Mathematics

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

AIMS Mathematics, 9(9): 23837–23858.
DOI: 10.3934/math.20241158
Received: 27 April 2024
Revised: 17 July 2024
Accepted: 29 July 2024
Published: 09 August 2024

Research article

Effective outcome space branch-and-bound algorithm for solving the sum of affine ratios problem

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

[1] College of Information Engineering, Henan University of Animal Husbandry and Economy, Zhengzhou 450000, China; wqmshiyan@163.com

[2] School of Statistics and Mathematics, Henan Finance University, Zhengzhou 450046, China

[3] School of Mathematical Sciences, Henan Institute of Science and Technology, Xinxiang 453003, China

* **Correspondence:** Email: zhengqunzhen123@163.com, jingbenyin@163.com.

Abstract: This paper proposes an efficient method for acquiring the global solution of the sum of affine ratios problem (SARP) in the reduced outer space. Using equivalence conversions, the original problem was transformed into an equivalent problem. Then, an affine relaxation problem of the equivalent problem was constructed by exploiting linearization techniques. Subsequently, an outcome space branch-and-bound algorithm was proposed, the convergence of the algorithm was proved and the computational complexity was estimated. Finally, numerical examples were presented to demonstrate the effectiveness and feasibility of the presented algorithm.

Keywords: fractional programs; global optimization; sum of affine ratios problems; linearization technique; outcome space branch-and-bound algorithm
Mathematical Subject Classification: 90C32, 90C26

1. Introduction

In this paper, we consider the following sum of affine ratios problem (where SARP is sum of affine ratios problem):

$$\text{(SARP)}:\begin{cases} \min H(x) = \sum\limits_{i=1}^{p} \dfrac{\sum_{j=1}^{n} c_{ij}x_j+f_i}{\sum_{j=1}^{n} d_{ij}x_j+g_i}, \\ \text{s.t. } x \in D = \{x \in \mathbb{R}^n \mid Ax \leq b, x \geq 0\}, \end{cases}$$

where A is an $m \times n$ order real matrix and b is an m dimension column vector. $D \neq \emptyset$, $c_{ij}, d_{ij} \in \mathbb{R}^n$, and $f_i, g_i \in \mathbb{R}$. The denominator of each ratio

$$\sum_{j=1}^{n} d_{ij}x_j + g_i \neq 0$$

over D. Due to

$$\sum_{j=1}^{n} d_{ij}x_j + g_i \neq 0$$

and the continuity of the ratio

$$\frac{\sum_{j=1}^{n} c_{ij}x_j + f_i}{\sum_{j=1}^{n} d_{ij}x_j + g_i},$$

we can obtain that

$$\sum_{j=1}^{n} d_{ij}x_j + g_i < 0$$

or

$$\sum_{j=1}^{n} d_{ij}x_j + g_i > 0.$$

If

$$\sum_{j=1}^{n} d_{ij}x_j + g_i < 0,$$

letting

$$\frac{\sum_{j=1}^{n} c_{ij}x_j + f_i}{\sum_{j=1}^{n} d_{ij}x_j + g_i} = \frac{-\left(\sum_{j=1}^{n} c_{ij}x_j + f_i\right)}{-\left(\sum_{j=1}^{n} d_{ij}x_j + g_i\right)},$$

it is obvious that

$$-\left(\sum_{j=1}^{n} d_{ij}x_j + g_i\right) > 0.$$

Therefore, without loss of generality, we assume that

$$\sum_{j=1}^{n} d_{ij}x_j + g_i > 0$$

always holds.

The SARP is a specific class of fractional programming problem. It has a wide range of applications in laminate manufacturing [1, 2], portfolio optimization [3–5], finance and investment [6], computer vision [7], system engineering [8], information theory [9], material layout [10, 11], and so on. In addition, it is obvious that the objective function of the SARP is neither quasi-convex nor quasi-concave, which is extremely challenging in terms of theory and computation. Thus, the research on the approach to solve the SARP has both theoretical and practical significance value.

