AIMS Mathematics, 10(9): 20843–20861.
DOI:10.3934/math.2025931
Received: 25 June 2025
Revised: 13 August 2025
Accepted: 26 August 2025
Published: 10 Sepetember 2025

http://www.aimspress.com/journal/Math

Research article

Global algorithm for addressing sum of linear ratios problem using the separable nature of relaxation problem

Qunzhen Zheng[1], **Chenglin He**[2], **Yan Shi**[3,*] **and Jingben Yin**[2,*]

[1] School of Statistics and Mathematics, Henan Finance University, Zhengzhou 450046, China
[2] School of Mathematical Sciences, Henan Institute of Science and Technology, Xinxiang 453003, China
[3] College of Information Engineering, Henan University of Animal Husbandry and Economy, Zhengzhou 450000, China

* **Correspondence:** Email: wqmshiyan@163.com, jingbenyin@163.com.

Abstract: This paper proposed an algorithm based on the branch-and-bound framework for globally solving the sum of linear ratios problem (SLRP) with a large number of ratios and a small number of variables. First, we introduced new variables to construct an equivalent problem of the problem (SLRP). Then, using a new linear relaxation technique, we obtained the linear relaxation problem for the equivalent problem. By utilizing the separable nature of the linear relaxation problem, we computed the linear relaxation problem by solving its p linear programming subproblems, thereby the lower bound for the problem (SLRP) could be obtained. Additionally, we conducted a theoretical analysis of the proposed algorithm and validated its feasibility and effectiveness through numerical experiments.

Keywords: sum of linear ratios problem; global optimization; linear relaxation technique; separable nature; computational complexity
Mathematics Subject Classification: 65K05, 90C26, 90C32

1. Introduction

Fractional programming is an important class of nonlinear optimization that has attracted significant interest from researchers over the past few decades. The sum of linear ratios problem (SLRP) is a type of fractional programming problem that has many real-world applications, such as investment and financial planning [9, 25], government contracting problems [7], transportation and logistics [2], economic benefits and production control [31], multi-objective optimization [3, 8], and so on [33, 34]. Specifically, a real-life example and application are given as follows:

Minimum area convex hull problem: Because the minimum area convex hull problem considered is in two-dimensional space, the dimension of the variable of the problem considered is 2. The minimum area convex hull problem can be transformed into a minimum convex polygon area problem containing all points. The minimum convex polygon area problem including all points can be transformed into a sum of linear ratios problem with 2 variables. When a convex polygon contains a large number of vertices, there are a large number of linear ratios in the minimum convex polygon area problem, that is, the number of linear ratios is much greater than the dimension 2 of the variable. In summary, the mathematical modeling of the minimum area convex hull problem in two-dimensional space can be written as follows:

$$\min \sum_{t=1}^{p} \frac{\bar{f}_t(x)}{\bar{g}_t(x)}, \text{ s.t. } x \in X,$$

where $\bar{f}_t(x)$ and $\bar{g}_t(x)$ are linear functions, $x \in \mathbb{R}^2$, X is the feasible region, $p \in N+$, and $p \gg 2$. Regarding this model, please refer to [1].

Furthermore, due to the non-convex objective function of this problem, there may be multiple local optima that are not global optima, which brings enormous theoretical and computational complexity to solving this problem. Therefore, the sum of linear ratios problem is NP hard [4, 28], and solving such a problem exists important practical applications and computational difficulties.

In this paper, we will investigate the following sum of linear ratios problem:

$$\text{(SLRP):} \begin{cases} \min \ f(x) = \sum_{i=1}^{p} \frac{d_i^\top x + c_i}{e_i^\top x + g_i}, \\ \text{s.t. } x \in \chi = \{x \in R^n \mid Ax \leq b, x > 0\}, \end{cases}$$

where $A \in R^{m\times n}$, $b \in R^m$; d_i and $e_i \in R^n$, c_i and $g_i \in R, i = 1, \ldots, p$; χ is a nonempty bounded set; and for any $i = 1, \ldots, p, c_i^\top x + f_i$ and $e_i^\top x + g_i$ are affine functions over χ; $e_i^\top x + g_i > 0$, for all $x \in \chi$.

Over the past few decades, many researchers have proposed various algorithms for the sum of ratios problem with a small number of ratios and large-size variables. For example, by utilizing the convex and concave hull approximation techniques of univariate fractional functions, Shen and Wang [32] proposed an outer space branch-and-bound algorithm for globally solving the sum of linear ratios problem with coefficients. Borza and Rambely [5] presented a non-iterative and straightforward method with less computational expenses to deal with the sum of linear fractional functions over a polyhedral set. Jiao and Liu [13] designed a global optimization algorithm for solving the sum of linear ratios problem by combining the bilinear relaxation technique with the outer space branch-reduction-bound search. Shen et al. [30] designed a spatial branch-and-bound algorithm by using the resulting second-order cone relaxation bounding technique. The literatures [16–18] presented several spatial algorithms for the sum of ratios problem with a small number of ratios and large-size variables. In addition, based on recent monotonic optimization theory, Phuong and Tuy [29] proposed a unified monotonic approach for the generalized linear fractional programming problem. Based on the variable dimensional space partitioning search and the different two-stage relaxation techniques, Jiao et al. [19, 20] proposed two global algorithms for the generalized linear fractional programming problem and generalized polynomial problem, respectively. Specifically, the literatures [14, 15] provided, for the first time, two outer space branch-relaxation-bound algorithms for generalized linear fractional programming problems and generalized affine multiple product programming problems.

Recently, Li et al. [26] proposed an outer space branch-and-bound algorithm for linear ratio problems based on the inverse denominator outer space partitioning search and the direct relaxation bounding technique. Based on the entire fractional image space partitioning search and direct relaxation techniques, Li et al. [27] proposed an image space branch-and-bound algorithm for affine comparison equations. Based on fractional image space partitioning search and two-stage relaxation technique, Hou and Liu [10] provided an image space branch-reduction-bound algorithm for generalized fractional programming problems. Based on new spatial branching and relaxation bounding techniques, Hou and Liu [11] provided a spatial branching-pruning-bounding algorithm for generalized linear fractional problems; Hou and Liu [12] provided an accelerated outer space algorithm for generalized linear multiple-product programming problems based on linear function like spatial branch search and region acceleration methods. Based on standard space search and new relaxation techniques, Jiao et al. [21] proposed a standard space acceleration algorithm for generalized multiple-product programming problems. For the generalized affine fractional programming problem, Jiao and Ma [22] proposed an efficient outer space algorithm based on different relaxation processes or techniques. In addition, using the Charns-Cooper transform, a reduced outer space equivalence problem was constructed, and a direct relaxation technique was used to construct the relaxation problem. Jiao et al. [23] proposed a reduced outer space algorithm for sum of linear fractional functions problems. In addition, the improved metaheuristic approach [37], the bioinspired discrete two-stage surrogate-assisted algorithm [35], and the migration-inspired meta-heuristic algorithm [36] in the recent literatures also provided some new ideas for solving the problem (SLRP) with a small number of ratios and small-size variables.

Although the algorithms (such as [19,20,29]) mentioned above can also be used to solve the problem (SLRP) with a small number of ratios and small-size variables, to the best of our knowledge, the global algorithms for efficiently solving the problem (SLRP) with a large number of ratios and a small number of variables are still scarce.

In this paper, we propose a global optimization algorithm based on the branch-and-bound framework to solve the SLRP with a large number of ratios and a small number of variables. First, we transform the original problem (SLRP) into an equivalent problem using the Charnes-Cooper transformation. Then, by using the new relaxation method, we obtain a decomposable linear relaxation problem of the equivalent problem. Utilizing the decomposability of the linear relaxation problem, we can solve the linear relaxation problem by computing its p linear programming subproblems for obtaining the lower bound in the branch-and-bound process, and propose a branch-and-bound algorithm. Finally, numerical experiments demonstrate that the proposed algorithm is feasible and efficient for solving the problem (SLRP) with a large number of ratios and a small number of variables.

