \text{s.t. } \mathbf{A}\mathbf{x} \leq \mathbf{b}, \quad 0 \leq x_j \leq 1, \quad j = 1, 2, \ldots, n, \end{cases}$$

where the elements of $\mathbf{c}_i, d_i, \mathbf{e}_i, f_i$ are pseudo-random numbers in $[-1, 1]$, the real elements of $\mathbf{A}, \mathbf{b}$ are pseudo-randomly generated in the range $[0, 1]$. This problem is a kind of LMP problems, which has been to run 15 times by five algorithms respectively, and the numerical results are listed in Table 4.8.

Table 4.1: Comparison of results in Examples 1-9.

Type	Example	Ref.	Solution	Optimum	Iter	Time
SLFP	1	[23]	(0.0000,0.2816)	1.6232	25498	257.7659
		[22]	(0.0000,0.2839)	1.6232	14126	140.6823
		BARON	(0.0000,0.2839)	1.6232	3	0.0637
		SCIP	(0.0000,0.2838)	1.6232	39	0.2208
		ours	(0.0000,0.2813)	1.6232	16	0.1673
	2	[23]	(5.0000,0.0000,0.0000)	2.8619	18239	192.2375
		[22]	(5.0000,0.0000,0.0000)	2.8619	132	1.7129
		[19]	(5.0000,0.0000,0.0000)	2.8619	24509	789.8082
		BARON	(5.0000,0.0000,0.0000)	2.8619	5	0.0646
		SCIP	(5.0000,0.0000,0.0000)	2.8619	25	0.1138
		ours	(5.0000,0.0000,0.0000)	2.8619	128	1.5806
	3	[23]	(1.1111,0.0000,0.0000)	-4.0907	131	1.2846
		[22]	(1.1111,0.0000,0.0000)	-4.0907	94	1.1804
		[19]	(1.1111,0.0000,0.0000)	-4.0907	2992	91.2958
		BARON	(1.1111,0.0000,0.0000)	-4.0907	1	0.0482
		SCIP	(1.1111,0.0000,0.0000)	-4.0907	7	0.0347
		ours	(1.1111,0.0000,0.0000)	-4.0907	18	0.2169
	4	[23]	(0.0000,1.6667,0.0000)	3.7109	138	1.2973
		[22]	(0.0000,1.6667,0.0000)	3.7109	63	0.7951
		[19]	(0.0000,1.6667,0.0000)	3.7109	1708	51.8738
		BARON	(0.0000,1.6667,0.0000)	3.7109	1	0.0461
		SCIP	(0.0000,1.6667,0.0000)	3.7109	7	0.0325
		ours	(0.0000,1.6667,0.0000)	3.7109	49	0.5869
	5	[23]	(3.0000,4.0000)	3.2917	1251	11.5671
		[22]	(3.0000,4.0000)	3.2917	8	0.2242
		[19]	(3.0000,4.0000)	3.2917	660	19.9942
		BARON	(3.0000,4.0000)	3.2917	1	0.0426
		SCIP	(3.0000,4.0000)	3.2917	1	0.0124
		ours	(3.0000,4.0000)	3.2917	1	0.0041
MLFP	6	[18]	(1.5,1.5)	1.4895	50	0.5184
		[47]	(1.5,1.5)	1.4895	1	0.0053
		BARON	(1.5,1.5)	1.4895	1	0.0322
		SCIP	(1.5,1.5)	1.4895	1	0.0156
		ours	(1.5,1.5)	1.4895	6	0.0672
	7	[18]	(1.0157,0.5905,1.4037)	0.5731	30	0.3235
		[47]	(1.0157,0.5905,1.4037)	0.5731	4	0.0388
		BARON	(1.0157,0.5905,1.4037)	0.5731	1	0.0307
		SCIP	(1.0157,0.5905,1.4037)	0.5731	1	0.0285
		ours	(1.0157,0.5905,1.4037)	0.5731	27	0.3486
	8	[18]	(1.0083,0.50,1.45)	2.2851	29	0.3792
		[47]	(1.0083,0.50,1.45)	2.2851	2	0.0143
		BARON	(1.0083,0.50,1.45)	2.2851	1	0.0416
		SCIP	(1.0083,0.50,1.45)	2.2851	1	0.0386
		ours	(1.0083,0.50,1.45)	2.2851	33	0.3457
	9	[18]	(1.5054,0.35,1.55)	1.1179	39	0.4721
		[47]	(1.5054,0.35,1.55)	1.1179	7	0.0558
		BARON	(1.5054,0.35,1.55)	1.1179	1	0.0535
		SCIP	(1.5054,0.35,1.55)	1.1179	1	0.0353
		ours	(1.5054,0.35,1.55)	1.1179	46	0.5127

Table 4.2: Comparison of results in Examples 10-18.