Up to now, a number of methods have been developed to solve the SARP. On the basis of the characteristics of the algorithms in each literature, it can be roughly separated into the following categories: the parametric simplex methods [12, 13], the unified monotonic approach [14], the interior-point method [15, 16], the image space analysis method [17], the trapezoidal branching searching algorithm [18], the branch-and-bound algorithms [19–21], and so on. It should be noted that the algorithms in the above references [19–21] can either only solve special forms of the SARP or the SARP with fewer variables. Recently, by exploiting equivalent transformation and the characteristics of general single ratio functions, Jiao and Ma [22] combined the acceleration technique to develop an efficient outer space rectangular branch-and-bound algorithm. Jiao et al. [23] put forward a practical algorithm for minimizing the SARP. In the same year, by using the equivalent conversion and linearization method, Jiao et al. [24] presented an effective branch-and-bound algorithm for the SARP. Meanwhile, Jiao et al. [25] also designed an image space branch-reduction-bound algorithm for globally solving the SARP. It should be emphasized that references [23–25] proposed several outer space branch-and-bound algorithms for the problem (SARP), and the partitioning spaces of these branch-and-bound algorithms all occur in the p-dimensional outer space. In addition, Pei and Zhu [26] designed a convex relaxation algorithm for maximizing the sum of the difference of convex functions ratios problems based on the branch-and-bound framework. Kuno [27] presented a trapezoidal branch-and-bound algorithm to solve the SARP. Based on the algorithm of Kuno [27], Shen et al. [28] proposed an accelerating trapezoidal branch-reduction-bound algorithm for globally solving the sum of linear ratios problems. By constructing the new accelerating technique, Jiao and Liu [29] developed a branch-reduction-bound algorithm for the sum of quadratic ratios problems. Especially, the literatures [30, 31] provided for the first time an outer space branch-relaxation-bound algorithm for generalized linear fractional programming problems and generalized affine multiple product programming problems, respectively. However, the above-reviewed methods only deal with some particular forms of the SARP or are difficult to solve the SARP with large-scale variables. Therefore, it is still necessary to propose a practical efficient algorithm for addressing the general form of the SARP.

The main purpose of this paper is to design an effective outcome space branch-and-bound algorithm to globally solve the SARP. Initially, we employ two equivalence conversions to transform the SARP into an equivalent problem (EP_3). Subsequently, we design a linearization technique that is used to construct the affine relaxation problem (ARP) of the problem (EP_3). Based on the ARP and the branch-and-bound framework, we present an outcome space branch-and-bound algorithm. Ultimately, the numerical experimental results demonstrate that our algorithm is feasible. In addition, the branch search process in this paper occurs in the $(p-1)$-dimensional outer space, which makes the computational cost much lower. Compared with other methods that take place in the n-dimension or p-dimensional space, our method is more efficient. Furthermore, the algorithm is shown to converge to a global optimal solution of the SARP eventually. Meanwhile, the computational complexity of the algorithm is deduced in detail.

The overall structure of the study adopts the format of six sections, including this introductory section. In Section 2, we convert the SARP to an equivalent problem (EP_3) and construct its affine relaxation programming problem. In Section 3, we design an outcome space branch-and-bound algorithm for globally solving the SARP, prove the convergence of the algorithm, and derive the derivation of the computational complexity. In Section 4, a comparison of the numerical experimental results indicates that the presented algorithm is reliable. Finally, some concluding remarks are given in Section 5.

2. Equivalence transformation and its affine relaxation programming

2.1. Equivalent transformation

To tackle the SARP globally, we transform the original problem (SARP) into an equivalence problem as below. For each $x \in D$, let

$$t = \frac{1}{\sum\limits_{j=1}^{n} d_{pj}x_j + g_p}$$

and

$$z_j = tx_j.$$

By utilizing the well-known Charnes-Cooper transformation [32], the SARP can be rewritten as problem (EP$_1$), as shown below.

$$(\text{EP}_1):\begin{cases} \min \hat{H}(t,z) = \sum\limits_{i=1}^{p-1} \frac{\sum_{j=1}^{n} c_{ij}z_j + f_i t}{\sum_{j=1}^{n} d_{ij}z_j + g_i t} + \sum\limits_{j=1}^{n} c_{pj}z_j + f_p t, \\ \text{s.t. } \sum\limits_{j=1}^{n} d_{pj}z_j + g_p t = 1, \\ Az \le bt,\ t \ge 0,\ z \ge 0. \end{cases}$$

At this time, the index set becomes $i \in \{1, 2, \ldots, p-1\}$, and let

$$\chi = \{(t,z) \in \mathbb{R}^{n+1} | \sum_{j=1}^{n} d_{pj}z_j + g_p t = 1,\ \ Az \le bt, t \ge 0, z \ge 0\},$$

where the set χ is non-empty and bounded.