The structure of this paper is as follows: In Section 2, we present the equivalent problem (EP) of the problem (SLRP) and provide a proof of equivalence. Subsequently, we construct the linear relaxation problem (LRP) for the EP using a new relaxation technique and decompose it into a series of subproblems. In Section 3, we propose a global relaxation method to solve the subproblems derived above and provide a convergence analysis and a computational complexity of the algorithm. Section 4 demonstrates the feasibility and efficiency of the algorithm through numerical experiments, and Section 5 presents the relevant conclusions.

2. Equivalent problem and its linear relaxation

In this section, to globally solve the problem (SLRP), by using Charnes-Cooper transformation [6], we first introduce new variables $y_i = \dfrac{1}{e_i^\top x + g_i}$, $s_i = y_i x, i = 1, \ldots, p$, and transform the problem (SLRP) into the following equivalent problem:

$$(\text{EP}): \begin{cases} \min \phi(y, s) = \sum_{i=1}^{p} (d_i^\top s_i + c_i y_i), \\ \text{s.t. } e_i^\top s_i + g_i^\top y_i = 1, \ i = 1, \ldots, p, \\ \quad s_i = y_i x, \ i = 1, \ldots, p, \\ \quad y_i > 0, \ i = 1, \ldots, p, \\ \quad Ax \le b, x > 0, s > 0. \end{cases}$$

Definition 2.1. *If there exists a point $x^* \in \chi$ such that $f(x^*) \le f(x)$ holds for any $x \in \chi$, then we call x^* an optimal solution of the problem (SLRP).*

Theorem 2.1. *x^* is the optimal solution of the problem (SLRP) if and only if (x^*, y^*, s^*) is the optimal solution of the problem (EP), where $y^* = (y_1^*, y_2^*, \ldots, y_p^*)^\top$ and $s^* = (s_1^*, s_2^*, \ldots, s_p^*)^\top$ with $y_i^* = \frac{1}{e_i^\top x^* + g_i}$ and $s_i^* = y_i^* x^*$ for each $i = 1, \ldots, p$.*

Proof. Let (x^*, y^*, s^*) be an optimal solution of the problem (EP), where $s_i^* = y_i^* x^*, y_i^* = \dfrac{1}{e_i^\top x^* + g_i}$, $i = 1, \ldots, p$. Next, we will prove that x^* is a global optimal solution to the problem (SLRP). Now, suppose x^* is not the global optimal solution for the problem (SLRP), then there exists a feasible solution $\overline{x}$ for the problem (SLRP) such that $f(\overline{x}) < f(x^*)$. Let

$$\overline{y}_i = \frac{1}{e_i^\top \overline{x} + g_i}, \quad \overline{s}_i = \overline{y}_i \overline{x}, \quad i = 1, \ldots, p, \tag{2.1}$$

and it is easy to see that $(\overline{x}, \overline{y}, \overline{s})$ is a feasible solution of the problem (EP), where $\overline{y} = (\overline{y}_1, \overline{y}_2, \ldots, \overline{y}_p)^\top$ and $\overline{s} = (\overline{s}_1, \overline{s}_2, \ldots, \overline{s}_p)^\top$. Through $f(\overline{x}) < f(x^*)$ and (2.1), we can obtain

$$\sum_{i=1}^{p} (d_i^\top \overline{s}_i + c_i \overline{y}_i) = \sum_{i=1}^{p} \frac{d_i^\top \overline{x} + c_i}{e_i^\top \overline{x} + g_i} < \sum_{i=1}^{p} \frac{d_i^\top x^* + c_i}{e_i^\top x^* + g_i}. \tag{2.2}$$

Additionally, since (x^*, y^*, s^*) is also a feasible point for the problem (EP) with

$$y_i^* = \frac{1}{e_i^\top x^* + g_i}, \quad s_i^* = y_i^* x^*, \quad i = 1, \ldots, p,$$

we have

$$\sum_{i=1}^{p} \frac{d_i^\top x^* + c_i}{e_i^\top x^* + g_i} = \sum_{i=1}^{p} (d_i^\top s_i^* + c_i y_i^*). \tag{2.3}$$

Combining (2.2) with (2.3), we can obtain

$$\sum_{i=1}^{p} (d_i^\top \overline{s}_i + c_i \overline{y}_i) < \sum_{i=1}^{p} (d_i^\top s_i^* + c_i y_i^*).$$

Since $(\overline{x}, \overline{y}, \overline{s})$ is a feasible solution of the problem (EP), this contradicts the optimality of (x^*, y^*, s^*) for the problem (EP). Therefore, the supposition that x^* is not a global optimal solution for the problem (SLRP) must be false.

Next, we show the converse case. Assuming that x^* is a global optimal solution for the problem (SLRP), and letting

$$s_i^* = y_i^* x^*, \quad y_i^* = \frac{1}{e_i^\top x^* + g_i}, \quad i = 1, \ldots, p, \tag{2.4}$$

then (x^*, y^*, s^*) is a feasible solution of the problem (EP). Now, suppose the problem (EP) exists a feasible solution $(\overline{x}, \overline{y}, \overline{s})$, such that

$$\sum_{i=1}^{p}(d_i^\top \overline{s}_i + c_i \overline{y}_i) < \sum_{i=1}^{p}(d_i^\top s_i^* + c_i y_i^*). \tag{2.5}$$

Since $(\overline{x}, \overline{y}, \overline{s})$ is a feasible solution of the problem (EP), we can see that (2.1) must be hold.

Hence, combining (2.1), (2.4) with (2.5), it holds that

$$f(\overline{x}) = \sum_{i=1}^{p} \frac{d_i^\top \overline{x} + c_i}{e_i^\top \overline{x} + g_i} = \sum_{i=1}^{p}(d_i^\top \overline{s}_i + c_i \overline{y}_i) < \sum_{i=1}^{p}(d_i^\top s_i^* + c_i y_i^*) = \sum_{i=1}^{p} \frac{d_i^\top x^* + c_i}{e_i^\top x^* + g_i} = f(x^*).$$

Since $\overline{x} \in \chi$, this contradicts that x^* is a global optimal solution of the problem (SLRP). Hence, the supposition that the problem (EP) exists a feasible solution $(\overline{x}, \overline{y}, \overline{s})$ satisfying (2.5) must be false, and the proof is completed. □

Next, multiplying both ends of inequality $Ax \le b$ by y_i simultaneously yields the new equivalent form of problem (EP) as follows:

$$(\text{EP1}): \begin{cases} \min \phi(y, s) = \sum_{i=1}^{p}(d_i^\top s_i + c_i y_i), \\ \text{s.t. } e_i^\top s_i + g_i^\top y_i = 1, \ i = 1, \ldots, p, \\ \quad s_i = y_i x, \ i = 1, \ldots, p, \\ \quad y_i > 0, \ i = 1, \ldots, p, \\ \quad A s_i - b y_i \le 0, \ i = 1, \ldots, p, \\ \quad x > 0, s > 0. \end{cases}$$

For the problem (SLRP), let $l_j^0 = \min\{x_j \mid x \in \chi\} > 0$ and $u_j^0 = \max\{x_j \mid x \in \chi\}$, and we can construct an initial rectangle H^0 as follows:

$$H^0 = [l^0, u^0] \triangleq \{x \in R^n \mid l_j^0 \le x_j \le u_j^0, \quad j = 1, \ldots, n\}.$$