Type	Example	Ref.	N_I	Solution	Optimum	Iter	Time
LMP	10	[34]	$\emptyset$	(0.0000,3.0000)	-2.5000	813	28.3545
		[36]	$\emptyset$	(0.0000,3.0000)	-2.5000	351	3.5209
		[11]	$\emptyset$	(0.0000,3.0000)	-2.5000	32	0.4012
		BARON	$\emptyset$	(0.0000,3.0000)	-2.5000	1	0.0371
		SCIP	$\emptyset$	(0.0000,3.0000)	-2.5000	1	0.0209
		ours	$\emptyset$	(0.0000,3.0000)	-2.5000	11	0.1076
	11	[34]	$\emptyset$	(1.3148,0.1396,0.0000,0.4233)	0.8902	697	7.8712
		[36]	$\emptyset$	(1.3148,0.1396,0.0000,0.4233)	0.8902	20	0.2249
		[11]	$\emptyset$	(1.3148,0.1396,0.0000,0.4233)	0.8902	16	0.1372
		BARON	$\emptyset$	(1.3148,0.1396,0.0000,0.4233)	0.8902	9	0.0844
		SCIP	$\emptyset$	(1.3148,0.1396,0.0000,0.4233)	0.8902	11	0.1364
		ours	$\emptyset$	(1.3148,0.1396,0.0000,0.4233)	0.8902	40	0.4036
	12	[34]	$\emptyset$	(0.0000,0.0000)	4	4955	245.6885
		[36]	$\emptyset$	(0.0000,0.0000)	4	26	0.2295
		[11]	$\emptyset$	(0.0000,0.0000)	4	1	0.0051
		BARON	$\emptyset$	(0.0000,0.0000)	4	1	0.0126
		SCIP	$\emptyset$	(0.0000,0.0000)	4	1	0.0104
		ours	$\emptyset$	(0.0000,0.0000)	4	1	0.0036
	13	[34]	$\emptyset$	(5.5556,1.7778,2.6667)	-112.7531	3276	112.3467
		[36]	$\emptyset$	(5.5556,1.7778,2.6667)	-112.7531	619	6.4089
		[11]	$\emptyset$	(5.5556,1.7778,2.6667)	-112.7531	49	0.5459
		BARON	$\emptyset$	(5.5556,1.7778,2.6667)	-112.7531	5	0.0993
		SCIP	$\emptyset$	(5.5556,1.7778,2.6667)	-112.7531	11	0.0936
		ours	$\emptyset$	(5.5556,1.7778,2.6667)	-112.7531	33	0.2863
MIQCQP	14	[11]	$\emptyset$	(50.5,99,0,99,59.8)	-57886.73	84	1.0743
		[46]	$\emptyset$	(50.5,99,0,99,59.8)	-57886.73	101	1.0695
		BARON	$\emptyset$	(50.5,99,0,99,59.8)	-57886.73	1	0.0301
		SCIP	$\emptyset$	(50.5,99,0,99,59.8)	-57886.73	1	0.0284
		ours	$\emptyset$	(50.5,99.0,0.0,99.0,59.8)	-57886.73	62	0.5437
		[46]	$\{1\}$	(50,99,0,99,59.8)	-57840.48	97	0.9853
		BARON	$\{1\}$	(50,99,0,99,59.8)	-57840.48	3	0.0471
		SCIP	$\{1\}$	(50,99,0,99,59.8)	-57840.48	1	0.0385
		ours	$\{1\}$	(50,99,0,99,59.8)	-57840.48	24	0.2355
		[46]	$\{5\}$	(50.5,99,0,99,59)	-57698.25	167	1.7661
		BARON	$\{5\}$	(50.5,99,0,99,59)	-57698.25	1	0.0312
		SCIP	$\{5\}$	(50.5,99,0,99,59)	-57698.25	1	0.0284
		ours	$\{5\}$	(50.5,99,0,99,59)	-57698.25	38	0.3595
		[46]	$\{1,5\}$	(50,99,0,99,59)	-57652.00	92	0.9455
		BARON	$\{1,5\}$	(50,99,0,99,59)	-57652.00	3	0.0461
		SCIP	$\{1,5\}$	(50,99,0,99,59)	-57652.00	1	0.0442
		ours	$\{1,5\}$	(50,99,0,99,59)	-57652.00	21	0.1858
		[46]	$\{1,2,3,4,5\}$	(50,98,0,99,59)	-57652.00	86	0.9265
		BARON	$\{1,2,3,4,5\}$	(50,98,0,99,59)	-57652.00	3	0.0406
		SCIP	$\{1,2,3,4,5\}$	(50,98,0,99,59)	-57652.00	1	0.0209
		ours	$\{1,2,3,4,5\}$	(50,99,0,99,59)	-57652.00	20	0.1761
	15	[46]	$\{1,2,3\}$	(2,6,3)	-427.8	54	0.5475
		BARON	$\{1,2,3\}$	(2,6,3)	-427.8	3	0.1178
		SCIP	$\{1,2,3\}$	(2,6,3)	-427.8	1	0.0526
		ours	$\{1,2,3\}$	(2,6,3)	-427.8	24	0.2638
	16	[46]	$\{1,2,3,4\}$	(2,7,3,2)	-481.2	138	1.5206
		BARON	$\{1,2,3,4\}$	(2,7,3,2)	-481.2	5	0.3830
		SCIP	$\{1,2,3,4\}$	(2,7,3,2)	-481.2	5	0.0986
		ours	$\{1,2,3,4\}$	(2,7,3,2)	-481.2	66	0.7536
	17	[46]	$\{1,2,3,4,5\}$	(2,6,3,2,8)	-585.2	609	7.0503
		BARON	$\{1,2,3,4,5\}$	(2,6,3,2,8)	-585.2	21	1.9436
		SCIP	$\{1,2,3,4,5\}$	(2,6,3,2,8)	-585.2	9	0.3029
		ours	$\{1,2,3,4,5\}$	(2,6,3,2,8)	-585.2	233	3.0048
	18	[46]	$\{4,5,6,7\}$	(0.2,0.8,1.9079,1,1,0,1)	4.5796	72	0.7666
		BARON	$\{4,5,6,7\}$	(0.2,0.8,1.9079,1,1,0,1)	4.5796	1	0.7841
		SCIP	$\{4,5,6,7\}$	(0.2,0.8,1.9079,1,1,0,1)	4.5796	1	0.0486
		ours	$\{4,5,6,7\}$	(0.2,0.8,1.9079,1,1,0,1)	4.5796	40	0.4639

In Table 4.8, the numerical results reveal that BBA uses more iterations and CPU running time than BRA. In fact, it was revealed in [36] that OSBBA is more suitable for solving some LMP problems with $p \ll n$. In Table 4.8, OSBBA uses less CPU running time and iterations than BBA only when $(p, m, n) = (5, 100, 100), (5, 100, 200)$ and $(10, 100, 200)$, which is precisely because $p \ll n$. Compared to SCIP and BARON, our algorithm BBA has inferior numerical results in the first 18 or 16 groups. However, BBA is insensitive to the size of parameter p, but the time taken by both software packages increases as p increases. It can be observed from the last three and five sets of numerical results, respectively, that SCIP and BARON eventually consume more CPU running time than BBA, especially BARON is quite sensitive to p. Thus, when solving LMP problems, there are the following three conclusions:

i) BBA performs worse than BRA, which is mainly caused by the linear relaxation of BBA not as tight as BRA.

ii) Except for the case of $p \ll n$, BBA has a stronger computing power than OSBBA.

iii) BBA performs better than SCIP and BARON in certain cases such as
$$(p, m, n) = (200, 100, 10), (300, 100, 10), (500, 100, 10).$$

Table 4.3: Constraints on the constructed test problems for X_1-X_4.

