Research Article

An Efficient Algorithm for Solving a Class of Fractional Programming Problems

Yongjian Qiu,1,2 Yuming Zhu,1 and Jingben Yin3

1School of Management, Northwestern Polytechnical University, Xi’an 710072, China
2School of Clinical, Xinxiang Medical University, Xinxiang 453003, China
3School of Mathematical Sciences, Henan Institute of Science and Technology, Xinxiang 453003, China

Correspondence should be addressed to Yongjian Qiu; 05091002@yu.edu.cn

Received 20 February 2021; Accepted 21 June 2021; Published 7 July 2021

Academic Editor: Leandro F. F. Miguel

Copyright ©2021YongjianQiuetal.-V_hisisanopenaccessarticledistributedundertheCreativeCommonsAttributionLicense,
whichpermitsunrestricteduse,distribution,andreproductioninanymedium,providedtheoriginalworkisproperlycited.

This paper presents an efficient branch-and-bound algorithm for globally solving a class of fractional programming problems, which are widely used in communication engineering, financial engineering, portfolio optimization, and other fields. Since the kind of fractional programming problems is nonconvex, in which multiple locally optimal solutions generally exist that are not globally optimal, so there are some vital theoretical and computational difficulties. In this paper, first of all, for constructing this algorithm, we propose a novel linearizing method so that the initial fractional programming problem can be converted into a linear relaxation programming problem by utilizing the linearizing method. Secondly, based on the linear relaxation programming problem, a novel branch-and-bound algorithm is designed for the kind of fractional programming problems, the global convergence of the algorithm is proved, and the computational complexity of the algorithm is analysed. Finally, numerical results are reported to indicate the feasibility and effectiveness of the algorithm.

1. Introduction

In this paper, we consider the following a class of fractional programming problems:

\[
\text{(FP): } \begin{cases} 
\max & g(x) = \sum_{j=1}^{p} f_{j_1}(x) / h_{j_1}(x) \\
\text{s.t.} & x \in D = \{x | Ax \leq b\}, 
\end{cases}
\]  

where \( p \) and \( m \) are all any arbitrary natural numbers and \( x \) is an \( N \)-dimensional variable, and for any \( x \in D \subset \mathbb{R}^N \), \( f_{j_1}(x) \) and \( h_{j_1}(x) (j = 1, \ldots, p \text{ and } t = 1, \ldots, m) \) are all affine functions such that \( f_{j_1}(x) > 0 \) and \( h_{j_1}(x) > 0 \). It should be pointed out that, in portfolio optimization, the variable \( x \) refers to the amount of investment; in computer vision, the variable \( x \) refers to the mapping space; in communication engineering, variable \( x \) refers to input signal.

During the past decades, as special cases of the (FP), the linear sum-of-ratios problem and linear multiplicative programming problem have attracted a huge attention of practitioners and researchers for many years. This is because the linear sum-of-ratios problem and linear multiplicative programming problem exist in very important applications such as chance optimization, portfolio optimization, engineering optimization, and data envelopment analysis [1]. In addition, the linear sum-of-ratios problem and linear multiplicative programming problem generally possess multiple local optima which are not globally optimum. The problem (FP) investigated in this paper can be looked as the extensions of the linear sum-of-ratios problem or linear multiplicative programming problem so that the problem (FP) has a broader of applications than the linear sum-of-ratios problem and the linear multiplicative programming problem, and it poses more complex theory and computational difficulties.

Over the two decades, a variety of algorithms have been developed for solving the special forms of the problem (FP). For example, for the linear sum-of-ratios problem, several algorithms can be obtained, such as simplex and parametric
simplex methods [2, 3], image space approach [4], branch-and-bound methods [5–12], trapezoidal algorithm [13, 14], and monotonic optimization algorithm [15]; for the linear multiplicative programming problem, some algorithms can be also found in literatures, such as branch-and-bound algorithms [16–18], polynomial time approximation algorithm [19], outcome space algorithms [20, 21], level set algorithm [22], heuristic method [23], and monotonic optimization algorithm [15]. Furthermore, Jiao et al. [24, 25] and Chen and Jiao [26] presented three different algorithms for solving the linear multiplicative programming problem. Recently, for the several kinds of special forms of the problem (FP), Huang et al. [27], Jiao et al. [12], and Wang and Zhang [28] presented three different global optimization algorithms for the sum of linear ratios’ problem; Jiao et al. [29] and Yin et al. [30] proposed two different outer space branch-and-bound algorithms for the generalized linear multiplicative programming problem; Jiao and Chen [31] and Jiao and Liu [32, 33] gave three branch-reduction-bound algorithms for solving the quadratically constrained quadratic programming problem and quadratically constrained sum of quadratic ratios’ problem; Jiao et al. [34, 35], Ghazi and Roubi [36], and Bennani et al. [37] presented four different algorithms for solving the generalized polynomial optimization problem and generalized linear fractional programming problem. In addition, several differential evolution algorithms [38–40], a novel gate resource allocation method using improved PSO-based QEA [41], and an enhanced MSIQDE algorithm with novel multiple strategies [42] are also proposed for solving the global optimization problems including the problem (FP). Up to today, although some researchers have proposed some algorithms for solving the linear sum-of-ratios problem, the linear multiplicative programming problem, or the special form of the problem (FP), to our knowledge, little work has been still done for globally the general form of the problem (FP) considered in this paper.

The purpose of this paper is to develop an effective algorithm for globally solve all variants of the problem (FP). First of all, based on equivalent transformation and characteristics of the exponential function and logarithmic function, a novel linearizing method is proposed. By utilizing the linearizing method, we can convert the original problem (FP) or its subproblem into a linear relaxation problem (LRP), whose solution can be infinitely close to the optimal solution of the problem (FP) by a successive refinement partition. Secondly, based on the branch-and-bound framework, a novel branch-and-bound algorithm is constructed for globally solving all variants of the problem (FP), and the global convergence of the proposed algorithm is proved. In addition, the computational complexity of the algorithm is analysed for the first time, and the maximum iterations of the algorithm are estimated for the first time. Finally, numerical experimental results demonstrate the feasibility and effectiveness of the proposed algorithm.

The remaining sections of this paper are organized as follows. A new linearizing method is constructed for deriving the problem (LRP) of the problem (FP) in Section 2. Based on the branch-and-bound scheme and the constructed problem (LRP), a global optimization algorithm is established in Section 3, and its convergence is derived. In Section 4, the computational complexity of the algorithm is analysed. In Section 5, numerical experiments are given to verify the feasibility and effectiveness of the proposed algorithm. Finally, some conclusions are given in Section 6.

2. Novel Linearizing Technique

In this section, we will present a new linearizing method for constructing the problem (LRP). For each $i = 1, \ldots, N$, we firstly solve the following two linear programming problems:

$$\begin{align*}
\hat{x}_i^0 &= \min_{x_i} \ x_i \\
&\text{subject to } Ax \leq b,
\end{align*}$$

We can get an initial rectangle $X_0 = [\bar{x}^0_0, \bar{x}^0_N] \subseteq X_0^0$, which contains the feasible region of the problem (FP).