Then, the equivalent problem (EP1) on rectangle H^0 is formulated as follows:

$$(\mathrm{EP1}(H^0)):\begin{cases}\min \phi(y,s)=\sum_{i=1}^{p}(d_i^{\top}s_i+c_iy_i),\\ \text{s.t. } e_i^{\top}s_i+g_i^{\top}y_i=1,\ i=1,\ldots,p,\\ \quad s_i=y_ix,\ i=1,\ldots,p,\\ \quad y_i>0,\ i=1,\ldots,p,\\ \quad As_i-by_i\le 0,\ i=1,\ldots,p,\\ \quad x\in H^0,s>0.\end{cases}$$

In this paper, we define

$$H=[l,u]\triangleq\{x\in R^n\mid l_j\le x_j\le u_j,\quad j=1,\cdots,n\}$$

as the initial rectangle H^0 or its sub-rectangle resulting from the branching process. Then, the problem (EP1) about H is given as follows:

$$(\mathrm{EP1}(H)):\begin{cases}\min \phi(y,s)=\sum_{i=1}^{p}(d_i^{\top}s_i+c_iy_i),\\ \text{s.t. } e_i^{\top}s_i+g_i^{\top}y_i=1,\ i=1,\ldots,p,\\ \quad s_i=y_ix,\ i=1,\ldots,p,\\ \quad y_i>0,\ i=1,\ldots,p,\\ \quad As_i-by_i\le 0,\ i=1,\ldots,p,\\ \quad x\in H,s>0.\end{cases}$$

After the above discussion, we can see that the feasible set of the problem (EP1(H)) is neither a polyhedron nor a convex set. By directly relaxing the constrained functions $s_i=y_ix, i=1,\ldots,p$, we construct the linear relaxation problem of the problem (EP1(H)) as follows:

$$(\mathrm{LRP}(H)):\begin{cases}\min \phi(y,s)=\sum_{i=1}^{p}(d_i^{\top}s_i+c_iy_i),\\ \text{s.t. } e_i^{\top}s_i+g_iy_i=1,\ i=1,\ldots,p,\\ \quad y_il\le s_i\le y_iu,\ i=1,\ldots,p,\\ \quad As_i-by_i\le 0,\ i=1,\ldots,p,\\ \quad y_i>0,\ i=1,\ldots,p,\\ \quad s>0.\end{cases}$$

Based on the characteristics of the problem (LRP(H)), it can be decomposed into p linear programming subproblems:

$$(\mathrm{P}_i(H)):\begin{cases}\phi_i=\min d_i^{\top}s_i+c_iy_i,\\ \text{s.t. } e_i^{\top}s_i+g_iy_i=1,\\ \quad y_il\le s_i\le y_iu,\\ \quad As_i-by_i\le 0,\\ \quad y_i>0,s_i>0,\end{cases}$$

where $i = 1, \ldots, p$. By solving these linear programming subproblems ($P_i(H)$), $i = 1, \ldots, p$, we can solve the problem (LRP(H)).

Theorem 2.2. *The linear relaxed problem (LRP(H)) has an optimal solution (y^*, s^*) if and only if the subproblem ($P_i(H)$) has an optimal solution (y_i^*, s_i^*) for each $i = 1, \ldots, p$.*

Proof. According to the characteristic structure of the problems (LRP(H)) and ($P_i(H)$), this conclusion is easily obtained, so the proof is omitted. □

Therefore, form Theorem 2.2, by solving the p linear programming subproblems ($P_i(H)$), we can compute the lower bound LB(H) of the problem (SLRP(H)), where $LB(H) = \sum_{i=1}^{p} \phi_i$.

Remark 2.1. *The special advantages and benefits of the relaxation method proposed in this article are given as follows. The $(pn + p)$-dimensional linear relaxation problem constructed using the relaxation method proposed in this article has the characteristic of variable separability. Solving the $(pn + p)$-dimensional linear relaxation problem can be decomposed into solving p linear programming subproblems with $(n + 1)$-dimensional variables, thereby reducing the difficulty of solving the problem and enabling the algorithm to handle the sum of the linear ratio problem with a large number of ratios and a small number of variables. Specifically, each of these lower-dimensional problems can be solved independently by optimizing over a subset of the original variables, which allows for efficient parallel computation and further accelerates the convergence of the overall algorithm.*

3. Algorithm and its theoretical analysis

In this section, we propose a branch-and-bound algorithm based on maximum edge bisecting for globally solving the problem (SLRP). We outline the basic workflow of the proposed algorithm, demonstrate its convergence, and estimate the maximum number of iterations of the algorithm.

3.1. Branching rule

In this subsection, we will introduce the branching rule used in the algorithm. Assume that the sub-rectangle $H^k = [l^k, u^k] \triangleq \{x \in R^n \mid l_j^k \le x_j \le u_j^k, j = 1, \ldots, n\} \subseteq H^0$ is selected, which will be subdivided in the iteration process. Let

$$\hat{j} = \arg\max\{u_j^k - l_j^k \mid j = 1, \ldots, n\},$$

and

$$\Theta_{\hat{j}}^k = \frac{l_{\hat{j}}^k + u_{\hat{j}}^k}{2},$$

and by bisecting along the maximum side of rectangle H^k, we can partition the rectangle H^k into the following two sub-rectangles:

$$H^{k,1} = \{x \in R^n \mid l_j^k \le x_j^k \le u_j^k, j = 1, \ldots, n, j \ne \hat{j}; l_{\hat{j}}^k \le x_{\hat{j}} \le \Theta_{\hat{j}}^k\},$$

and

$$H^{k,2} = \{x \in R^n \mid l_j^k \le x_j^k \le u_j^k, j = 1, \ldots, n, j \ne \hat{j}; \Theta_{\hat{j}}^k \le x_{\hat{j}} \le u_{\hat{j}}^k\}.$$

3.2. Branch-and-bound algorithm

In this subsection, we briefly outline the basic steps of the algorithm as follows.

Definition 3.1. *Let v be the global optimum value for the problem (SLRP), and let $\tilde{x}$ be a feasible solution for the MLRP. For any given termination error $\epsilon > 0$, if $f(\tilde{x}) - v \leq \epsilon$, we call $\tilde{x}$ an ϵ-global optimum solution for the problem (SLRP).*

Algorithm steps:

Step 0: Given the convergence error $\epsilon > 0$ and calculating the initial rectangle

$$H^0 = [l^0, u^0] \triangleq \{x \in \mathbb{R}^n \mid l_j^0 \leq x_j \leq u_j^0, \quad j = 1, \cdots, n\}.$$

For each $i = 1, \ldots, p$, solve the linear programming subproblem ($\mathrm{P}_i(H^0)$).

If the subproblem ($\mathrm{P}_i(H^0)$) is not feasible, then the problem (SLRP) is also not feasible and the algorithm will be terminated.

Otherwise, obtain the optimal solution ($y_i(H^0)$, $s_i(H^0)$) and optimal value $\phi_i(H^0)$ of the subproblem ($\mathrm{P}_i(H^0)$), and let $LB_0 = \sum_{i=1}^{p} \phi_i(H^0)$

For each $i = 1, \ldots, p$, let $x_i(H^0) = \frac{s_i(H^0)}{y_i(H^0)}$, and let

$$UB_0 = \min_{i \in \{1,2,\ldots,p\}} f(x_i(H^0)),$$

and

$$x^0 = \arg \min_{i \in \{1,2,\ldots,p\}} f(x_i(H^0)).$$

If $UB_0 - LB_0 \leq \epsilon$, the algorithm stops and x^0 is an ϵ-global optimum solution for the problem (SLRP). Otherwise, let $Z^0 = \{H^0\}$, $\Lambda = \{x^0\}$, $k = 0$, and proceed to Step 1.