Problem	Numerator curvature
X_1	$\begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 \le 400, \\ 2x_1 + x_2 + 6x_3 \le 200, \\ x_1 + 2x_2 + 2x_3 + x_4 + 6x_5 \le 800, \\ x_3 - x_4 + 5x_5 \le 200, \\ 0 \le x_i \le 99, \quad x_i \in \mathbb{Z}, \quad \forall\, i = 1, 2, 3, 4, 5. \end{cases}$
X_2	$\begin{cases} 9x_1^2 + 10x_1x_2 + 8x_2^2 + 5x_3^2 + 6x_1x_3 + 10x_2x_3 \le 1000, \\ 6x_1^2 + 8x_1x_2 + 6x_2^2 + 4x_3^2 + 2x_1x_3 + 2x_2x_3 \le 440, \\ 9x_1^2 + 6x_2^2 + 8x_3^2 - 2x_1x_2 - 2x_2x_3 \le 340, \\ 1 \le x_i \le 200, \quad x_i \in \mathbb{Z}, \quad \forall\, i = 1, 2, 3. \end{cases}$
X_3	$\begin{cases} 9x_1^2 + 10x_1x_2 + 8x_2^2 + 5x_3^2 + 6x_1x_3 + 10x_2x_3 + 7x_4^2 + 10x_1x_4 + 6x_2x_4 + 2x_3x_4 \le 1100, \\ 6x_1^2 + 8x_1x_2 + 6x_2^2 + 4x_3^2 + 2x_1x_3 + 2x_2x_3 + 8x_4^2 - 2x_1x_4 - 10x_2x_4 \le 440, \\ 9x_1^2 + 6x_2^2 + 8x_3^2 - 2x_1x_2 - 2x_2x_3 + 6x_4^2 - 4x_1x_4 - 4x_2x_4 + 2x_3x_4 \le 310, \\ 8x_1^2 + 4x_2^2 + 9x_3^2 + 7x_4^2 + 2x_1x_2 + 2x_1x_3 + 4x_2x_3 - 6x_1x_4 - 2x_2x_4 + 2x_3x_4 \le 460, \\ 1 \le x_i \le 200, \quad x_i \in \mathbb{Z}, \quad \forall\, i = 1, 2, 3, 4. \end{cases}$
X_4	$\begin{cases} 9x_1^2 + 10x_1x_2 + 8x_2^2 + 5x_3^2 + 6x_1x_3 + 10x_2x_3 + 7x_4^2 + 10x_1x_4 \\ \quad + 6x_2x_4 + 2x_3x_4 + 2x_2x_5 + 7x_5^2 \le 1000, \\ 6x_1^2 + 8x_1x_2 + 6x_2^2 + 4x_3^2 + 2x_1x_3 + 2x_2x_3 + 8x_4^2 \\ \quad - 2x_1x_4 - 10x_2x_4 + 2x_1x_5 + 6x_2x_5 - 6x_4x_5 - 7x_5^2 \le 1150, \\ 9x_1^2 + 6x_2^2 + 8x_3^2 - 2x_1x_2 - 2x_2x_3 + 6x_4^2 - 4x_1x_4 \\ \quad - 4x_2x_4 + 2x_3x_4 + 6x_1x_5 + 2x_2x_5 - 4x_4x_5 + 6x_5^2 \le 800, \\ 8x_1^2 + 4x_2^2 + 9x_3^2 + 7x_4^2 + 2x_1x_2 + 2x_1x_3 + 4x_2x_3 \\ \quad - 6x_1x_4 - 2x_2x_4 + 2x_3x_4 + 6x_1x_5 + 4x_2x_5 + 2x_3x_5 - 6x_5^2 \le 1100, \\ 4x_1^2 + 5x_2^2 + 8x_3^2 + 6x_4^2 - 2x_1x_2 + 6x_1x_3 + 2x_1x_4 \\ \quad - 6x_2x_4 + 2x_3x_4 + 4x_1x_5 - 2x_2x_5 + 6x_3x_5 + 7x_5^2 + 8x_4x_5 \le 975, \\ 1 \le x_i \le 200, \quad x_i \in \mathbb{Z}, \quad \forall\, i = 1, 2, 3, 4, 5. \end{cases}$

Table 4.4: Constraints on the constructed test problems for X_5-X_7.

Problem	Numerator curvature
X_5	$\begin{cases} -5x_1^2-6x_2^2-3x_3^2-5x_4^2-7x_5^2-2x_1x_2-4x_1x_3-2x_1x_4-2x_2x_3+2x_2x_5 \\ \quad -2x_3x_4+9x_1-10x_2-6x_3+8x_4-8x_5\le 1320, \\ -4x_1^2-3x_2^2-2x_3^2-3x_4^2-5x_5^2-2x_1x_2-2x_1x_3+2x_1x_5-4x_2x_3-2x_2x_4 \\ \quad -2x_3x_4+4x_4x_5-9x_1-6x_2-4x_4-x_5\le 1250, \\ -4x_1^2-5x_2^2-4x_3^2-5x_4^2-3x_5^2-4x_1x_4-2x_3x_4-2x_3x_5 \\ \quad -4x_4x_5-4x_1-9x_2-8x_3+3x_4-3x_5\le 950, \\ -4x_1^2-4x_2^2-3x_3^2-6x_4^2-8x_5^2+4x_1x_2-2x_1x_4+2x_2x_3 \\ \quad +2x_2x_4-4x_3x_4+2x_4x_5-x_1-5x_2-6x_3-4x_4+5x_5\le 1000, \\ 1\le x_i\le 20,\quad x_i\in\mathbb{Z},\quad \forall\, i=1,2,3,4,5. \end{cases}$
X_6	$\begin{cases} 2x_1^2+8x_2^2+5x_3^2+2x_4^2+4x_5^2-4x_1x_2-4x_1x_3+4x_1x_5-2x_2x_3 \\ \quad -2x_2x_4-8x_3x_4-2x_3x_5+4x_4x_5\le 378, \\ -2x_1^2-3x_2^2-4x_3^2-5x_4^2-3x_5^2-4x_1x_3-4x_1x_4-2x_1x_5-2x_2x_3 \\ \quad -2x_2x_4-2x_3x_4+2x_3x_5+2x_4x_5\le 216, \\ 3x_1^2-x_2^2+5x_3^2+4x_4^2+4x_5^2+2x_1x_2+2x_1x_4+2x_1x_5+2x_2x_4 \\ \quad +2x_2x_5-2x_3x_4-4x_3x_5\le 1511, \\ 2x_1^2+x_2^2-x_3^2-x_4^2-3x_5^2-2x_1x_2-2x_2x_3-2x_2x_4+10x_4x_5 \\ \quad +x_1+4x_2-6x_3+8x_4-6x_5\le 671, \\ 1\le x_i\le 100,\quad x_i\in\mathbb{Z},\quad \forall\, i=1,2,3,4,5. \end{cases}$
X_7	$\begin{cases} 9x_1^2+10x_1x_2+8x_2^2+5x_3^2+6x_1x_3+10x_2x_3+7x_4^2+10x_1x_4 \\ \quad +6x_2x_4+2x_3x_4+2x_2x_5+7x_5^2\le 1430, \\ 6x_1^2+8x_1x_2+6x_2^2+4x_3^2+2x_1x_3+2x_2x_3+8x_4^2 \\ \quad -2x_1x_4-10x_2x_4+2x_1x_5+6x_2x_5-6x_4x_5+7x_5^2\le 1150, \\ 9x_1^2+6x_2^2+8x_3^2-2x_1x_2-2x_2x_3+6x_4^2-4x_1x_4 \\ \quad -4x_2x_4+2x_3x_4+6x_1x_5+2x_2x_5-4x_4x_5+6x_5^2\le 850, \\ 4x_1^2+5x_2^2+8x_3^2+6x_4^2-2x_1x_2+6x_1x_3+2x_1x_4 \\ \quad -6x_2x_4+2x_3x_4+4x_1x_5-2x_2x_5+6x_3x_5+7x_5^2+8x_4x_5\le 1030, \\ 1\le x_i\le 200,\quad x_i\in\mathbb{Z},\quad \forall\, i=1,2,3,4,5. \end{cases}$