Next, since $f_{\mu}(x) > 0$ and $h_{\mu}(x) > 0$ for all $x \in X_0^0$, it can easily follow that

$$g(x) = \sum_{j=1}^p \prod_{t=1}^m f_{\mu}(x) = \prod_{j=1}^p \exp \left( \sum_{t=1}^m \left( \ln(f_{j\mu}(x)) - \ln(h_{j\mu}(x)) \right) \right).$$

For any $x \in X = \{ x | \bar{x} \leq x \leq \bar{x} \} \subseteq X_0$, for each $j = 1, \ldots, p$ and $t = 1, \ldots, m$, we let

$$f_{j\mu}^t = \min_{x \in X} f_{j\mu}(x),$$

$$f_{j\mu}^u = \max_{x \in X} f_{j\mu}(x),$$

$$k_{j\mu}^1 = \frac{\ln(f_{j\mu}^u) - \ln(f_{j\mu}^t)}{f_{j\mu}^u - f_{j\mu}^t},$$

$$h_{j\mu}^t = \min_{x \in X} h_{j\mu}(x),$$

$$h_{j\mu}^u = \max_{x \in X} h_{j\mu}(x),$$

$$k_{j\mu}^2 = \frac{\ln(h_{j\mu}^u) - \ln(h_{j\mu}^t)}{h_{j\mu}^u - h_{j\mu}^t}.$$ 

By the characteristics of the logarithmic functions $\ln(f_{j\mu}(x))$ and $\ln(h_{j\mu}(x))$ over $X$, it can follow that

$$\ln(f_{j\mu}(x)) \leq k_{j\mu}^1 f_{j\mu}(x) - 1 - \ln(k_{j\mu}^1),$$

$$k_{j\mu}^2 h_{j\mu}(x) - (h_{j\mu}(x) + \ln(h_{j\mu}(x)) \leq \ln(h_{j\mu}(x)).$$

Then, by the above inequalities, we can follow that
For each \( j = 1, \ldots, p \) and \( t = 1, \ldots, m \), we let
\[
\varphi_j(x) = k^j_{f_j}(h^j_{f_j}(x)) - \ln\left(k^j_{f_j}(h^j_{f_j}(x)) - \ln(h^j_{f_j})\right) + \ln(h^j_{f_j})].
\] (7)

Thus, by equations (3)–(7), it follows that
\[
g(x) = \sum_{j=1}^{p} \exp\left(\sum_{t=1}^{m} \ln(f_j(x)) - \ln(h_j(x))\right) - \sum_{j=1}^{p} \exp\left(\sum_{t=1}^{m} \varphi_j(x)\right).
\] (8)

For any \( x \in X = \{x | x \leq x \leq \bar{x}\} \subseteq X^0 \), we let
\[
\varphi_j(x) = \sum_{t=1}^{m} \varphi_j(x),
\]
\[
\varphi^l_j = \min_{x \in X} \varphi_j(x),
\]
\[
\varphi^u_j = \max_{x \in X} \varphi_j(x),
\]
\[
K_j = \frac{\exp(\varphi^u_j) - \exp(\varphi^l_j)}{\varphi^u_j - \varphi^l_j}.
\] (9)

By the characters of the exponential function \( \exp(\varphi_j(x)) \) over \([\varphi^l_j, \varphi^u_j]\), we can get that
\[
\exp(\varphi_j(x)) \leq K_j(\varphi_j(x) - \varphi^l_j) + \exp(\varphi^l_j).
\] (10)

Therefore, we have
\[
\sum_{j=1}^{p} \exp(\varphi_j(x)) \leq \sum_{j=1}^{p} \left[K_j(\varphi_j(x) - \varphi^l_j) + \exp(\varphi^l_j)\right].
\] (11)

Let
\[
Ug(x) = \sum_{j=1}^{p} \left[K_j(\varphi_j(x) - \varphi^l_j) + \exp(\varphi^l_j)\right].
\] (12)

Combining equations (8)–(12), it can follow that
\[
g(x) \leq \sum_{j=1}^{p} \exp(\varphi_j(x)) \leq \sum_{j=1}^{p} \left[K_j(\varphi_j(x) - \varphi^l_j) + \exp(\varphi^l_j)\right] = Ug(x).
\] (13)

Consequently, based on the above discussions, we can establish the corresponding linear relaxation problem (LRP) of the problem (FP) over \( X \) as follows:

\[
\text{LRP}: \begin{cases}
\max & Ug(x) = \sum_{j=1}^{p} [K_j(\varphi_j(x) - \varphi^l_j) + \exp(\varphi^l_j)], \\
\text{s.t.} & Ax \leq b, x \in X,
\end{cases}
\] (14)

where \( \varphi_j(x), \varphi^l_j, \) and \( K_j, j = 1, 2, \ldots, p \), are given by (9).

Based on the constructing method of the linear relaxation problem (LRP), for any \( X^0 \subseteq X^0 \), the global optimal value of the problem (LRP) can offer a reliable upper bound for the global optimal value of the problem (FP) over \( X \).

Theorem 1 guarantees that \( Ug(x) \) will infinitely approximate \( g(x) \) as \( \|\bar{x} - x\| \to 0 \).

**Theorem 1.** For any \( x \in X = \{x | x \leq x \leq \bar{x}\} \subseteq X^0 \), we have
\[
\lim_{\|\bar{x} - x\| \to 0} Ug(x) - g(x) \to 0.
\] (15)

**Proof.** For any \( x \in X \), without loss of generality, let \( \Delta^1 = Ug(x) - \sum_{j=1}^{p} \exp(\varphi_j(x)) \) and \( \Delta^2 = \sum_{j=1}^{p} \exp(\varphi_j(x)) - g(x) \); we can obtain that
\[
Ug(x) - g(x) = \left[Ug(x) - \sum_{j=1}^{p} \exp(\varphi_j(x))\right] + \left[\sum_{j=1}^{p} \exp(\varphi_j(x)) - g(x)\right] = \Delta^1 + \Delta^2.
\] (16)

Therefore, to prove that \( \lim_{\|\bar{x} - x\| \to 0} Ug(x) - g(x) \to 0 \), we just prove that \( \lim_{\|\bar{x} - x\| \to 0} \Delta^1 \to 0 \) and \( \lim_{\|\bar{x} - x\| \to 0} \Delta^2 \to 0 \).