Step 1: Applying the branching rule to partition the rectangle H^k into two sub-rectangles H_1^k and H_2^k, forming the set $\mathcal{H} = \{H_1^k, H_2^k\}$. For each $H_t^k \in \mathcal{H}$, $t \in \{1, 2\}$, solve the linear programming subproblem ($\mathrm{P}_i(H_t^k)$).

If the subproblem ($\mathrm{P}_i(H_t^k)$) is not feasible, then the problem (EP1(H_t^k)) is also not feasible, and let $\mathcal{H} = \mathcal{H} \setminus H_t^k$. Otherwise, obtain the optimal solution ($y_i(H_t^k)$, $s_i(H_t^k)$) and optimal value $\phi_i(H_t^k)$ of the subproblem ($\mathrm{P}_i(H_t^k)$), and let $LB(H_t^k) = \sum_{i=1}^{p} \phi_i(H_t^k)$.

If $LB(H_t^k) > UB_k$, let $\mathcal{H} = \mathcal{H} \setminus H_t^k$. Otherwise, let $x_i(H_t^k) = \frac{s_i(H_t^k)}{y_i(H_t^k)}$ and let

$$UB(H_t^k) = \min_{i \in \{1,2,\ldots,p\}} f(x_i(H_t^k)),$$

and

$$x(H_t^k) = \arg \min_{i \in \{1,2,\ldots,p\}} f(x_i(H_t^k)).$$

Also, update the upper bound $UB_k = \min\{UB_k, UB(H_t^k)\}$ and $\Lambda = \Lambda \cup \{x(H_t^k)\}$. Meanwhile, refer to x^k as the current best feasible solution for the problem (SLRP), which satisfies $UB_k = f(x^k)$.

Step 2: Update the set $Z_k = (Z_k \setminus H^k) \bigcup \mathcal{H}$ and the lower bound $LB_k = \min\{LB(H) \mid H \in Z_k\}$.

Step 3: If $UB_k - LB_k \leq \epsilon$, the algorithm stops and x^k is an ϵ-global optimal solution for the problem (SLRP). Otherwise, let $k = k + 1$, and select a sub-rectangle H^k satisfying $LB_k = LB(H^k)$, and return to Step 1.

3.3. Global convergence analysis

Theorem 3.1. *For any given $\epsilon \in [0, 1)$, the algorithm will either terminate after finite iterations with yielding a global optimal solution of the problem (SLRP), or it will generate an infinite sequence $\{x^k\}$ of feasible solutions, whose accumulation point will be a global optimal solution of the problem (SLRP).*

Proof. If the algorithm terminates after finite k iterations, we can obtain the best feasible solution x^k for the problem (SLRP). For any given $\epsilon > 0$, we have that

$$UB_k - LB_k \leq \epsilon. \tag{3.1}$$

Let v be the optimal value of the problem (SLRP), and we can get that

$$LB_k \leq v \text{ and } UB_k \geq v. \tag{3.2}$$

By combining (3.1) and (3.2), we can derive the following inequality:

$$f(x^k) + \epsilon = UB_k + \epsilon \geq v + \epsilon \geq LB_k + \epsilon \geq UB_k = f(x^k). \tag{3.3}$$

Therefore, x^k is an ϵ-global optimal solution to the problem (SLRP).

If the algorithm generates an infinite solution sequence $\{x^k\}$, let x^* be any accumulation point of sequence $\{x^k\}$. Without loss of generality, we assume that

$$\lim_{k\to\infty} x^k = x^*.$$

Then, by the branch-and-bound process of the algorithm, we can conclude that

$$LB_k \leq f(x^*) \leq f(x^k) = UB_k,\ k = 0, 1, 2 \ldots. \tag{3.4}$$

Since $\{LB_k\}$ is a monotonic nondecreasing sequence, and $\{UB_k\}$ is a nonincreasing sequence, both of them are convergent sequences. Thus, taking the limits on both sides of (3.4), we get that

$$\lim_{k\to\infty} LB_k \leq f(x^*) \leq \lim_{k\to\infty} f(x^k) = \lim_{k\to\infty} UB_k.$$

By combining the continuity of the function $f(x)$ with the termination condition that $UB_k - LB_k \leq \epsilon$ for any given $\epsilon \in [0, 1)$, it can deduce that

$$\lim_{k\to\infty} LB_k = f(x^*) = \lim_{k\to\infty} f(x^k) = \lim_{k\to\infty} UB_k. \tag{3.5}$$

Thus, the accumulation point x^* of the sequence $\{x^k\}$ is a global optimal solution of the problem (SLRP), and the proof is complete. □

3.4. Computational complexity analysis

In this subsection, we provide the computational complexity analysis of the algorithm to estimate the maximum number of iterations.

First, for any $H^k = [l^k, u^k] \triangleq \{x \in R^n \mid l_j^k \leq x_j \leq u_j^k,\ j = 1, \ldots, n\} \subseteq H^0$, we define

$$\sigma_i = \min_{x\in\chi} e_i^\top x + g_i,\ i = 1, \ldots, p, \tag{3.6}$$

$$\rho = \max_{i\in\{1,\ldots,p\}, j\in\{1\ldots,n\}} \left|g_i d_{ij} + c_i e_{ij}\right|, \tag{3.7}$$

$$\eta = \max_{i\in\{1,\ldots,p\}} \frac{1}{\sigma_i^2}. \tag{3.8}$$

Theorem 3.2. *For any given termination error ϵ, during the k_{th} iteration of the algorithm, if the generated rectangle H^k satisfies*

$$u_{\hat{j}}^k - l_{\hat{j}}^k \leq \frac{\epsilon}{p\eta n\rho},$$

then the algorithm will terminate and return an ϵ-globally optimal solution of the problem (SLRP).

Proof. Without losing generality, suppose that $(\hat{y}^k, \hat{s}^k)$ is the optimal solution of the problem (LRP). Let $\hat{x}_i^k = \frac{\hat{s}_i^k}{\hat{y}_i^k}$, and we have

$$x^k = \arg\min_{i\in\{1,2,\ldots,p\}} f(\hat{x}_i^k),$$

$$y_i^k = \frac{1}{e_i^\top x^k + g_i}, i = 1\ldots, p,$$

and

$$s_i^k = y_i^k x^k, i = 1, \ldots, p,$$

Obviously, (x^k, y^k, s^k) is a feasible solution to the problem (EP1) at the k-th iteration, and we have that

$$LB_k = \phi(\hat{y}^k, \hat{s}^k) \leq UB_k = f(x^k) = \phi(y^k, s^k).$$

Then, when the generated rectangle H^k satisfies $u_{\hat{j}}^k - l_{\hat{j}}^k \leq \frac{\epsilon}{p\eta n\rho}$, we get that

$$\begin{aligned}
UB_k - LB_k &= \phi(y^k, s^k) - \phi(\hat{y}^k, \hat{s}^k) \\
&= \sum_{i=1}^{p}(d_i^\top s_i^k + c_i y_i^k) - \sum_{i=1}^{p}(d_i^\top \hat{s}_i^k + c_i \hat{y}_i^k) \\
&= \sum_{i=1}^{p} d_i(s_i - \hat{s}_i^k) + \sum_{i=1}^{p} c_i(y_i^k - \hat{y}_i^k) \\
&= \sum_{i=1}^{p} \frac{g_i d_i^\top (x^k - \hat{x}^k)}{(e_i^\top x^k + g_i)(e_i^\top \hat{x}^k + g_i)} + \sum_{i=1}^{p} \frac{c_i e_i^\top (x^k - \hat{x}^k)}{(e_i^\top x^k + g_i)(e_i^\top \hat{x}^k + g_i)} \\
&= \sum_{i=1}^{p} \frac{(g_i d_i^\top + c_i e_i^\top)(x^k - \hat{x}^k)}{(e_i^\top x^k + g_i)(e_i^\top \hat{x}^k + g_i)} \\
&\leq \sum_{i=1}^{p} \frac{1}{\sigma_i^2} \times \sum_{j=1}^{n} \left|g_i d_{ij} + c_i e_{ij}\right| (u_j^k - l_j^k) \\
&\leq p\eta n\rho(u_{\hat{j}}^k - l_{\hat{j}}^k) \\
&\leq \epsilon.
\end{aligned}$$