Problem 16.

$$\begin{cases} \min\ \dfrac{x^TQ_0x+c_0^Tx+d_0}{x^THx+h^Tx+q}, \\ \text{s.t. } x^TQ_jx+c_j^Tx+d_j\le 0, \qquad j=1,2, \\ \quad Ax\le b,\quad x\ge 0,\quad x_i\in\mathbb{Z},\quad i\in\{1,2,\ldots,n\} \end{cases}$$

where Q_0, Q_1, Q_2, H are $n\times n$ real symmetric matrices. A is an $m\times n$ real matrix, the elements of Q_0, H, vectors c_0, h and real number d_0, q are randomly generated in interval $[0, 10]$, the elements of Q_1, Q_2 and vectors c_1, c_2 are randomly generated in interval $[-10, 10]$, the entries of matrix A are randomly generated in the range $[0, 10]$, d_1, d_2 are randomly generated between -10 and 0, the elements of vector b are randomly generated in $[0, 100]$. For this problem, we tested 20 groups of instances with different sizes. For each group (m, n) in Table 4.9, we randomly generate 15 instances by using Problem 16, so a total of 300 instances are required to be solved.

Table 4.9 lists the numerical results of our algorithm BBA and two software packages (SCIP and BARON). Firstly, the global optimal values found by these three methods are close to each other, which again shows the effectiveness of BBA. Secondly, it can be observed that, with the

Table 4.5: Properties of objective functions and constraints of the Problems 1-14.

Problem	Numerator curvature	Denominator curvature	Constraints curvature
1	concave	convex	linear
2	convex	concave	linear
3	convex	concave	convex
4	convex	convex	convex
5	concave	convex	convex
6	concave	concave	convex
7	concave	concave	concave
8	convex	convex	concave
9	convex	concave	concave
10	concave	convex	concave
11	convex	convex	indefinite
12	convex	concave	indefinite
13	concave	convex	indefinite
14	concave	concave	indefinite

exception of random instances with $(m, n) = (20, 200), (20, 250)$, the larger the scale of problem, the more iterations and CPU running time the three algorithms require. Thirdly, from the first 11 groups of numerical results, the time taken of BBA is about $1.2-2.1$ times that of SCIP, while the number of iterations of the former is about $1.9 - 48.5$ times that of the latter. However, as the size of the problem increases, the advantage of BBA will gradually become apparent even though it still has far more iterations than SCIP. Depending on the experimental results of the last 9 groups, SCIP takes more time than BBA and fails to solve the last six groups of problems in less than 3600 seconds. Finally, BBA requires more iterations than BARON except for problems with $(m, n) = (5, 20)$, $(5, 30)$, $(5, 40)$ and $(5, 50)$, while BBA consumes less time than BARON except for problems with $(m, n) = (5, 5)$, $(5, 10)$, $(10, 20)$, $(10, 30)$ and $(20, 50)$. Significantly, BARON was also unable to solve the last five sets of problems in 3600 seconds. As a result, our algorithm can handle specific medium-scale problems faster than SCIP and BARON. Of course, for certain small-scale problems (e.g. $(m, n) = (5, 20)$, $(5, 30)$, $(5, 40)$ and $(5, 50)$), our algorithm can perform better than BARON.

To sum up, our algorithm is effective and robust, suitable for solving Problem 16 and four varieties of problem MIQCQFP, and can address Problem 16 with at least 700 dimensions in 3600 seconds.

5. Concluding Remarks

In this study, we propose a new global optimization algorithm for solving mixed integer quadratically constrained quadratic fractional programming problems. This problem extends both continuous quadratic fractional programming and pure integer quadratic fractional programming. By taking advantage of the simple property of the multiplication of amatrix and a vector, the numerator, denominator, and quadratic constraints of the problem are transformed into the bilinear form. Taking into account this bilinear structure, we establish a linear fractional relaxation programming problem, and then the rest of our algorithm is to address the general linear fractional programming problem. To enhance the efficiency of the algorithm,

Table 4.6: Comparison of results in Problems 1-7.