First of all, let \( \Delta^1_j = K_j(\varphi_j(x) - \varphi^l_j) + \exp(\varphi^l_j) - \exp(\varphi_j(x)) \); we have that
\[
\Delta^1 = Ug(x) - \sum_{j=1}^{p} \exp(\varphi_j(x))
\] (17)
\[
= \sum_{j=1}^{p} [K_j(\varphi_j(x) - \varphi^l_j) + \exp(\varphi^l_j)] - \sum_{j=1}^{p} \exp(\varphi_j(x))
\] (17)
\[
= \sum_{j=1}^{p} [K_j(\varphi_j(x) - \varphi^l_j) + \exp(\varphi^l_j) - \exp(\varphi_j(x))]
\] (17)
\[
= \sum_{j=1}^{p} \Delta^1_j.
\] (17)
Since
\[
\lim_{\|x\| \to 0} \Delta_j^{\max} \to 0.
\]
Therefore, by (17) and (20), we have
\[
\lim_{\|x\| \to 0} \Delta_j^1 = \lim_{\|x\| \to 0} \sum_{j=1}^P \Delta_j^1 \to 0.
\]
Secondly, let \( \phi_j(x) = \sum_{t=1}^P \ln(f_{jt}(x)) - \ln(h_{jt}(x)) \) and \( \Delta_j^2 = \exp(\phi_j(x)) - \exp(\phi_j(x)) \); we have
\[
\Delta_j^2 = \sum_{j=1}^P \exp(\phi_j(x)) - g(x)
\]
\[
= \sum_{j=1}^P \exp(\phi_j(x)) - \sum_{j=1}^P \exp\left( \sum_{t=1}^m [\ln(f_{jt}(x)) - \ln(h_{jt}(x))] \right)
\]
\[
= \sum_{j=1}^P [\exp(\phi_j(x)) - \exp(\phi_j(x))]
\]
\[
= \sum_{j=1}^P \Delta_j^2.
\]
Since
\[
\Delta_j^2 = \exp(\phi_j(x)) - \exp(\phi_j(x))
\]
\[
\leq \|\phi_j(x) - \phi_j(x)\| \sup_{\theta \in \mathbb{R} \setminus \{\phi_j(x), \phi_j(x)\}} \exp(\theta),
\]  
where
\[
L(\phi_j(x), \phi_j(x)) = \beta \phi_j(x) + (1 - \beta) \phi_j(x), \quad \beta \in [0, 1].
\]
For each \( j \in \{1, \ldots, T\} \), let
\[
\Delta_{jt,1} = k_{jt}^j(f_{jt}(x)) - 1 - \ln k_{jt}^j, \quad \Delta_{jt,2} = \ln(h_{jt}(x)) - k_{jt}^j(h_{jt}(x) - h_{jt}^j + \ln h_{jt}^j),
\]
and then, it follows that
\[
\varphi_j(x) - \varphi_j(x) = \sum_{t=1}^T \left[ \left[ k_{jt}^j(f_{jt}(x)) - 1 - \ln k_{jt}^j \right] - \ln(f_{jt}(x)) \right]
\]
\[
+ \sum_{t=1}^T \left[ \ln(h_{jt}(x)) - k_{jt}^j(h_{jt}(x) - h_{jt}^j) + \ln h_{jt}^j \right],
\]
\[
= \sum_{t=1}^T (\Delta_{jt,1} + \Delta_{jt,2}).
\]
Since the function \( \Delta_{jt,1}^2 \) is a convex function about \( (f_{jt}(x)) \), it can obtain the maximum value \( \Delta_{jt,1}^2 \) at the point \( f_{jt}^1 \) or \( f_{jt}^2 \). Then, through computing, we get
\[
\Delta_{jt,1}^{\max} = \frac{\ln(f_{jt}^1/f_{jt}^2)}{f_{jt}^1/f_{jt}^2} - 1 - \ln(f_{jt}^1/f_{jt}^2) - 1
\]
Since \( (f_{jt}^1/f_{jt}^2) \to 1 \) as \( \|x - \bar{x}\| \to 0 \), we have
\[
\Delta_{jt,1}^{\max} \to 0 \quad \text{as} \quad \|x - \bar{x}\| \to 0.
\]
Similarly, we can prove that
\[
\Delta_{jt,2}^{\max} \to 0 \quad \text{as} \quad \|x - \bar{x}\| \to 0.
\]
By (28) and (29), it follows that, as \( \|x - \bar{x}\| \to 0 \),
\[
\varphi_j(x) - \varphi_j(x) = \sum_{t=1}^m (\Delta_{jt,1} + \Delta_{jt,2}) \to 0.
\]
Since \( \exp(\theta) \) is a continuous and bounded function about \( x \), then there exists some \( \bar{M} > 0 \) such that \( \exp(\theta) \leq \bar{M} \).
Therefore, by (23) and (30), it follows that
\[
\Delta_j^2 \leq \bar{M}(\varphi_j(x) - \varphi_j(x)) \to 0 \quad \text{as} \quad \|x - \bar{x}\| \to 0.
\]
By (30) and (31), it follows that
\[
\Delta^2 = \sum_{j=1}^P \Delta_j^2 \to 0 \quad \text{as} \quad \|x - \bar{x}\| \to 0.
\]
By the above discussions, it is obvious that the conclusion can be followed.
From Theorem 1, it follows that \( U_g(x) \) will infinitely approximate \( g(x) \) as \( \|x - \bar{x}\| \to 0 \).

3. Algorithm and Its Global Convergence

In this section, based on the above linear relaxation problem, a branch-and-bound algorithm is proposed for globally solving the problem (FP). We firstly select a rectangle bisection technique of maximum edge, which ensures the global convergence of the present algorithm. The detailed branching
technique is described as follows. Suppose that $X^k = [x^k, x^k]\subseteq X^0$ is the selected rectangle for partitioning; let $\sigma \in \arg \max \{x^k - x^k : i = 1, 2, \ldots, N\}$; subdivide the interval $[x^k, x^k]$ into two subintervals $[x^k, (x^k + x^k)/2]$ and $[(x^k + x^k)/2, x^k]$ so that we can subdivide $X^k$ into two subrectangles. Next, by solving a sequence of linear relaxation problems (LRP) of the problem (FP), the upper bound of the optimal value of the problem (FP) can be updated. Moreover, by detecting the feasible points and computing the objective functional values of these feasible points, the lower bound of the optimal value of the problem (FP) can be updated.

3.1. The Proposed Branch-and-Bound Algorithm. Let $UB(X^k)$ and $x^k = x(X^k)$ be the optimal value and the optimal solution of the problem (LRP) over $X^k$, respectively. The corresponding branch-and-bound algorithm is given as follows.

Steps of the proposed branch-and-bound algorithm:

Step 1. Given the convergence error $\varepsilon > 0$ and the initial lower bound $LB_0 = -\infty$, solve the problem (LRP) over $X^0$ to get $x^0$ and $UB(X^0)$, respectively. Let $UB_0 = UB(X^0)$ and $LB_0 = g(x^0)$. If $UB_0 - LB_0 \leq \varepsilon$, then the algorithm terminates, and $x^0$ is a global $\varepsilon$-optimal solution of the problem (FP). Otherwise, let $\Omega = \{X^0\}$, $\Lambda = \emptyset$, and $k = 1$.