Therefore, when the generated rectangle H^k satisfies $u_{\hat{j}}^k - l_{\hat{j}}^k \leq \frac{\epsilon}{p\eta n\rho}$, the algorithm will be terminated. At the same time, referring to υ^* as the optimal value of the problem (SLRP), from Steps 0, 2, and 3 of the algorithm, noting that $\upsilon^* \leqslant UB_k = f(x^k)$, $LB_k \leqslant \upsilon^*$, and the termination condition $UB_k - LB_k \leq \epsilon$, we can further follow that $\upsilon^* \leqslant f(x^k) \leqslant \epsilon + LB_k \leqslant \epsilon + \upsilon^*$. Then, the algorithm returns an ϵ-globally optimum solution of the problem (SLRP), and the proof is complete. □

Theorem 3.3. *Given an error ϵ >0, the algorithm can find a global ϵ-optimal solution in at most*

$$\left\lceil \prod_{j=1}^{n} \max\left\{\frac{p\eta n\rho}{\epsilon}(u_j^0 - l_j^0), 1\right\}\right\rceil$$

iterations.

Proof. If the algorithm does not terminate at the k-th iteration, then in the previous iteration, at most $k+1$ sub-rectangles are partitioned from the initial rectangle H^0. Based on Theorem 3.2, in the k-th iteration, we choose $\hat{j}$ and get $u_{\hat{j}}^k - l_{\hat{j}}^k \geq \frac{\epsilon}{p\eta n\rho}$. Since the interval $[l_{\hat{j}}^k, u_{\hat{j}}^k]$ is divided into two subintervals at the midpoint $\frac{l_{\hat{j}}^k + u_{\hat{j}}^k}{2}$, the resulting two subintervals satisfy $u_{\hat{j}}^{k1} - l_{\hat{j}}^{k1} = u_{\hat{j}}^{k2} - l_{\hat{j}}^{k2} \geq \frac{\epsilon}{2p\eta n\rho}$. This indicates that each rectangle $H = \prod_{j=1}^{n}[l_j, u_j]$ satisfies $u_j - l_j \geq \min\{\frac{\epsilon}{2p\eta n\rho}, u_j^0 - l_j^0\},\ j = 1, \ldots, n$. Here, it should be noted that, when $u_j^0 - l_j^0 \leq \frac{\epsilon}{p\eta n\rho}$, the j-th direction has never been divided in these k iterations.

For all $k+1$ sub-rectangles, define their total volume as V, then we have

$$V \geq (k+1)\prod_{j=1}^{n} \min\left\{\frac{\epsilon}{2p\eta n\rho}, u_j^0 - l_j^0\right\}. \tag{3.9}$$

Besides,

$$V = \prod_{j=1}^{n}(u_j^0 - l_j^0) \tag{3.10}$$

is apparent. Thus, from (3.9) and (3.10), it can be concluded that

$$\prod_{j=1}^{n}(u_j^0 - l_j^0) \geq (k+1)\prod_{j=1}^{n} \min\left\{\frac{\epsilon}{2p\eta n\rho}, u_j^0 - l_j^0\right\}. \tag{3.11}$$

This indicates that

$$k \leq \prod_{j=1}^{n} \max\left\{\frac{2p\eta n\rho}{\epsilon}(u_j^0 - l_j^0), 1\right\}.$$

In other words, by Theorem 3.2, the algorithm iterates at most

$$\left\lceil \prod_{j=1}^{p} \max\left\{\frac{2p\eta n\rho}{\epsilon}(u_j^0 - l_j^0), 1\right\}\right\rceil$$

times and will be terminated. If not, we have that

$$(k+1)\prod_{j=1}^{p} \min\left\{\frac{\epsilon}{2p\eta n\rho}, u_j^0 - l_j^0\right\} > \prod_{j=1}^{n} \max\{\frac{2p\eta n\rho}{\epsilon}(u_j^0 - l_j^0), 1\} \times \min\{\frac{\epsilon}{2p\eta n\rho}, u_j^0 - l_j^0\}. \tag{3.12}$$

Subsequently, we will consider the following two cases:
(i) If $\frac{\epsilon}{2p\eta n\rho} < u_j^0 - l_j^0$, then we have

$$\max\{\frac{2p\eta n\rho}{\epsilon}(u_j^0 - l_j^0), 1\} \times \min\{\frac{\epsilon}{2p\eta n\rho}, u_j^0 - l_j^0\} = \frac{2p\eta n\rho}{\epsilon}(u_j^0 - l_j^0) \times \frac{\epsilon}{2p\eta n\rho} = u_j^0 - l_j^0. \tag{3.13}$$

(ii) If $\frac{\epsilon}{2p\eta n\rho} > u_j^0 - l_j^0$, then we have

$$\max\{\frac{2p\eta n\rho}{\epsilon}(u_j^0 - l_j^0), 1\} \times \min\{\frac{\epsilon}{2p\eta n\rho}, u_j^0 - l_j^0\} = 1 \times (u_j^0 - l_j^0) = u_j^0 - l_j^0. \tag{3.14}$$

By (3.10)–(3.14), it can be concluded that

$$\prod_{j=1}^{n}(u_j^0 - l_j^0) = V > \prod_{j=1}^{n}(u_j^0 - l_j^0).$$

Obviously, this is a contradiction. Thus, the algorithm can find a global ϵ-optimal solution in at most $\left\lceil \prod_{j=1}^{n} \max\{\frac{2p\eta n\rho}{\epsilon}(u_j^0 - l_j^0), 1\} \right\rceil$ iterations. This proof is complete. □

Remark 3.1. *According to the complexity theory of algorithms, the complexity of algorithms is an exponential function of the dimensionality n of variable x. Therefore, the performance of algorithms is sensitive to the number n of variables x. When n is fixed, the average running time of the algorithm increases with the increase of p.*

4. Numerical comparisons

In this section, we present computational results of the proposed algorithm on the randomly generated test problems. The algorithm is coded in Matlab(2023a) and all computations are implemented on an AMD Ryzen 5 5000 CPU 2.1GHz with 16G memory microcomputer.

First of all, some small-size deterministic examples (see Examples 1–12 in the Appendix) are tested with our algorithm for comparison with the known extant algorithms [24, 32], and numerical comparisons between some existing algorithms and our algorithm on Examples 1–10 are reported in Table 1 with the given convergence tolerance, where some notations have been used for column headers in Table 1: Opt. val.: global optimal value; Iter.: number of iterations of the algorithm; Time: the CPU execution time of the algorithm in seconds.

From the numerical results in Table 1, for Examples 1–10, we can follow that our algorithm can obtain the same global optimal solutions and optimal values as the existing algorithms of [24, 32]. Our algorithm performs better than the methods of [24, 32] for finding the optimal solution in less time and fewer iterations. Therefore, in terms of test Examples 1–10, the experimental results verify that our algorithm is valid and feasible.