Problem	Methods	N_I	Solution	Optimum	Iter	Time
1	Dinkelbach	$\emptyset$	(50.5,99.0,0.0,99.0,59.8)	0.0131	2	0.0817
	BARON	$\emptyset$	(50.5,99.0,0.0,99.0,59.8)	0.0131	2	0.2457
	SCIP	$\emptyset$	(50.5,99.0,0.0,99.0,59.8)	0.0131	13	0.1225
	ours	$\emptyset$	(50.5,99.0,0.0,99.0,59.8)	0.0131	20	0.2126
	BARON	$\{1,2,3,4,5\}$	(50,99,0,99,59)	0.0147	15	0.2149
	SCIP	$\{1,2,3,4,5\}$	(50,99,0,99,59)	0.0147	8	0.1053
	ours	$\{1,2,3,4,5\}$	(50,99,0,99,59)	0.0147	21	0.2176
2	Dinkelbach	$\emptyset$	(99.0,0.0,0.0,0.0,0.0)	-0.9999	5	0.2408
	BARON	$\emptyset$	(99.0,0.0,0.0,0.0,0.0)	-0.9999	147	1.3051
	SCIP	$\emptyset$	(99.0,0.0,0.0,0.0,0.0)	-0.9999	9181	39.6775
	ours	$\emptyset$	(98.9924,0.0,0.0,0.0,0.0)	-0.9999	68	0.6471
	BARON	$\{1,3,5\}$	(99,0.0,0,0.0,0)	-0.9999	51	1.2511
	SCIP	$\{1,3,5\}$	(99,0.0,0,0.0,0)	-0.9999	1658	3.8786
	ours	$\{1,3,5\}$	(99,0.0,0,0.0,0)	-0.9999	45	0.4589
	BARON	$\{1,3,5,4,5\}$	(99,0,0,0,0)	-0.9999	5	0.4315
	SCIP	$\{1,2,3,4,5\}$	(99,0,0,0,0)	-0.9999	455	1.2033
	ours	$\{1,2,3,4,5\}$	(99,0,0,0,0)	-0.9999	38	0.3978
3	Dinkelbach	$\emptyset$	(2.0036,6.2239,2.9595)	0.00018	3	0.2847
	BARON	$\emptyset$	(2.0041,6.2199,2.9605)	0.00018	1390	2.6331
	SCIP	$\emptyset$	(2.0036,6.2239,2.9595)	0.00018	21	0.2669
	ours	$\emptyset$	(2.0238,6.2173,2.9351)	0.00018	47	0.4063
	BARON	$\{1,2,3\}$	(2,6,3)	0.00020	5	0.8401
	SCIP	$\{1,2,3\}$	(2,6,3)	0.00020	2	0.1347
	ours	$\{1,2,3\}$	(2,6,3)	0.00020	29	0.2476
4	Dinkelbach	$\emptyset$	(1.0000,1.0000,1.0000,7.4821)	1.9087	4	0.6405
	BARON	$\emptyset$	(1.0000,1.0000,1.0000,7.4821)	1.9087	3	1.0681
	SCIP	$\emptyset$	(1.0000,1.0000,1.0000,7.4821)	1.9087	41	0.2567
	ours	$\emptyset$	(1.0000,1.0000,1.0000,7.4821)	1.9087	80	0.8243
	BARON	$\{1,2,3,4\}$	(1,1,1,7)	1.95604	7	1.0012
	SCIP	$\{1,2,3,4\}$	(1,1,1,7)	1.95604	5	0.1507
	ours	$\{1,2,3,4\}$	(1,1,1,7)	1.95604	58	0.5536
5	Dinkelbach	$\emptyset$	(6.1491,1.0000,1.0000)	138.78045	12	3.5731
	BARON	$\emptyset$	(6.1491,1.0000,1.0000)	138.78045	1835	2.5305
	SCIP	$\emptyset$	(6.1491,1.0000,1.0000)	138.78045	504	0.2673
	ours	$\emptyset$	(6.1491,1.0000,1.0000)	138.78045	39	0.4161
	BARON	$\{1,2,3\}$	(6,1,1)	141.87469	61	0.3741
	SCIP	$\{1,2,3\}$	(6,1,1)	141.87469	778	0.2933
	ours	$\{1,2,3\}$	(6,1,1)	141.87469	23	0.2587
6	Dinkelbach	$\emptyset$	(1.00,1.5797,8.6188,5.9170,1.00)	0.2847	3	1.0107
	BARON	$\emptyset$	(1.00,1.5795,8.6188,5.9170,1.00)	0.2847	77	2.5617
	SCIP	$\emptyset$	(1.00,1.5795,8.6188,5.9170,1.00)	0.2847	605	1.0998
	ours	$\emptyset$	(1.00,1.5769,8.6187,5.9164,1.00)	0.2847	909	10.0894
	BARON	$\{1,2,3,4,5\}$	(1,1,8,6,1)	0.2907	37	1.2665
	SCIP	$\{1,2,3,4,5\}$	(1,1,8,6,1)	0.2907	49	0.3152
	ours	$\{1,2,3,4,5\}$	(1,1,8,6,1)	0.2907	251	2.4681
7	Dinkelbach	$\emptyset$	(1.00,2.7544,20.00,14.0398,1.00)	0.2421	4	0.9504
	BARON	$\emptyset$	(1.00,2.7548,20.00,14.0398,1.00)	0.2421	239	1.4847
	SCIP	$\emptyset$	(1.00,2.7548,20.00,14.0398,1.00)	0.2421	377	0.9706
	ours	$\emptyset$	(1.00,2.7412,20.00,14.0428,1.00)	0.2421	1346	13.8745
	BARON	$\{1,2,3,4,5\}$	(1,3,20,14,1)	0.2422	33	0.8574
	SCIP	$\{1,2,3,4,5\}$	(1,3,20,14,1)	0.2422	74	0.2628
	ours	$\{1,2,3,4,5\}$	(1,3,20,14,1)	0.2422	572	4.7131

Table 4.7: Comparison of results in Problems 8-14.