Step 2. Let $LB_k = LB_{k-1}$. Use the selected branching method to subdivide the rectangle $X^{k-1}$ into two new subrectangles $X^{k,1}$ and $X^{k,2}$, and let $\Lambda = \Lambda \cup \{X^{k-1}\}$.

Step 3. For each subrectangle $X^{k,j}$, where $j = 1$ and 2, solve the problem (LRP) to get $x^{k,j}$ and $UB(X^{k,j})$, and let $LB_k = \max\{LB_j, g(x^{k,j})\}$. If the midpoint $x^{\text{mid}}$ of $X^{k,j}$ ($j \in \{1, 2\}$) is feasible to the problem (FP), let $LB_k = \max\{LB_j, g(x^{\text{mid}})\}$, and let $x^k$ be the best feasible solution satisfying that $LB_k = g(x^k)$.

Step 4. If $LB_k \geq UB(X^{k,j})$, then let $\Lambda = \Lambda \cup \{X^{k,j}\}$, where $j = 1$ and 2, and let $\Omega_k = \{X|X \in \Omega_{k-1} \cup \{X^{k,1}, X^{k,2}\}, X \notin \Lambda\}$.

Step 5. Let $UB_k = \max\{UB(X)|X \in \Omega_k\}$ and $X \in \Omega_k$ satisfy that $UB_k = UB(X)$. If $UB_k - LB_k \leq \varepsilon$, then the algorithm terminates with that $x^k$ is the global $\varepsilon$-optimal solution of the problem (FP). Otherwise, let $k = k + 1$, and return to Step 2.

3.2. Convergence Analysis. In this section, the global convergence of the proposed algorithm is given as follows.

**Theorem 2.** If the proposed algorithm is finite, then the algorithm will terminate after $k$ iteration, and $x^k$ is the global $\varepsilon$-optimal solution of the problem (FP). Otherwise, the proposed algorithm will produce an infinite sequence of iterations such that any accumulation point of the sequence $\{x^k\}$ will be the global optimal solution of the problem (FP).

If the proposed algorithm terminates after $k$ iteration, by the termination condition of the proposed algorithm, we can get that $UB_k - LB_k \leq \varepsilon$. By the updating method of the upper bound, we have that $LB_k = g(x^k)$. By the structure of the branch-and-bound algorithm, we have that $\nu \leq UB_k$ and $LB_k \leq \nu$. By combining these inequalities, we have that $\nu \leq g(x^k) \leq \nu + \varepsilon$. Thus, $x^k$ is a global $\varepsilon$-optimal solution of the problem (FP).

If the present algorithm does not stop after $k$ iterations, by the branch-and-bound structure of the algorithm, it is well known that we can obtain a nondecreasing sequence $\{UB_k\}$ with that the lower bound is $\max_{\sigma \in \Omega} g(x)$, where $\Omega$ is the set of the known feasible point. Thus, we get that $\lim_{k \to \infty} UB_k \geq \max_{\sigma \in \Omega} g(x)$.

Because $\{x^k\} \subseteq X^0$ and $X^0$ is a bounded close set, we can follow that there must exist a convergent subsequence $\{x^k\}$ satisfying that $\lim_{k \to \infty} x^k = x^*$. It is obvious that we have $x^* \in X^0$. By the branch-and-bound framework of the algorithm, there must exist a subsequence $\{X^k\} \subseteq \{X^0\}$, where $X^k \in \Omega_k$ with $x^k \in X^k$, $UB_k = UB(X^k) = Ug(x^k)$, $\lim_{k \to \infty} x^k = x^*$, and $\lim_{k \to \infty} X^k = x^*$. By Theorem 1 and continuity of the function $g(x)$, we can follow that $\lim_{k \to \infty} UB_k = \lim_{k \to \infty} UB(X^k) = \lim_{k \to \infty} Ug(x^k) = \lim_{k \to \infty} g(x^k) = g(x^*)$. From the steps of the algorithm, we know that $x^k$ is always a feasible solution to the problem (FP), thus $x^*$ is the global optimal solution for the problem (FP), and the proof of the theorem is completed.

4. Computational Complexity of the Algorithm

In this section, first of all, we derive the difference between $Ug(x)$ and $g(x)$ with the relationship of $(x - x)$ to analyse the computational complexity of the present algorithm.

For any $x \in X \subseteq X^0$, for each $j = 1, \ldots, p$ and $t = 1, \ldots, m$, without loss of generality, assume that

$$f_{j,t}(x) = \sum_{i=1}^{N} \alpha_{j,t} x_i + d_{j,t},$$

and define

$$\bar{f}_{j,t} = \min_{x \in X} \sum_{i=1}^{N} \alpha_{j,t} x_i + d_{j,t},$$

$$\bar{f}_{j,t} = \max_{x \in X} \sum_{i=1}^{N} \alpha_{j,t} x_i + d_{j,t},$$

Obviously, we have that

$$k_{j,t}^1 \leq \frac{1}{\bar{f}_{j,t}},$$

$$k_{j,t}^2 \leq \frac{1}{\bar{f}_{j,t}}, \quad j = 1, \ldots, p \text{ and } t = 1, \ldots, m,$

and for any $x \in X \subseteq X^0$ and $j = 1, \ldots, p$, we have that
\[ \phi_j^* = \max_{x \in X} \phi_j(x) \geq \max_{x \in X} \phi_j^*(x) = \phi_j^* \tag{37} \]

**Theorem 3.** For any \( x \in X \subseteq X^0 \), the difference between \( U g(x) \) and \( g(x) \) with the relationship of \( (x - \bar{x}) \) satisfies the following inequality:

\[ U g(x) - g(x) \leq \sum_{i=1}^{N} \left\{ \left( \sum_{j=1}^{p} \exp\left( \phi_j^* \right) \left( \frac{1}{\xi_{ij}} |x_{ji}| + \frac{1}{\eta_{ij}} |\delta_{ji}| \right) \right) \right\} (x_i - \bar{x}_i). \tag{38} \]

By Theorem 1, we have that

\[ U g(x) - g(x) = U g(x) - \sum_{j=1}^{p} \exp(\varphi_j(x)) + \sum_{j=1}^{p} \exp(\varphi_j(x)) - g(x). \tag{39} \]

By the proving process of Theorem 1, we have

\[ U g(x) - \sum_{j=1}^{p} \exp(\varphi_j(x)) = \sum_{j=1}^{p} [K_j(\varphi_j(x) - \varphi'_j) + \exp(\varphi'_j)] - \sum_{j=1}^{p} \exp(\varphi_j(x)) \]

\[ = \sum_{j=1}^{p} [K_j(\varphi_j(x) - \varphi'_j) + \exp(\varphi'_j) - \exp(\varphi_j(x))]. \tag{40} \]