Then we focus on the following equivalence conversion, for each $i \in \{1, 2, \ldots, p-1\}$, by introducing the variable

$$s_i = \frac{1}{\sum\limits_{j=1}^{n} d_{ij}z_j + g_i t},$$

the lower bound

$$\underline{s}_i^0 = \min_{(t,z)\in\chi} \frac{1}{\sum\limits_{j=1}^{n} d_{ij}z_j + g_i t}$$

and upper bound

$$\overline{s}_i^0 = \max_{(t,z)\in\chi} \frac{1}{\sum\limits_{j=1}^{n} d_{ij}z_j + g_i t}$$

can be calculated, and we can obtain the initial rectangle

$$S^0 = [\underline{s}^0, \overline{s}^0],$$

where

$$\underline{s}^0 = (\underline{s}_1^0, \underline{s}_2^0, \ldots, \underline{s}_{p-1}^0)$$

and

$$\overline{s}^0 = (\overline{s}_1^0, \overline{s}_2^0, \ldots, \overline{s}_{p-1}^0).$$

Further, the problem (EP$_1$) can be reduced into the problem (EP$_2$) by the variable s_i as follows:

$$(\mathrm{EP}_2): \begin{cases} \min\ \Phi(s,t,z) = \sum\limits_{i=1}^{p-1} s_i\Big(\sum\limits_{j=1}^{n} c_{ij}z_j + f_i t\Big) + \sum\limits_{j=1}^{n} c_{pj}z_j + f_p t, \\ \quad \text{s.t. } s_i = \frac{1}{\sum_{j=1}^{n} d_{ij}z_j + g_i t}, \quad i = 1,\ldots,p-1, \\ (t,z) \in \chi, s \in S^0. \end{cases}$$

Remark 1. If (t^*, z^*) is a global optimal solution of the problem (EP$_1$), then (s^*, t^*, z^*) is a global optimal solution of the problem (EP$_2$) with

$$s_i^* = \frac{1}{\sum\limits_{j=1}^{n} d_{ij}z_j^* + g_i t^*}, i = 1, 2, \ldots, p-1.$$

Conversely, if (s^*, t^*, z^*) is a global optimal solution of the problem (EP$_2$), then (t^*, z^*) is a global optimal solution of the problem (EP$_1$). In addition, the global optimal value of the problems (EP$_1$), (EP$_2$), and (SARP) are equal.

Based on

$$\sum_{j=1}^{n} d_{ij}z_j + g_i t \neq 0,$$

the problem (EP$_2$) can be rewritten as the following equivalent form:

$$(\mathrm{EP}_3): \begin{cases} \min \Phi(s,t,z) = \sum\limits_{i=1}^{p-1} s_i\Big(\sum\limits_{j=1}^{n} c_{ij}z_j + f_i t\Big) + \sum\limits_{j=1}^{n} c_{pj}z_j + f_p t \\ \text{s.t. } s_i\Big(\sum\limits_{j=1}^{n} d_{ij}z_j + g_i t\Big) = 1, \quad i = 1, \cdots, p-1, \\ (t,z) \in \chi, s \in S^0. \end{cases}$$

Define S to be S^0 or a sub-rectangle of S^0, where

$$S = [\underline{s}, \overline{s}] \subseteq [\underline{s}^0, \overline{s}^0], \quad \underline{s} = (\underline{s}_1, \underline{s}_2, \ldots, \underline{s}_{p-1}) \geq \underline{s}^0$$

and

$$\overline{s} = (\overline{s}_1, \overline{s}_2, \ldots, \overline{s}_{p-1}) \leq \overline{s}^0.$$

2.2. *Affine relaxation programming*

For any $S \subseteq S^0$, we construct the affine relaxation programming problem of the problem (EP$_3$) over S as follows:

First, investigating the objective function, we can follow that

$$\sum_{i=1}^{p-1} s_i\Big(\sum_{j=1}^{n} c_{ij}z_j + f_i t\Big) \geq \sum_{i=1}^{p-1}\Big(\sum_{j=1,c_{ij}>0}^{n} c_{ij}\underline{s}_i z_j + \sum_{j=1,c_{ij}<0}^{n} c_{ij}\overline{s}_i z_j\Big) + \sum_{i=1,f_i>0}^{p-1} f_i\underline{s}_i t + \sum_{i=1,f_i<0}^{p-1} f_i\overline{s}_i t. \tag{1}$$

Hence, we reformulate the objective function of the problem (EP$_3$) as

$$\Phi^R(s,t,z) = \sum_{i=1}^{p-1}\Big(\sum_{j=1,c_{ij}>0}^{n} c_{ij}\underline{s}_i z_j + \sum_{j=1,c_{ij}<0}^{n} c_{ij}\overline{s}_i z_j\Big) + \sum_{i=1,f_i>0}^{p-1} f_i\underline{s}_i t + \sum_{i=1,f_i<0}^{p-1} f_i\overline{s}_i t + \sum_{j=1}^{n} c_{pj}z_j + f_p t.$$