Problem	Methods	N_I	Solution	Optimum	Iter	Time
8	Dinkelbach	$\emptyset$	(1.00,1.00,1.00,1.00,20.00)	1.4789	5	0.2689
	BARON	$\emptyset$	(1.00,1.00,1.00,1.00,20.00)	1.4789	41	0.3476
	SCIP	$\emptyset$	(1.00,1.00,1.00,1.00,20.00)	1.4789	813	3.3611
	ours	$\emptyset$	(1.00,1.00,1.00,1.00,20.00)	1.4789	737	6.3781
	BARON	$\{1,2,3,4,5\}$	(1,1,1,1,20)	1.4789	7	0.2912
	SCIP	$\{1,2,3,4,5\}$	(1,1,1,1,20)	1.4789	648	2.2879
	ours	$\{1,2,3,4,5\}$	(1,1,1,1,20)	1.4789	441	4.3374
9	Dinkelbach	$\emptyset$	(1.0,1.0,17.7665,20.0,1.0)	-2.8727	7	1.3542
	BARON	$\emptyset$	(1.0,1.0,17.7664,20.0,1.0)	-2.8727	6539	26.1264
	SCIP	$\emptyset$	(1.0,1.0,17.7664,20.0,1.0)	-2.8727	5223	19.3193
	ours	$\emptyset$	(1.0,1.0,17.7665,20.0,1.0)	-2.8727	3977	40.4857
	BARON	$\{1,2,3,4,5\}$	(1,1,18,20,1)	-2.8724	15558	57.9524
	SCIP	$\{1,2,3,4,5\}$	(1,1,18,20,1)	-2.8724	3736	7.4178
	ours	$\{1,2,3,4,5\}$	(1,1,18,20,1)	-2.8724	2031	18.7865
10	Dinkelbach	$\emptyset$	(1.00,1.00,1.00,20.00,1.00)	-0.6331	5	0.2348
	BARON	$\emptyset$	(1.00,1.00,1.00,20.00,1.00)	-0.6331	29	1.1565
	SCIP	$\emptyset$	(1.00,1.00,1.00,20.00,1.00)	-0.6331	157	0.9143
	ours	$\emptyset$	(1.00,1.00,1.00,20.00,1.00)	-0.6331	148	1.5433
	BARON	$\{1,2,3,4,5\}$	(1,1,1,20,1)	-0.6331	39	1.0801
	SCIP	$\{1,2,3,4,5\}$	(1,1,1,20,1)	-0.6331	103	0.6961
	ours	$\{1,2,3,4,5\}$	(1,1,1,20,1)	-0.6331	94	0.8521
11	Dinkelbach	$\emptyset$	(1.00,1.00,1.00,1.00,10.8240)	1.6061	4	0.4507
	BARON	$\emptyset$	(1.00,1.00,1.00,1.00,10.8240)	1.6061	13	1.1555
	SCIP	$\emptyset$	(1.00,1.00,1.00,1.00,10.8240)	1.6061	321	1.3854
	ours	$\emptyset$	(1.00,1.00,1.00,1.00,10.8239)	1.6061	183	1.9868
	BARON	$\{1,2,3,4,5\}$	(1,1,1,1,10)	1.6328	27	1.5688
	SCIP	$\{1,2,3,4,5\}$	(1,1,1,1,10)	1.6328	50	0.1805
	ours	$\{1,2,3,4,5\}$	(1,1,1,1,10)	1.6328	165	1.6907
12	Dinkelbach	$\emptyset$	(1.00,1.00,1.00,1.00,1.00)	0.0174	2	0.0658
	BARON	$\emptyset$	(1.00,1.00,1.00,1.00,1.00)	0.0174	15945	63.2852
	SCIP	$\emptyset$	–	–	–	–
	ours	$\emptyset$	(1.00,1.00,1.00,1.00,1.00)	0.0174	82	0.6964
	BARON	$\{1,2,3,4,5\}$	(1,1,1,1,1)	0.0174	5363	14.7129
	SCIP	$\{1,2,3,4,5\}$	(1,1,1,1,1)	0.0174	40518	19.6671
	ours	$\{1,2,3,4,5\}$	(1,1,1,1,1)	0.0174	77	0.6873
13	Dinkelbach	$\emptyset$	(1.00,1.00,1.00,10.3919,1.00)	-0.5811	3	0.3582
	BARON	$\emptyset$	(1.00,1.00,1.00,10.3919,1.00)	-0.5811	23	1.1289
	SCIP	$\emptyset$	(1.00,1.00,1.00,10.3919,1.00)	-0.5811	71	0.5104
	ours	$\emptyset$	(1.00,1.00,1.00,10.3919,1.00)	-0.5811	125	1.2657
	BARON	$\{1,2,3,4,5\}$	(1,1,1,10,1)	-0.5762	33	1.2849
	SCIP	$\{1,2,3,4,5\}$	(1,1,1,10,1)	-0.5762	57	0.2739
	ours	$\{1,2,3,4,5\}$	(1,1,1,10,1)	-0.5762	87	0.8939
14	Dinkelbach	$\emptyset$	(3.1597,1.00,1.00,1.00,1.00)	-0.2660	5	0.2832
	BARON	$\emptyset$	(3.1597,1.00,1.00,1.00,1.00)	-0.2660	225	2.4403
	SCIP	$\emptyset$	(3.1597,1.00,1.00,1.00,1.00)	-0.2660	513	2.2057
	ours	$\emptyset$	(3.1599,1.00,1.00,1.00,1.00)	-0.2660	629	6.5334
	BARON	$\{1,2,3,4,5\}$	(3,1,1,1,1)	-0.2653	831	5.3614
	SCIP	$\{1,2,3,4,5\}$	(3,1,1,1,1)	-0.2653	368	0.8281
	ours	$\{1,2,3,4,5\}$	(3,1,1,1,1)	-0.2653	489	5.1104

Table 4.8: Numerical results for Problem 15 compared with OSBBA, BRA, SCIP and BARON.

(p,m,n)	BBA		BRA		OSBBA		SCIP		BARON	
	Avg.iter	Avg.time	Avg.iter	Avg.time	Avg.iter	Avg.time	Avg.iter	Avg.time	Avg.iter	Avg.time
(5,10,10)	45.6667	0.6629	36.6667	0.5279	2574.0667	28.0407	11.5333	0.3214	24.7333	0.2727
(5,10,20)	293.5333	4.0133	78.5333	1.3228	17785.9333	207.4884	19.5333	0.6064	20.2000	0.2920
(5,20,20)	164.3333	2.2988	72.0667	1.3572	3030.4000	33.2958	19.1333	0.8841	54.1333	0.4668
(5,30,20)	94.7333	1.3694	22.3333	0.7423	1658.6667	19.8844	5.9333	0.4810	21.9333	0.1453
(5,30,50)	13219.4667	182.1351	5342.6000	67.3081	18843.6000	267.4946	22.7333	4.8601	51.5333	0.5253
(5,50,50)	974.8667	13.3334	87.6000	2.3797	1055.6000	20.5542	17.8000	8.2939	63.5333	0.7327
(5,100,100)	5648.6000	268.3961	593.8667	27.5891	1088.4667	81.0076	11.9333	74.4139	106.8000	1.5747
(5,100,200)	10271.4667	849.3485	1160.5333	274.6944	1634.4000	178.7851	37.5333	791.3414	196.7333	8.6647
(10,100,100)	9629.3333	354.8235	238.1333	16.1573	7635.0667	748.4875	29.9333	65.4982	473.8000	6.5833
(10,100,200)	18351.8000	1529.5781	1840.6667	165.8318	5796.1333	800.4769	63.1333	777.9274	1503.3333	45.3067
(20,100,100)	8827.0000	225.7681	337.9333	19.0229	9560.2667	1269.9644	60.0667	71.3325	3645.7333	163.6533
(5,100,10)	18.0000	0.4779	2.5333	0.2716	70.2000	1.4560	1.1333	0.0509	10.6000	0.0893
(10,100,10)	32.7333	0.5273	8.3333	0.3439	994.2000	29.4666	1.2667	0.0780	7.9333	0.1103
(20,100,10)	27.1333	0.4355	4.3333	0.2860	1158.7333	33.3676	1.1333	0.0857	7.0000	0.1213
(30,100,10)	18.5333	0.3172	1.6000	0.2560	2302.6667	69.0055	1.4000	0.1082	6.5333	0.1814
(50,100,10)	17.5333	0.2990	1.4667	0.2612	2832.2000	110.6273	2.6000	0.1568	12.4667	0.2527
(80,100,10)	20.0667	0.3351	2.6000	0.2747	4362.6667	206.9593	1.4000	0.2347	10.2000	0.3781
(100,100,10)	38.0667	0.5751	6.8667	0.2987	9100.0667	467.2352	1.1333	0.2953	27.0000	11.9787
(200,100,10)	33.0667	0.5365	6.9333	0.3464	80101.5333	2880.5421	1.4000	0.9855	42.8667	21.3873
(300,100,10)	34.6667	0.5345	5.9333	0.3209	–	–	1.3333	2.5525	105.1333	148.2053
(500,100,10)	29.2667	0.4813	6.1333	0.3299	–	–	1.4000	10.8483	138.8667	716.5332

Table 4.9: Numerical results for Problem 16 compared with SCIP and BARON.