From the proving process of Theorem 1, for any \( x \in X \subseteq X^0 \), we have that

\[ \sum_{j=1}^{p} \left\{ K_j(\varphi_j(x) - \varphi'_j) + \exp(\varphi'_j) - \exp(\varphi_j(x)) \right\} \]

\[ \leq \sum_{j=1}^{p} \left\{ K_j(\varphi_j^* - \varphi'_j) + \exp(\varphi'_j) - \exp(\varphi_j(x)) \right\} \]

\[ = \sum_{j=1}^{p} \left\{ \exp(\varphi_j^*) - \exp(\varphi'_j) + \exp(\varphi'_j) - \exp(\varphi_j(x)) \right\} \]

\[ = \sum_{j=1}^{p} \left\{ \exp(\varphi_j^*) - \exp(\varphi_j(x)) \right\} \]

\[ \leq \sum_{j=1}^{p} \left\{ \exp(\varphi_j^*)(\varphi_j^* - \varphi'_j) \right\} \]

\[ \leq \sum_{j=1}^{p} \left\{ \exp(\max_{x \in X^*}(\varphi_j^*)) (\varphi_j^* - \varphi'_j) \right\}. \]
\[
\begin{align*}
&= \sum_{j=1}^{p} \left\{ \exp(\varphi_j^u) \times \left[ \sum_{i=1}^{m} k_j \sum_{i=1}^{N} \alpha_{ji} |(x_i - x_j) + \sum_{i=1}^{m} k_j \sum_{i=1}^{N} \delta_{ji} |(x_i - x_j) \right] \right\} \\
&= \sum_{j=1}^{p} \left\{ \exp(\varphi_j^u) \times \left[ \sum_{i=1}^{N} \left( \sum_{i=1}^{m} k_j \alpha_{ji} + \sum_{i=1}^{m} k_j \delta_{ji} \right) |(x_i - x_j) \right] \right\} \\
&= \sum_{j=1}^{p} \left\{ \exp(\varphi_j^u) \times \left[ \sum_{i=1}^{N} \left( \sum_{i=1}^{m} k_j \alpha_{ji} + \sum_{i=1}^{m} k_j \delta_{ji} \right) |(x_i - x_j) \right] \right\} \\
&\leq \sum_{j=1}^{n} \left\{ \exp\left(\varphi_j^u\right) \times \sum_{i=1}^{N} \left( \frac{1}{\xi_j} |\alpha_{ji}| + \frac{1}{\eta_j} |\delta_{ji}| \right) |(x_i - x_j) \right\} \\
&= \sum_{i=1}^{N} \left\{ \sum_{j=1}^{p} \left( \frac{1}{\xi_j} |\alpha_{ji}| + \frac{1}{\eta_j} |\delta_{ji}| \right) \right\} |(x_i - x_j). \\
\end{align*}
\]

By the proving process of Theorem 1, for any \( x \in X \subseteq X^0 \), we also have

\[
\begin{align*}
&= \sum_{j=1}^{p} \exp(\varphi_j(x)) - g(x) = \sum_{j=1}^{p} \left[ \exp(\varphi_j(x)) - \exp\left( \sum_{i=1}^{m} \ln(f(x_i)) - \ln(h(x_i)) \right) \right] \\
&\leq \sum_{j=1}^{p} \exp(\varphi_j(x)) \left[ \varphi_j(x) - \sum_{i=1}^{m} \ln(f(x_i)) - \ln(h(x_i)) \right] \\
&\leq \sum_{j=1}^{p} \exp\left(\max_{x \in X^0} \varphi_j(x)\right) \left[ \varphi_j(x) - \sum_{i=1}^{m} \ln(f(x_i)) - \ln(h(x_i)) \right] \\
&\leq \sum_{j=1}^{p} \exp\left(\varphi_j^u\right) \times \max_{x \in X^0} \left[ \varphi_j(x) - \sum_{i=1}^{m} \ln(f(x_i)) - \ln(h(x_i)) \right] \\
&\leq \sum_{j=1}^{p} \exp\left(\varphi_j^u\right) \times \left[ \sum_{i=1}^{N} \left( \frac{1}{\xi_j} |\alpha_{ji}| + \frac{1}{\eta_j} |\delta_{ji}| \right) \right] |(x_i - x_j) | \\
&= \sum_{i=1}^{N} \left( \sum_{j=1}^{p} \left( \frac{1}{\xi_j} |\alpha_{ji}| + \frac{1}{\eta_j} |\delta_{ji}| \right) \right) |(x_i - x_j). \\
\end{align*}
\]

Therefore, by the above conclusions, we can get that

\[
Ug(x) - g(x) = Ug(x) - \sum_{j=1}^{p} \exp(\varphi_j(x)) + \sum_{j=1}^{p} \exp(\varphi_j(x)) - g(x) \\
\leq \sum_{i=1}^{N} \left( \sum_{j=1}^{p} \left( \frac{1}{\xi_j} |\alpha_{ji}| + \frac{1}{\eta_j} |\delta_{ji}| \right) \right) |(x_i - x_j). \\
\]
Without losing generality, we define the size $\Theta(X)$ of a rectangle $X = \{ x \in \mathbb{R}^N \mid x_i \leq x \leq x_i, i = 1, 2, \ldots, N \} \subseteq \mathbb{R}^N$ by

$$\Theta(X) = \max \{ \| x_i - x \| \mid i = 1, 2, \ldots, N \}.$$  

Besides, for convenience, we let

$$\beta = \max \left\{ 3 \sum_{j=1}^p \exp(\tilde{g}^n_j) \left( \sum_{t=1}^m \left( \frac{1}{\xi^t} | \alpha_{jt} | + \frac{1}{\eta^t} | \delta_{jt} | \right) \right), i = 1, \ldots, N \right\}.$$  

**Theorem 4.** For any given convergence error $\epsilon > 0$, if there exists a rectangle $X^k$ at the $k_{th}$ iteration, which satisfies $\Theta(X^k) \leq (\epsilon/N\beta)$, then we have that

$$UB(X^k) - LB_k \leq \epsilon,$$  

where $UB(X^k)$ is the optimum value to the problem (LRP) over $X^k$ and $LB_k$ is the known best lower bound of the global minimum value of the problem (FP).

**Proof.** Assume that $x^k$ is the optimal solution to the problem (LRP) over $X^k$; obviously, $x^k$ is also feasible to the problem (FP) over $X^k$. Therefore, we have that

$$g(x^k) \leq UB(X^k) = U g(x^k).$$  

By the conclusion of Theorem 3, we can get that

$$UB(X^k) - LB_k \leq UB(X^k) - g(x^k) \leq N \beta \Theta(X^k).$$  

Furthermore, by $\Theta(X^k) \leq (\epsilon/N\beta)$, we can get that

$$UB(X^k) - LB_k \leq N \beta \Theta(X^k) \leq \epsilon,$$  

and the proof of the theorem is completed. From the steps of the algorithm and the conclusions of Theorem 4, when $\Theta(X^k) \leq (\epsilon/N\beta)$, the investigated rectangle $X^k$ can be deleted from the active nodes’ set. Therefore, when the sizes of all investigated rectangles $X$ satisfy $\Theta(X) \leq (\epsilon/N\beta)$, the present algorithm will be terminated. From the conclusions of Theorem 4, we can estimate the maximum iterations of the present algorithm as follows.