Second, we solve the following randomly generated test problem:

Problem 1.

$$\begin{cases} \min \ \sum\limits_{i=1}^{p} \dfrac{\hat{c}_i^\top x + \hat{f}_i}{\hat{e}_i^\top x + \hat{g}_i}, \\ \text{s.t.} \ \ Ax \le b, \ x > 0, \end{cases}$$

where $\hat{c}_i \in R^n$, $\hat{e}_i \in R^n$, $\hat{f}_i \in R$, $\hat{g}_i \in R$, $i = 1, 2, \ldots, p$; $A \in R^{m\times n}$, $b \in R^m$; each element of $\hat{c}_i$ and $\hat{e}_i$ is randomly generated from the interval [0, 1]; every element in the matrix A and b is randomly generated from the interval [0, 1], and $\hat{f}_i$ and $\hat{g}_i$ are equal to 100.

For Problem 1, we record the number of iterations and computational time for the experimental data. For each set of parameters with the different (p, m, n), each element of $\hat{c}_i$ and $\hat{e}_i$ is randomly generated from $[0, 1]$, each element of A and b is randomly generated from $[0, 1]$, and $\hat{f}_i$ and $\hat{g}_i$ are equal to 100. We perform 10 calculations to obtain the average number of iterations and average computational time (in seconds), and we also separately record the minimum, maximum, and average values of the number of iterations and computational time. The termination error in numerical experiments is 10^{-2}. The resulting experimental data is presented in Table 2.

Table 1. Numerical comparisons between some existing algorithms and our algorithm on test Examples 1–10.

No.	Algorithms	Opt. val.	Optimal solution	Iter.	Time	ϵ
1	Ours	-4.84151	(0.1000, 2.3750)	1	0.0890	10^{-6}
	Algorithm of [24]	-4.84151	(0.1000, 2.3750)	4	0.1283	10^{-6}
	Algorithm of [27]	-4.84151	(0.1000, 2.3750)	14	0.3654	10^{-6}
2	Ours	-2.47143	(1.0000, 0.0000, 0.0000)	73	1.1267	10^{-6}
	Algorithm of [24]	-2.47143	(1.0000, 0.0000, 0.0000)	82	1.3452	10^{-6}
	Algorithm of [27]	-2.47143	(1.0000, 0.0000, 0.0000)	113	2.0513	10^{-6}
3	Ours	-1.90000	(0.0000, 3.3333, 0.0000)	1	0.1245	10^{-6}
	Algorithm of [24]	-1.90000	(0.0000, 3.3333, 0.0000)	8	0.945	10^{-6}
	Algorithm of [27]	-1.90000	(0.0000, 3.3333, 0.0000)	424	6.907	10^{-6}
4	Ours	-4.09070	(1.1111, 0.0000, 0.0000)	1	0.1392	10^{-6}
	Algorithm of [24]	-4.09070	(1.1111, 0.0000, 0.0000)	4	0.4386	10^{-6}
	Algorithm of [27]	-4.09070	(1.1111, 0.0000, 0.0000)	82	2.359	10^{-6}
5	Ours	3.71092	(0.0000, 1.6667, 0.0000)	1	0.0274	10^{-6}
	Algorithm of [24]	3.71092	(0.0000, 1.6667, 0.0000)	4	0.1356	10^{-6}
	Algorithm of [27]	3.71092	(0.0000, 1.6667, 0.0000)	112	1.4586	10^{-6}
6	Ours	-3.00292	(0.0000, 3.3333, 0.0000)	1	0.1156	10^{-6}
	Algorithm of [24]	-3.00292	(0.0000, 3.3333, 0.0000)	89	1.2364	10^{-6}
	Algorithm of [27]	-3.00225	(0.0000, 2.8455, 0.0000)	132	2.5458	10^{-6}
7	Ours	4.91259	(1.5000, 1.5000)	165	3.9902	10^{-6}
	Algorithm of [24]	4.91259	(1.5000, 1.5000)	166	4.0871	10^{-6}
	Algorithm of [27]	4.91259	(1.5000, 1.5000)	330	8.4212	10^{-6}
8	Ours	-4.09070	(1.1111, 0.0000, 0.0000)	1	0.0376	10^{-6}
	Algorithm of [24]	-4.09070	(1.1111, 0.0000, 0.0000)	8	0.1090	10^{-6}
	Algorithm of [27]	-4.09070	(1.1111, 0.0000, 0.0000)	977	32.41	10^{-6}
9	Ours	3.29167	(3.0000, 4.0000)	1	0.0282	10^{-6}
	Algorithm of [24]	3.29167	(3.0000, 4.0000)	9	0.489	10^{-6}
	Algorithm of [27]	3.29167	(3.0000, 4.0000)	138	1.902	10^{-6}
10	Ours	4.42857	(5.0000, 0.0000, 0.0000)	1	0.0308	10^{-6}
	Algorithm of [24]	4.42857	(5.0000, 0.0000, 0.0000)	12	0.2132	10^{-6}
	Algorithm of [27]	4.42857	(5.0000, 0.0000, 0.0000)	98	1.2931	10^{-6}

Table 2. Experimental results on the randomly generated test Problem 1.

	Iter			Time		
(p, m, n)	min.	ave.	max.	min.	ave.	max.
(50,10,2)	0	3.9	12	0.17265	1.585562	4.73441
(100,10,2)	1	26.8	71	1.17688	21.305212	54.18782
(150,10,2)	1	37.8	117	1.86746	46.537853	139.25536
(200,10,2)	1	38	152	2.37256	59.682521	125.03535
(300,10,2)	2	45.1	147	4.92376	91.516705	294.0104
(400,10,2)	2	133.6	459	6.28794	345.899427	1168.62086
(500,10,2)	2	149.2	390	7.84995	470.904921	1225.48065
(600,10,2)	3	78.2	209	13.58283	506.971382	1642.41682
(700,10,2)	27	330.3	826	123.55211	613.648123	1536.35623
(800,10,2)	32	312.9	555	165.4178	679.852562	1705.53504
(900,10,2)	82	318.5	481	561.23789	984.004699	3966.5198
(1000,10,2)	63	295.7	987	690.18154	1030.719038	2163.83152
(100,10,3)	2	77.3	302	1.54206	48.773773	190.15702
(200,10,3)	2	113.7	518	3.30311	161.120675	770.38062
(300,10,3)	3	289	601	8.36701	584.99691	1153.47521
(400,10,3)	10	264	900	26.29532	685.743374	2351.9021
(500,10,3)	1	381.9	1335	5.47715	1026.24487	3461.11334
(50,10,4)	9	107.6	276	3.92405	40.717008	79.75275
(100,10,4)	1	169	335	0.31642	106.856346	211.11143
(200,10,4)	1	85.6	216	2.38088	125.808901	303.00551
(300,10,4)	1	246.7	993	3.39098	484.209614	1946.05488

By the computational results for the randomly generated test problems, it can be seen that the algorithm can effectively solve the sum of linear ratios problem with a substantial quantity of ratios and low dimensional variables.

Remark 4.1. *From Table 2, we can see that the algorithm's performance is sensitive to variations in the number of variables. However, when n is fixed, the average running time of the algorithm increases with the increase of p. In addition, we don't find that the specific problem characteristics (e.g., sparsity, degeneracy) significantly impact the algorithm's performance.*

5. Conclusions

This paper present a branch-and-bound algorithm for globally addressing the SLRP with a substantial quantity of ratios and low dimensional variables. For addressing the problem (SLRP), we first transform it into the EP using the Charnes-Cooper transformation, and then apply the linear relaxation technique to construct the LRP of the problem (EP). Subsequently, based on the branch-and-bound framework, we propose a global algorithm using the separability of the problem (LRP). Finally, we validate the effectiveness of the algorithm by solving large-scale stochastic test problems. In future work, we will promote the separability of relaxation problems and design global optimization

algorithms for solving the generalized linear fractional multiplicative programming problem with a substantial quantity of ratios and low dimensional variables.