(m,n)	BBA			SCIP			BARON		
	Avg.iter	Avg.time	Opt.val	Avg.iter	Avg.time	Opt.val	Avg.iter	Avg.time	Opt.val
(5,5)	37.7333	0.3729	0.879822	15.7333	0.2627	0.879822	17.7333	0.3187	0.879822
(5,10)	79.1333	0.8936	0.939956	34.6667	0.4481	0.939956	64.6667	0.8347	0.939956
(5,20)	267.6000	1.6023	0.965676	80.6667	1.3274	0.965676	624.4000	7.0347	0.965676
(5,30)	503.8667	5.6721	0.960491	138.6667	3.0068	0.960491	668.0667	14.8752	0.960543
(5,40)	882.3333	11.4537	0.982090	463.5333	6.3554	0.982090	2112.5333	78.3750	0.982090
(5,50)	1351.8667	18.7243	0.993946	370.8000	10.8044	0.993946	4179.0667	315.8417	0.993946
(10,20)	110.2000	1.4683	0.989525	20.6667	0.7808	0.989525	22.4667	1.0253	0.989525
(10,30)	218.1333	3.2277	0.990251	25.0667	1.6721	0.990251	63.6000	2.4847	0.990251
(10,40)	412.0667	6.4886	0.992798	48.0667	3.7727	0.992798	293.8000	17.1160	0.992798
(20,50)	547.0667	9.2630	0.998811	11.9333	5.7336	0.998811	26.5333	6.9246	0.998811
(20,60)	694.4667	14.3036	0.999540	14.3333	8.3332	0.999540	31.1333	19.3853	0.999540
(20,80)	886.4667	20.7495	0.999712	34.4000	21.5692	0.999712	61.9333	24.3907	0.999712
(20,100)	1841.8667	48.5929	0.999532	56.7333	58.4821	0.999532	225.3333	53.5538	0.999532
(20,200)	2988.3333	313.0292	0.999831	127.3333	1556.3246	0.999825	172.0000	899.3313	0.999843
(20,250)	2767.8667	364.1949	0.999905	–	–	–	35.8667	1587.7987	0.999905
(20,300)	2732.1333	628.8606	0.999935	–	–	–	–	–	–
(20,400)	3151.6000	1250.0636	0.999952	–	–	–	–	–	–
(20,500)	4958.1333	2088.1734	0.999975	–	–	–	–	–	–
(20,600)	5551.0667	2654.3681	0.999981	–	–	–	–	–	–
(20,700)	6542.0667	3566.0278	0.999991	–	–	–	–	–	–

the rectangular adjustment-segmentation technique and the midpoint sampling strategy are integrated into the branch-and-bound framework. Numerical results show that the proposed algorithm can be employed to solve problems SLFP, MLFP, LMP and MIQCQP. In addition, this algorithm can solve small and medium-scale MIQCQFP problems, and can address specific medium-sized problems faster than SCIP and BARON. For solving large-scale MIQCQFP problems, techniques that can be integrated into this algorithm are currently being studied.

Moreover, it can be seen from Remark 2.1 that when the parameters λ_j and η_{ij} are set to 1 and 0 respectively, the constructed relaxation problem is obtained. However, the parameters λ_j and η_{ij} can also be selected within the interval $[0, 1]$, so a useful parameter selection method may encourage us to construct a better relaxation problem for the original problem, which may be particularly important for improving the performance of the algorithm. In the future, we will also focus on the selection of parameter vectors in order to give a more appropriate parameter selection scheme.

Acknowledgements. This research is supported by the National Natural Science Foundation of China (Grant 11961001), by the Construction Project of first-class subjects in Ningxia higher Education (NXYLXK2017B09) and by the major proprietary funded project of North Minzu University (ZDZX201901).

References

[1] A. Beck and M. Teboulle, A convex optimization approach for minimizing the ratio of indefinite quadratic functions over an ellipsoid, *Math. Program.*, **118**:1 (2009), 13–35.

[2] A. Beck and M. Teboulle, On minimizing quadratically constrained ratio of two quadratic functions, *J. Convex Anal.*, **17**:1 (2010), 789–804.

[3] I. Bomze and P. Amaral, Copositivity-based approximations for mixed-integer fractional quadratic optimization, *Pac. J. Optim.*, **11**:2 (2014), 225–238.

[4] I. Bomze, M. Locatelli, and F. Tardella, New and old bounds for standard quadratic optimization: Dominance, equivalence and incomparability, *Math. Program.*, **115**:1 (2008), 31–64.

[5] I. Bomze and M. Overton, Narrowing the difficulty gap for the Celis-Dennis-Tapia problem, *Math. Program.*, **151**:2 (2015), 459–476.

[6] A. Charnes and W. Cooper, Programming with linear fractional functionals, *Nav. Res. Logist.*, **10**:1 (1963), 273–274.

[7] A. Del Pia, On approximation algorithms for concave mixed-integer quadratic programming, *Math. Program.*, **172**:1-2 (2018), 3–16.

[8] W. Dinkelbach, On non-linear fractional programming, *Manage. Sci.*, **13**:7 (1967), 492–498.

[9] S. Fallahi and M. Salahi, On the indefinite quadratic fractional optimization with two quadratic constraints, *J. Optim. Theory Appl.*, **162**:1 (2014), 249–256.

[10] L. Galli and A.N. Letchford, A binarisation heuristic for non-convex quadratic programming with box constraints, *Oper. Res. Lett.*, **46**:5 (2018), 529–533.

[11] Y. Gao and F. Wei, A new bound-and-reduce approach of nonconvex quadratic programming problems, *Appl. Math. Comput.*, **250** (2015), 298–308.