**Theorem 5.** For any given convergence error $\epsilon \in (0, 1)$, the present algorithm can obtain a global $\epsilon$-optimal solution to the problem (FP) in at most

$$N \cdot \left\lceil \log_2 \frac{N \beta \Theta(X^0)}{\epsilon} \right\rceil$$  

iterations.

Without loss of generality, assume that the subrectangle $X^k \subseteq X^0$ is selected for subdividing in Step 2 of the present algorithm for every iteration. After $k \cdot N$ iterations, we can get that

$$\Theta(X^k) \leq \frac{1}{2^k} \Theta(X^0).$$  

From the conclusions of Theorem 4, when

$$\frac{1}{2^k} \Theta(X^0) \leq \frac{\epsilon}{N \beta},$$  

i.e.,

$$k \geq \log_2 \frac{N \beta \Theta(X^0)}{\epsilon},$$  

we can obtain that $UB(X^k) - LB_k \leq \epsilon$. Therefore, after at most

$$N \cdot \left\lceil \log_2 \frac{N \beta \Theta(X^0)}{\epsilon} \right\rceil$$  

iterations, we can obtain a global $\epsilon$-optimal solution to the problem (FP), and the proof of the theorem is completed.
5. Numerical Experiments

To verify the feasibility of the proposed algorithm, the algorithm is coded in C++ software, and the simplex method is used to solve the problem (LRP); some numerical examples are solved on the microcomputer with Intel(R) Core(TM)2 i5-4590s CPU @3.0GHz; the computational comparison of numerical results is listed Tables 1 and 2; although these numerical examples have a relatively small-size variable, they are still challenging. In Table 1, we denote “Iter” as the number of iteration of the present algorithm.

In the following, first of all, some small-size certainty examples in Appendix were tested with the present algorithm; a deep and accurate numerical comparison with the current state-of-the-art “BARON” is given in Table 1. Next, to verify the robustness and reliability of the proposed algorithm, with the convergence error $\varepsilon = 10^{-2}$, we have also solved following randomly generated test Problem 1; numerical comparisons with the current state-of-the-art “BARON” are listed in Table 2.

Problem 1.

\[
\text{(FP): } \quad \max g(x) = \sum_{j=1}^{p} \left( \frac{\sum_{i=1}^{N} c_{ji}^{1} x_{i} + d_{j}^{1}}{\sum_{i=1}^{N} c_{ji}^{1} x_{i} + f_{j}^{1}} \right) \times \left( \frac{\sum_{i=1}^{N} c_{ji}^{2} x_{i} + d_{j}^{2}}{\sum_{i=1}^{N} c_{ji}^{2} x_{i} + f_{j}^{2}} \right)
\]

\[
s.t. \quad Ax \leq b, x \in [0, 1],
\]

where $p$ is a natural number, $A \in \mathbb{R}^{M \times N}$, $b \in \mathbb{R}^{M}$, and $c_{ji}^{1}, c_{ji}^{2}, d_{j}^{1}, f_{j}^{1}, c_{ji}^{2}, e_{ji}^{2}, d_{j}^{2}$, and $f_{j}^{2}$, $j = 1, \ldots, p$ and $i = 1, \ldots, N$, are all randomly generated between 0 and 1; all elements of the matrix $A$ are randomly generated between 0 and 1; all elements of vector $b$ are randomly generated between 0 and 16; the number of the constraints $M = 5$.

When $N \geq 4$, the software BARON failed to solve all arbitrary ten tests in 4800s. Thus, we just report the numerical computational results of our algorithm in Table 2.

Table 1: Numerical comparison among the algorithm BARON and our algorithm on Examples A.1–A.5.

| Example | Refs. | Optimal solution | Optimal value | Iter | CPU time |
|---------|-------|------------------|---------------|------|----------|
| A.1     | This paper | (1.5, 1.5) | 4.912587413 | 17 | 0.079 |
|         | BARON   | (1.5, 1.5) | 4.9126 | | |
| A.2     | This paper | (3.0, 4.0) | 5.0 | 52 | 0.24 |
|         | BARON   | (3.0, 4.0) | 5.0 | 9 | |
| A.3     | This paper | (0.0, 0.0) | 3.0 | 25 | 0.101 |
|         | BARON   | (0.0, 0.0) | 3.0 | 5 | |
| A.4     | This paper | (1.111111111, 0.0, 0.0) | 4.090702948 | 11 | 0.061 |
|         | BARON   | (1.111111111, 0.0000, 0.0000) | 4.0907 | 1 | 0.08 |
| A.5     | This paper | (0, 0) | 0.533333333 | 8 | 0.063 |
|         | BARON   | (0, 0) | 0.5333 | 3 | |

Table 2: Numerical computational comparisons among the algorithm BARON and our algorithm on Problem 1.

| (p, N) | Our algorithm | BARON |
|-------|---------------|-------|
|       | Avg (std).NT | Avg (std).time | Avg (std).NT | Avg (std).time |
| (2, 2) | 24 (15.75506973) | 0.15023405 (0.093567381) | 9.4 (11.06746985) | 0.228 (0.132815662) |
| (2, 3) | 75.4 (43.58440088) | 0.7650699 (0.449198958) | 300 (824.0233276) | 0.887 (1.666667) |
| (2, 4) | 154 (80.77955356) | 2.299828 (1.027666995) | — | — |
| (2, 5) | 422.8 (260.6222469) | 8.358369 (5.156726039) | — | — |
| (2, 6) | 505.5 (528.8942869) | 9.886233 (10.21000994) | — | — |
| (2, 7) | 3243.3 (1540.659603) | 107.42306 (50.68952342) | — | — |
| (2, 8) | 9178.5 (7330.518357) | 389.16934 (315.0401155) | — | — |
| (2, 9) | 11195.2 (7294.911283) | 575.2941 (382.8851755) | — | — |

In the following, first of all, some small-size certainty examples in Appendix were tested with the present algorithm; a deep and accurate numerical comparison with the current state-of-the-art “BARON” is given in Table 1. Next, to verify the robustness and reliability of the proposed algorithm, with the convergence error $\varepsilon = 10^{-2}$, we have also solved following randomly generated test Problem 1; numerical comparisons with the current state-of-the-art “BARON” are listed in Table 2.

In Table 2, some notations have been used as follows: Avg.NT denotes the average number of iterations for the algorithm; Std.NT denotes the standard deviation of number of iterations for the algorithm; Avg.Time denotes the average running time of the algorithm in seconds; Std.Time denotes the standard deviation of running time for the algorithm.

From Table 1, it is observed that, our algorithm spends less running time than the software BARON. Thus, our algorithm is the most efficient one for solving the problem (FP) with the small-size variable.
From Table 2, we can observe that, when $N \geq 4$, the software BARON failed to solve all arbitrary ten tests in 4800 s, but our algorithm can obtain the global optimal solution of the problem (FP) for arbitrary ten tests in a short time. So this demonstrates that our algorithm has the stronger robustness and stability than the software BARON.