Author contributions

Qunzhen Zheng: Formal analysis, investigation, resources, methodology, writing-original draft, validation, data curation, and funding acquisition; Chenglin He: Formal analysis, investigation, writing review & editing, software, data curation; Yan Shi: Conceptualization, supervision; Jingben Yin: Project administration, methodology, validation, and formal funding acquisition. All authors have read and agreed to the published version of the manuscript.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This paper is supported by the Key Scientific and Technological Research Projects of Henan Province (202102210147, 192102210114).

Conflict of interest

The authors declare that they have no conflicts of interest.

References

1. E. Arkin, Y. Chiang, M. Held, J. Mitchell, V. Sacristan, S. Skiena, et al., On minimum-area hulls, *Algorithmica*, **21** (1998), 119–136. https://doi.org/10.1007/PL00009204
2. E. B. Bajalinov, *Linear-fractional programming theory, methods, applications and software*, Dordrecht: Kluwer Academic Publishers, 2003. https://doi.org/10.1007/978-1-4419-9174-4
3. H. P. Benson, Vector maximization with two objective functions, *J. Optim. Theory Appl.*, **28** (1979), 253–257. https://doi.org/10.1007/BF00933245
4. H. P. Benson, A simplicial branch and bound duality-bounds algorithm for the linear sum-of-ratios problem, *Eur. J. Oper. Res.*, **182** (2007), 597–611. https://doi.org/10.1016/j.ejor.2006.08.036
5. M. Borza, A. S. Rambely, A linearization to the sum of linear ratios programming problem, *Mathematics*, **9** (2021), 1004. https://doi.org/10.3390/math9091004
6. A. Charnes, W. W. Cooper, Programming with linear fractional functionals, *Naval Research Logistics Quarterly*, **9** (1962), 181–186. https://doi.org/10.1002/nav.3800090303
7. C. S. Colantoni, R. P. Manes, A. Whinston, Programming, profit rates and pricing decisions, *Account. Rev.*, **44** (1969), 467–481.
8. D. F. Dennis, Analyzing public inputs to multiple objective decisions on national forests using conjoint analysis, *Forest Sci.*, **44** (1998), 421–429. https://doi.org/10.1093/forestscience/44.3.421

9. J. E. Falk, S. W. Palocsay, Optimizing the sum of linear fractional functions, In: *Recent advances in global optimization*, Princeton: Princeton University Press, 1991, 221–258. https://doi.org/10.1515/9781400862528.221
10. Z. Hou, S. Liu, An efficient image space branch-reduction-bound algorithm to globally solve generalized fractional programming problems for large-scale real applications, *J. Comput. Appl. Math.*, **451** (2024), 116070. https://doi.org/10.1016/j.cam.2024.116070
11. Z. Hou, S. Liu, A spatial branch-reduction-bound algorithm for solving generalized linear fractional problems globally, *Chaos Soliton. Fract.*, **176** (2023), 114144. https://doi.org/10.1016/j.chaos.2023.114144
12. Z. Hou, S. Liu, An accelerating outer space algorithm for globally solving generalized linear multiplicative problems, *Numer. Algor.*, **94** (2023), 877–904. https://doi.org/10.1007/s11075-023-01523-y
13. H. Jiao, S. Liu, A practicable branch and bound algorithm for sum of linear ratios problem, *Eur. J. Oper. Res.*, **243** (2015), 723–730. https://doi.org/10.1016/j.ejor.2015.01.039
14. H. Jiao, S. Liu, Y. Zhao, Effective algorithm for solving the generalized linear multiplicative problem with generalized polynomial constraints, *Appl. Math. Model.*, **39** (2015), 7568–7582. https://doi.org/10.1016/j.apm.2015.03.025
15. H. Jiao, B. Li, Y. Shang, An outer space approach to tackle generalized affine fractional program problems, *J. Optim. Theory Appl.*, **201** (2024), 1–35. https://doi.org/10.1007/s10957-023-02368-0
16. H. Jiao, J. Ma, P. Shen, Y. Qiu, Effective algorithm and computational complexity for solving sum of linear ratios problem, *J. Ind. Manag. Optim.*, **19** (2023), 4410–4427. https://doi.org/10.3934/jimo.2022135
17. H. Jiao, J. Ma, An efficient algorithm and complexity result for solving the sum of general ratios problem, *Chaos Soliton. Fract.*, **164** (2022), 112701. https://doi.org/10.1016/j.chaos.2022.112701
18. H. Jiao, Y. Shang, R. Chen, A potential practical algorithm for minimizing the sum of affine fractional functions, *Optimization*, **72** (2023), 1577–1607. https://doi.org/10.1080/02331934.2022.2032051
19. H. Jiao, Y. Shang, Two-level linear relaxation method for generalized linear fractional programming, *J. Oper. Res. Soc.*, **11** (2023), 569–594. https://doi.org/10.1007/s40305-021-00375-4
20. H. Jiao, Y. Shang, W. Wang, Solving generalized polynomial problem by using new affine relaxed technique, *Int. J. Comput. Math.*, **99** (2022), 309–331. https://doi.org/10.1080/00207160.2021.1909727
21. H. Jiao, B. Li, W. Yang, A criterion-space branch-reduction-bound algorithm for solving generalized multiplicative problems, *J. Glob. Optim.*, **89** (2024), 597–632. https://doi.org/10.1007/s10898-023-01358-w
22. H. Jiao, J. Ma, Optimizing generalized linear fractional program using the image space branch-reduction-bound scheme, *Optimization*, **74** (2025), 1–32. https://doi.org/10.1080/02331934.2023.2253816

23. H. Jiao, J. Ma, Y. Shang, Reduced outer space algorithm for globally computing affine sum-of-ratios problems, *Asia Pac. J. Oper. Res.*, **42** (2025), 2450015. https://doi.org/10.1142/S0217595924500155

24. H. Jiao, Y. Shang, Image space branch-reduction-bound algorithm for globally solving the sum of affine ratios problem, *J. Comput. Math.*, **43** (2025), 203–228. https://doi.org/10.4208/jcm.2203-m2021-0085

25. H. Konno, M. Inori, Bond portfolio optimization by bilinear fractional programming, *J. Oper. Res. Soc. Jpn.*, **32** (1989), 143–158. https://doi.org/10.15807/jorsj.32.143

26. H. Li, L. Wang, Y. Zhao, Global optimization algorithm for a class of linear ratios optimization problem, *AIMS Mathematics*, **9** (2024), 16376–16391. https://doi.org/10.3934/math.2024793

27. H. Li, Y. Feng, H. Jiao, Y. Shang, A novel algorithm for solving sum of several affine fractional functions, *AIMS Mathematics*, **8** (2023), 9247–9264. https://doi.org/10.3934/math.2023464

28. T. Matsui, NP-hardness of linear multiplicative programming and related problems, *J. Glob. Optim.*, **9** (1996), 113–119. https://doi.org/10.1007/BF00121658

29. N. Phuong, H. Tuy, A unified monotonic approach to generalized linear fractional programming, *J. Global Optim.*, **26** (2003), 229–259. https://doi.org/10.1023/A:1023274721632

30. P. Shen, Y. Wang, D. Wu, A spatial branch and bound algorithm for solving the sum of linear ratios optimization problem, *Numer. Algor.*, **93** (2023), 1373–1400. https://doi.org/10.1007/s11075-022-01471-z

31. S. Schaible, Fractional programming, In: *Handbook of global optimization*, Boston: Springer, 1995, 495–608. https://doi.org/10.1007/978-1-4615-2025-2_10

32. P. P. Shen, C. F. Wang, Global optimization for sum of linear ratios problem with coefficients, *Appl. Math. Comput.*, **176** (2006), 219–229. https://doi.org/10.1016/j.amc.2005.09.047