[12] A. Gleixner et al., The SCIP Optimization Suite 5.0, `https://www.scipopt.org/`.

[13] F. Glover, Improved linear integer programming formulations of nonlinear integer problems, *Manage. Sci.*, **22**:4 (1975), 455–460.

[14] N. Gould and P. Toint, Numerical methods for large-scale non-convex quadratic programming, in: *Trends in Industrial and Applied Mathematics*, Springer, (2002), 149–179.

[15] T. Ibaraki, H. Ishii, J. Iwase, T. Hasegawa, and H. Mine, Algorithms for quadratic fractional programming problems, *J. Oper. Res. Soc. Japan*, **19**:2 (1976), 174–191.

[16] E. Jain, K. Dahiya, and V. Verma, Integer quadratic fractional programming problems with bounded variables, *Eur. J. Oper. Res.*, **269**:1-2 (2018), 269–295.

[17] V. Jeyakumar and G. Li, Trust-region problems with linear inequality constraints: Exact SDP relaxation, global optimality and robust optimization, *Math. Program.*, **147**:1-2 (2014), 171–206.

[18] H. Jiao and S. Liu, A new linearization technique for minimax linear fractional programming, *Int. J. Comput. Math.*, **91**:7-8 (2014), 1730–1743.

[19] H. Jiao and S. Liu, A practicable branch and bound algorithm for sum of linear ratios problem, *Eur. J. Oper. Res.*, **243**:3 (2015), 723–730.
[20] A. Khurana and S. Arora, An algorithm for solving quadratic fractional program with linear homogeneous constraints, *Croat. Oper. Res. Rev.*, **39**:4 (2011), 391–404.
[21] S. Kim and M. Kojima, Exact solutions of some nonconvex quadratic optimization problems via SDP and SOCP relaxations, *Comput. Optim. Appl.*, **26**:2 (2003), 143–154.
[22] S. Liu and L. Ge, An outcome space algorithm for minimizing a class of linear ratio optimization problems, *Comput. Appl. Math.*, **40** (2021), 225.
[23] X. Liu, Y.L. Gao, B. Zhang, and F.P. Tian, A new global optimization algorithm for a class of linear fractional programming, *Mathematics*, **7**:9 (2019), 867.
[24] A. Lo and A. Mackinlay, Maximizing predictability in the stock and bond markets, *Macroecon. Dyn.*, **1**:1 (1995), 102–134.
[25] V.-B. Nguyen, R.-L. Sheu, and Y. Xia, An SDP approach for quadratic fractional problems with a two-sided quadratic constraint, *Optim. Methods Softw.*, **31**:4 (2016), 701–719.
[26] J. Ohlson, The asymptotic validity of quadratic utility as the trading interval approaches zero, in: *Stochastic Optimization Models in Finance*, **1**:1 (1975), 221–234.
[27] B. Ozkok, An iterative algorithm to solve a linear fractional programming problem, *Comput. Ind. Eng.*, **140** (2020), 106234.
[28] P. Pandian and M. Jayalakshmi, On solving linear fractional programming problems, *Math. Models Methods Appl. Sci.*, **7**:6 (2013), 90–100.
[29] P. Pardalos and G. Schnitger, Checking local optimality in constrained quadratic programming is NP-hard, *Oper. Res. Lett.*, **7**:1 (1988), 33–35.
[30] N. Phuong and H. Tuy, A unified monotonic approach to generalized linear fractional programming, *J. Glob. Optim.*, **26**:3 (2003), 229–259.
[31] N. Sahinidis, Baron user manual v.21.1.13, `http://minlp.com`.
[32] M. Salahi and S. Fallahi, Parametric approach for solving quadratic fractional optimization with a linear and a quadratic constraint, *Comput. Appl. Math.*, **35**:2 (2016), 439–446.
[33] V. Seerengasamy and K. Jeyaraman, A new approach for easy computation by using θ-matrix for solving integer linear fractional programming problems, *Int. J. Comput. Appl.*, **69**:7 (2013), 26–30.
[34] P. Shen and B. Huang, Global algorithm for solving linear multiplicative programming problems, *Optim. Lett.*, **14**:3 (2020), 693–710.
[35] P. Shen, B. Huang, and L. Wang, Range division and linearization algorithm for a class of linear ratios optimization problems, *J. Comput. Appl. Math.*, **350** (2019), 324–342.
[36] P. Shen, K. Wang, and T. Lu, Outer space branch and bound algorithm for solving linear multiplicative programming problems, *J. Glob. Optim.*, **78**:3 (2020), 453–482.
[37] M. Sivri, I. Albayrak, and G. Temelcan, A novel approach for solving quadratic fractional programming problems, *Croat. Oper. Res. Rev.*, **9**:2 (2018), 199–209.
[38] N. Suleimann and M. Nawkhass, Solving quadratic fractional programming problem, *Int. J. Appl. Math. Res.*, **2**:2 (2013), 303–309.
[39] K. Swarup, Some aspects of linear fractional programming, *Aust. N. Z. J. Stat.*, **7**:3 (2008), 90–104.
[40] S. Tantawy, Using feasible direction linear programming problems, *Aust. J. Basic Appl. Sci.*, **1**:2 (2007), 109–114.
[41] C. Tofallis, Fractional programming: Theory, methods and applications, *J. Oper. Res. Soc.*, **49**:8 (1998), 895.
[42] W. Xia, J. Vera, and L. Zuluaga, Globally solving nonconvex quadratic programs via linear integer programming techniques, *INFORMS J. Comput.*, **32**:1 (2020), 40–56.
[43] R. Yamamoto and H. Konno, An efficient algorithm for solving convexconvex quadratic fractional programs, *J. Optim. Theory Appl.*, **133**:2 (2007), 241–255.
[44] D. Yue, G.G. Gosalbez, and F. You, Global optimization of large-scale mixedinteger linear frac-

tional programming problems: A reformulation-linearization method and process scheduling applications, *AIChE J.*, **59**:11 (2013), 4255–4272.

[45] A. Zhang and S. Hayashi, Celis-Dennis-Tapia based approach to quadratic fractional programming problems with two quadratic constraints, *Numer. Algebra, Control. Optim.*, **1**:1 (2011), 83–98.

[46] Y. Zhao and S. Liu, Global optimization algorithm for mixed integer quadratically constrained quadratic program, *J. Comput. Appl. Math.*, **319** (2017), 159–169.

[47] Y. Zhao, S. Liu, and H. Jiao, A new branch and bound algorithm for minimax ratios problems, *Open Math.*, **15**:1 (2017), 840–851.