6. Conclusions

In this paper, based on the branch-and-bound framework, we present an efficient global optimization algorithm for solving the problem (FP). In this algorithm, a novel linearizing technique is proposed for deriving the linear relaxation problem of the problem (FP), which can provide a reliable upper bound in the branch-and-bound algorithm. By subsequently partitioning the initial rectangle and solving a series of linear relaxation problems, the proposed algorithm is convergent to the global optimal solution for the problem (FP). Furthermore, based on the steps of the branch-and-bound algorithm, the computational complexity of the algorithm is analysed. Finally, numerical experiments verify the feasibilities and effectiveness of the proposed algorithm. The future work is to extend our new algorithm to the min-max affine fractional programming problem and generalized linear fractional programming problems.

Example 1 (see Ref. [9]).

$$\begin{align*}
\min & \quad \frac{37x_1 + 73x_2 + 13}{\sum_{i=1}^{n} a_i x_i + 13} + \frac{63x_1 - 18x_2 + 39}{\sum_{i=1}^{n} a_i x_i + 13} \\
\text{s.t.} & \quad 5x_1 - 3x_2 = 3, \\
& \quad 1.5 \leq x_1 \leq 3.
\end{align*}$$

Example 2 (see Ref. [43]).

$$\begin{align*}
\max & \quad \frac{37x_1 + 73x_2 + 13}{\sum_{i=1}^{n} a_i x_i + 13} + \frac{63x_1 - 18x_2 + 39}{\sum_{i=1}^{n} a_i x_i + 13} \\
\text{s.t.} & \quad 5x_1 - 3x_2 = 3, \\
& \quad 1.5 \leq x_1 \leq 3.
\end{align*}$$

Example 3 (see Ref. [43]).

$$\begin{align*}
\max & \quad \frac{3x_1 + 5x_2 + 3x_3 + 50}{\sum_{i=1}^{n} a_i x_i + 13} + \frac{3x_1 + 5x_2 + 50}{\sum_{i=1}^{n} a_i x_i + 13} + \frac{4x_1 + 2x_2 + 4x_3 + 50}{\sum_{i=1}^{n} a_i x_i + 13} \\
\text{s.t.} & \quad 6x_1 + 3x_2 + 3x_3 \leq 10, \\
& \quad 10x_1 + 3x_2 + 8x_3 \leq 10, \\
& \quad x_1, x_2, x_3 \geq 0.
\end{align*}$$

Example 4 (see Ref. [24]).

$$\begin{align*}
\max & \quad \frac{4x_1 + 3x_2 + 3x_3 + 50}{\sum_{i=1}^{n} a_i x_i + 13} + \frac{3x_1 + 4x_2 + 50}{\sum_{i=1}^{n} a_i x_i + 13} + \frac{x_1 + 2x_2 + 5x_3 + 50}{\sum_{i=1}^{n} a_i x_i + 13} + \frac{x_1 + 2x_2 + 4x_3 + 50}{\sum_{i=1}^{n} a_i x_i + 13} \\
\text{s.t.} & \quad 2x_1 + x_2 + 5x_3 \leq 10, \\
& \quad x_1 + 6x_2 + 3x_3 \leq 10, \\
& \quad 5x_1 + 9x_2 + 2x_3 \leq 10, \\
& \quad 9x_1 + 7x_2 + 3x_3 \leq 10, \\
& \quad x_1, x_2, x_3 \geq 0.
\end{align*}$$

Appendix

Examples of some small-size certainty are given in the following.

$$\begin{align*}
\max & \quad \frac{3x_1 + 5x_2 + 3x_3 + 50}{\sum_{i=1}^{n} a_i x_i + 13} + \frac{3x_1 + 5x_2 + 50}{\sum_{i=1}^{n} a_i x_i + 13} + \frac{4x_1 + 2x_2 + 4x_3 + 50}{\sum_{i=1}^{n} a_i x_i + 13} \\
\text{s.t.} & \quad 6x_1 + 3x_2 + 3x_3 \leq 10, \\
& \quad 10x_1 + 3x_2 + 8x_3 \leq 10, \\
& \quad x_1, x_2, x_3 \geq 0.
\end{align*}$$
Example 5 (see Ref. [24]).

\[
\begin{align*}
\min & \quad -x_1 + 2x_2 + 2 - 4x_1 - 3x_2 + 4 \\
& \quad 3x_1 - 4x_2 + 5 \quad 2x_1 + x_2 + 3 \\
\text{s.t.} & \quad x_1 + x_2 \leq 1.5, \\
& \quad x_1 \leq x_2, \\
& \quad 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1.
\end{align*}
\](A.5)

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (11871196 and 61304061) and Key Scientific and Technological Research Projects of Henan Province (202102210147 and 192102210114).

References

[1] I. M. Stancu-Minasian, *Fractional Programming: Theory, Methods and Applications*, Kluwer, Dordrecht, Netherlands, 1997.

[2] H. Konno, Y. Yajima, and T. Matsui, "Parametric simplex algorithms for solving a special class of nonconvex minimization problems," *Journal of Global Optimization*, vol. 1, no. 1, pp. 65–81, 1991.

[3] A. Cambini, L. Martein, and S. Schaible, "On maximizing a sum of ratios," *Journal of Information and Optimization Sciences*, vol. 10, no. 1, pp. 65–79, 1989.

[4] J. E. Falk and S. W. Palocsay, "Image space analysis of generalized fractional programs," *Journal of Global Optimization*, vol. 4, no. 1, pp. 63–88, 1994.

[5] J. Quesada and I. Grossmann, "A global optimization algorithm for linear fractional and bilinear programs," *Journal of Global Optimization*, vol. 6, no. 1, pp. 39–76, 1995.

[6] H. Konno and K. Fukaishi, "A branch-and-bound algorithm for solving low-rank linear multiplicative and fractional programming problems," *Journal of Global Optimization*, vol. 18, no. 3, pp. 283–299, 2000.

[7] C. F. Wang and P. P. Shen, "A global optimization algorithm for linear fractional programming," *Applied Mathematics and Computation*, vol. 204, no. 1, pp. 281–287, 2008.

[8] H. Jiao and S. Liu, "A practicable branch and bound algorithm for sum of linear ratios problem," *European Journal of Operational Research*, vol. 243, no. 3, pp. 723–730, 2015.

[9] Y. Gao and S. Jin, "A global optimization algorithm for sum of linear ratios problem," *Journal of Applied Mathematics*, vol. 2013, Article ID 276245, 7 pages, 2013.

[10] Y. Ji, K. Zhang, and S. Qu, "A deterministic global optimization algorithm," *Applied Mathematics and Computation*, vol. 185, no. 1, pp. 382–387, 2007.