33. I. M. Stancu-Minasian, A ninth bibliography of fractional programming, *Optimization*, **68** (2019), 2125–2169. https://doi.org/10.1080/02331934.2019.1632250

34. I. M. Stancu-Minasian, A eighth bibliography of fractional programming, *Optimization*, **66** (2017), 439–470. https://doi.org/10.1080/02331934.2016.1276179

35. A. Q. Tian, H. X. Lv, X. Y. Wang, J. S. Pan, V. Sn*á*šel, Bioinspired discrete two-stage surrogate-assisted algorithm for large-scale traveling salesman problem, *J. Bionic. Eng.*, **22** (2025), 1926–1939. https://doi.org/10.1007/s42235-025-00724-6

36. A. Q. Tian, F. F. Liu, H. X. Lv, Snow Geese algorithm: a novel migration-inspired meta-heuristic algorithm for constrained engineering optimization problems, *Appl. Math. Model.*, **126** (2024), 327–347. https://doi.org/10.1016/j.apm.2023.10.045

37. D. Zhan, A. Q. Tian, S. Q. Ni, Optimizing PID control for multi-model adaptive high-speed rail platform door systems with an improved metaheuristic approach, *Int. J. Elec. Power.*, **169** (2025), 110738. https://doi.org/10.1016/j.ijepes.2025.110738

Appendix

Some deterministic examples are given as follows.

Example 1. ([24])

$$\begin{cases} \min & f(x) = \frac{-3.333x_1-3x_2-1}{1.666x_1+x_2+1} + \frac{-4x_1-3x_2-1}{x_1+x_2+1}, \\ \text{s.t.} & 5x_1 + 4x_2 \leq 10, \\ & -x_1 \leq -0.1, \\ & -x_2 \leq -0.1, \\ & -2x_1 - x_2 \leq -2, \\ & x_1, x_2 \geq 0. \end{cases}$$

Example 2. ([24,29])

$$\begin{cases} \max & \frac{3x_1+x_2-2x_3+0.8}{2x_1-x_2+x_3} + \frac{4x_1-2x_2+x_3}{7x_1+3x_2-x_3}, \\ \text{s.t.} & x_1 + x_2 - x_3 \leq 1, \\ & -x_1 + x_2 - x_3 \leq -1, \\ & 12x_1 + 5x_2 + 12x_3 \leq 34.8, \\ & 12x_1 + 12x_2 + 7x_3 \leq 29.1, \\ & -6x_1 + x_2 + x_3 \leq -4.1. \end{cases}$$

Example 3. ([24,32])

$$\begin{cases} \max & \frac{3x_1+4x_2+50}{3x_1+5x_2+4x_3+50} - \frac{3x_1+5x_2+3x_3+50}{5x_1+5x_2+4x_3+50} - \frac{x_1+2x_2+4x_3+50}{5x_2+4x_3+50} - \frac{4x_1+3x_2+3x_3+50}{3x_2+3x_3+50}, \\ \text{s.t.} & 6x_1 + 3x_2 + 3x_3 \leq 10, \\ & 10x_1 + 3x_2 + 8x_3 \leq 10, \\ & x_1, x_2, x_3 \geq 0. \end{cases}$$

Example 4. ([24])

$$\begin{cases} \max & \frac{4x_1+3x_2+3x_3+50}{3x_2+3x_3+50} + \frac{3x_1+4x_3+50}{4x_1+4x_2+5x_3+50} + \frac{x_1+2x_2+5x_3+50}{x_1+5x_2+5x_3+50} + \frac{x_1+2x_2+4x_3+50}{5x_2+4x_3+50}, \\ \text{s.t.} & 2x_1 + x_2 + 5x_3 \leq 10, \\ & x_1 + 6x_2 + 3x_3 \leq 10, \\ & 5x_1 + 9x_2 + 2x_3 \leq 10, \\ & 9x_1 + 7x_2 + 3x_3 \leq 10, \\ & x_1, x_2, x_3 \geq 0. \end{cases}$$

Example 5. ([24])

$$\begin{cases} \min & \frac{4x_1+3x_2+3x_3+50}{3x_2+3x_3+50} + \frac{3x_1+4x_3+50}{4x_1+4x_2+5x_3+50} + \frac{x_1+2x_2+4x_3+50}{x_1+5x_2+5x_3+50} + \frac{x_1+2x_2+4x_3+50}{5x_2+4x_3+50}, \\ \text{s.t.} & 2x_1 + x_2 + 5x_3 \leq 10, \\ & x_1 + 6x_2 + 3x_3 \leq 10, \\ & 9x_1 + 7x_2 + 3x_3 \geq 10, \\ & x_1, x_2, x_3 \geq 0. \end{cases}$$

Example 6. ([24,32])

$$\begin{cases} \max & \frac{3x_1+5x_2+3x_3+50}{3x_1+4x_2+5x_3+50} + \frac{3x_1+4x_2+50}{4x_1+3x_2+2x_3+50} + \frac{4x_1+2x_2+4x_3+50}{5x_1+4x_2+3x_3+50}, \\ \text{s.t.} & 6x_1 + 3x_2 + 3x_3 \leq 10, \\ & 10x_1 + 3x_2 + 8x_3 \leq 10, \\ & x_1, x_2, x_3 \geq 0. \end{cases}$$

Example 7. ([24])

$$\begin{cases} \min & \frac{37x_1+73x_2+13}{13x_1+13x_2+13} + \frac{63x_1-18x_2+39}{13x_1+26x_2+13}, \\ \text{s.t.} & 5x_1 - 3x_2 = 3, \\ & 1.5 \leq x_1 \leq 3. \end{cases}$$

Example 8. ([24,32])

$$\begin{cases} \max & \frac{4x_1+3x_2+3x_3+50}{3x_2+2x_3+50} + \frac{3x_1+4x_2+50}{4x_1+4x_2+5x_3+50} + \frac{x_1+2x_2+5x_3+50}{x_1+5x_2+5x_3+50} + \frac{x_1+2x_2+4x_3+50}{5x_2+4x_3+50}, \\ \text{s.t.} & 2x_1 + x_2 + 5x_3 \leq 10, \\ & x_1 + 6x_2 + 3x_3 \leq 10, \\ & 5x_1 + 9x_2 + 2x_3 \leq 10, \\ & 9x_1 + 7x_2 + 3x_3 \leq 10, \\ & x_1, x_2, x_3 \geq 0. \end{cases}$$

Example 9. ([24,32])

$$\begin{cases} \max & \frac{37x_1+73x_2+13}{13x_1+13x_2+13} + \frac{63x_1-18x_2+39}{-13x_1-26x_2-13} + \frac{13x_1+13x_2+13}{63x_1-18x_2+39} + \frac{13x_1+26x_2+13}{-37x_2-73x_3-13}, \\ \text{s.t.} & 5x_1 - 3x_2 = 3, \\ & 1.5 \leq x_1 \leq 3. \end{cases}$$

Example 10. ([24])

$$\begin{cases} \max & \frac{4x_1+3x_2+3x_3+50}{3x_2+3x_3+50} + \frac{3x_1+4x_3+50}{4x_1+4x_2+5x_3+50} + \frac{x_1+2x_2+5x_3+50}{x_1+5x_2+5x_3+50} + \frac{x_1+2x_2+4x_3+50}{5x_2+4x_3+50}, \\ \text{s.t.} & 2x_1 + x_2 + 5x_3 \leq 10, \\ & x_1 + 6x_2 + 2x_3 \leq 10, \\ & 9x_1 + 7x_2 + 3x_3 \geq 10, \\ & x_1, x_2, x_3 \geq 0. \end{cases}$$