[11] P. Shen and C. Wang, "Global optimization for sum of linear ratios problem with coefficients," *Applied Mathematics and Computation*, vol. 176, no. 1, pp. 219–229, 2006.

[12] H. Jiao, S. Liu, J. Yin, and Y. Zhao, "Outcome space range reduction method for global optimization of sum of affine ratios problem," *Open Mathematics*, vol. 14, no. 1, pp. 736–746, 2016.

[13] T. Kuno, "A branch-and-bound algorithm for maximizing the sum of several linear ratios," *Journal of Global Optimization*, vol. 22, pp. 155–174, 2002.

[14] T. Kuno, "A revision of the trapezoidal branch-and-bound algorithm for linear sum-of-ratios problems," *Journal of Global Optimization*, vol. 33, no. 2, pp. 215–234, 2005.

[15] T. H. P. Nguyen and H. Tuy, "A unified monotonic approach to generalized linear fractional programming," *Journal of Global Optimization*, vol. 26, pp. 229–259, 2003.

[16] C. F. Wang and S. Y. Liu, "A new linearization method for generalized linear multiplicative programming," *Computers & Operations Research*, vol. 38, no. 7, pp. 1008–1103, 2011.

[17] P. Shen and H. Jiao, "Linearization method for a class of multiplicative programming with exponent," *Applied Mathematics and Computation*, vol. 183, no. 1, pp. 328–336, 2006.

[18] H. S. Ryoo and N. V. Sahinidis, "Global optimization of multiplicative programs," *Journal of Global Optimization*, vol. 26, no. 4, pp. 387–418, 2003.

[19] P. Shen and L. Wang, "A Fully polynomial time approximation algorithm for generalized linear multiplicative programming," *Mathematica Applicata*, vol. 31, pp. 208–213, 2018.

[20] Y. L. Gao, C. X. Xu, and Y. L. Yang, "An outcome-space finite algorithm for solving linear multiplicative programming," *Applied Mathematics and Computation*, vol. 179, no. 2, pp. 494–505, 2006.

[21] B. Zhang, Y. Gao, X. Liu, and X. Huang, "Output-space branch-and-bound reduction algorithm for a class of linear multiplicative programs," *Mathematics*, vol. 8, no. 3, p. 315, 2020.

[22] E. A. Youness, "Level set algorithm for solving convex multiplicative programming problems," *Applied Mathematics and Computation*, vol. 167, no. 2, pp. 1412–1417, 2005.

[23] X. J. Liu, "Heuristic methods for linear multiplicative programming," *Journal of Global Optimization*, vol. 4, no. 15, pp. 433–447, 1999.

[24] H. Jiao, "A branch and bound algorithm for globally solving a class of nonconvex programming problems," *Nonlinear Analysis: Theory, Methods & Applications*, vol. 70, no. 2, pp. 1113–1123, 2009.

[25] H. Jiao, S. Liu, and Y. Zhao, "Effective algorithm for solving the generalized linear multiplicative problem with generalized polynomial constraints," *Applied Mathematical Modelling*, vol. 39, no. 23-24, pp. 7568–7582, 2015.

[26] Y. Q. Chen and H. W. Jiao, "A nonisolated optimal solution of general linear multiplicative programming problems," *Computers & Operations Research*, vol. 36, no. 9, pp. 2573–2579, 2009.

[27] X. L. Huang, Y. L. Gao, B. Zhang, and X. Liu, "An effective computational algorithm for the global solution of a class of linear fractional programming," *Mathematical Problems in Engineering*, vol. 2020, Article ID 5380419, 14 pages, 2020.

[28] Z. Wang and Y. Zhang, "A deterministic method for solving the sum of linear ratios problem," *Mathematical Problems in Engineering*, vol. 2020, Article ID 6174352, 8 pages, 2020.

[29] H. Jiao, W. Wang, R. Chen, Y. Shang, and J. Yin, "An efficient outer space algorithm for generalized linear multiplicative
programming problem,” *IEEE Access*, vol. 8, pp. 80629–80637, 2020.

[30] J. Yin, H. Jiao, and Y. Shang, “Global algorithm for generalized affine multiplicative programming problem,” *IEEE Access*, vol. 7, pp. 162245–162253, 2019.

[31] H. Jiao and R. Chen, “A parametric linearizing approach for quadratically inequality constrained quadratic programs,” *Open Mathematics*, vol. 16, no. 1, pp. 407–419, 2018.

[32] H. Jiao and S. Liu, “An efficient algorithm for quadratic sum-of-ratios fractional programs problem,” *Numerical Functional Analysis and Optimization*, vol. 38, no. 11, pp. 1426–1445, 2017.

[33] H. Jiao and S. Liu, “Range division and compression algorithm for quadratically constrained sum of quadratic ratios,” *Computational and Applied Mathematics*, vol. 36, no. 1, pp. 225–247, 2017.

[34] H. Jiao, Y. Shang, and W. Wang, “Solving generalized polynomial problem by using new affine relaxed technique,” *International Journal of Computer Mathematics*, 2021.

[35] H. Jiao, Y. Guo, and P. Shen, “Global optimization of generalized linear fractional programming with nonlinear constraints,” *Applied Mathematics and Computation*, vol. 183, no. 2, pp. 717–728, 2006.

[36] A. Ghazi and A. Roubi, “Optimality conditions and DC-dinkelbach-type algorithm for generalized fractional programs with ratios of difference of convex functions,” *Optimization Letters*, 2021.

[37] A. Bennani, D. Benterki, and H. Grar, “Optimality conditions and DC-dinkelbach-type algorithm for generalized fractional programs with ratios of difference of convex functions,” *RAIRO Operations Research*, vol. 55, pp. S2383–S2392, 2021.

[38] Y. Song, D. Wu, W. Deng, and X. Z. Gao, “MPPCEDE: multi-population parallel co-evolutionary differential evolution for parameter optimization,” *Energy Conversion and Management*, vol. 228, Article ID 113661, 2021.

[39] W. Deng, J. Xu, Y. Song, and H. Zhao, “Differential evolution algorithm with wavelet basis function and optimal mutation strategy for complex optimization problem,” *Applied Soft Computing*, vol. 100, Article ID 106724, 2021.

[40] X. Cai, H. Zhao, S. Shang et al., “An improved quantum-inspired cooperative co-evolution algorithm with multi-strategy and its application,” *Expert Systems with Applications*, vol. 171, Article ID 114629, 2021.

[41] W. Deng, J. Xu, H. Zhao, and Y. Song, “A novel gate resource allocation method using improved PSO-based QEA,” *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[42] W. Deng, J. Xu, X. Z. Gao, and H. Zhao, “An enhanced MSIQDE algorithm with novel multiple strategies for global optimization problems,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[43] Y. Pei and D. Zhu, “Global optimization method for maximizing the sum of difference of convex functions ratios over nonconvex region,” *Journal of Applied Mathematics and Computing*, vol. 41, no. 1-2, pp. 153–169, 2013.