Next, investigating the constrained function, for any $i \in \{1,2,\cdots,p-1\}$, $s \in S \subseteq S^0$, define

$$\begin{aligned}
\Psi_i(s_i,t,z) &= s_i\Big(\sum_{j=1}^{n} d_{ij}z_j + g_i t\Big)\\
&= \sum_{j=1}^{n} d_{ij}s_i z_j + g_i s_i t,\\
\underline{G}_i &= \begin{cases} g_i\underline{s}_i, & \text{if } g_i > 0,\\ g_i\overline{s}_i, & \text{if } g_i < 0,\end{cases}\\
\overline{G}_i &= \begin{cases} g_i\underline{s}_i, & \text{if } g_i < 0,\\ g_i\overline{s}_i, & \text{if } g_i > 0,\end{cases}
\end{aligned}$$

and

$$\underline{\Psi}_i(\underline{s}_i,\overline{s}_i,t,z) = \sum_{j=1,d_{ij}>0}^{n} d_{ij}\underline{s}_i z_j + \sum_{j=1,d_{ij}<0}^{n} d_{ij}\overline{s}_i z_j + \underline{G}_i t, \tag{2}$$

$$\overline{\Psi}_i(\underline{s}_i,\overline{s}_i,t,z) = \sum_{j=1,d_{ij}>0}^{n} d_{ij}\overline{s}_i z_j + \sum_{j=1,d_{ij}<0}^{n} d_{ij}\underline{s}_i z_j + \overline{G}_i t. \tag{3}$$

Clearly, for any $i \in \{1,2,\dots,p-1\}$, we have that

$$\underline{\Psi}_i(\underline{s}_i,\overline{s}_i,t,z) \leq \Psi_i(s_i,t,z) \leq \overline{\Psi}_i(\underline{s}_i,\overline{s}_i,t,z)$$

always hold.

Integrating (1)–(3), for any $i \in \{1,2,\dots,p-1\}$, the affine relaxation programming problem of the problem (EP$_3$) over S is constructed as below:

$$(\text{ARP}): \begin{cases} \min\ \Phi^R(s,t,z)\\ \text{s.t. } \underline{\Psi}_i(\underline{s}_i,\overline{s}_i,t,z) \leq 1, \quad i = 1,\cdots,p-1,\\ \qquad \overline{\Psi}_i(\underline{s}_i,\overline{s}_i,t,z) \geq 1, \quad i = 1,\cdots,p-1,\\ (t,z) \in \chi,\ s \in S.\end{cases}$$

By the construction process of the ARP, it is obvious that the optimal solution of the ARP provides a valid lower bound for that of the EP$_3$ over S^0.

Theorem 1. For each $i \in \{1, 2, \ldots, p-1\}$, we have

$$\left|\Psi_i(s_i, t, z) - \underline{\Psi}_i(\underline{s}_i, \overline{s}_i, t, z)\right| \to 0 \text{ and } \left|\overline{\Psi}_i(\underline{s}_i, \overline{s}_i, t, z) - \Psi_i(s_i, t, z)\right| \to 0 \text{ as } |\overline{s}_i - \underline{s}_i| \to 0,$$

where "→" means "approaching".

Proof. Based on the previous definitions of the functions $\underline{\Psi}_i(\underline{s}_i, \overline{s}_i, t, z)$, $\Psi_i(s_i, t, z)$, and $\overline{\Psi}_i(\underline{s}_i, \overline{s}_i, t, z)$, for any $(t, z) \in \chi$, $s_i \in [\underline{s}_i, \overline{s}_i]$, we have that

$$\begin{aligned}\left|\Psi_i(s_i, t, z) - \underline{\Psi}_i(\underline{s}_i, \overline{s}_i, t, z)\right| &\le \left|\overline{\Psi}_i(\underline{s}_i, \overline{s}_i, t, z) - \underline{\Psi}_i(\underline{s}_i, \overline{s}_i, t, z)\right| \\ &= \Bigg| \sum_{j=1, d_{ij}>0}^{n} d_{ij}\overline{s}_i z_j + \sum_{j=1, d_{ij}<0}^{n} d_{ij}\underline{s}_i z_j + \overline{G}_i t \\ &\quad - \Big(\sum_{j=1, d_{ij}>0}^{n} d_{ij}\underline{s}_i z_j + \sum_{j=1, d_{ij}<0}^{n} d_{ij}\overline{s}_i z_j + \underline{G}_i t\Big)\Bigg| \\ &= \sum_{j=1}^{n} |d_{ij}||\overline{s}_i - \underline{s}_i| z_j + |g_i||\overline{s}_i - \underline{s}_i| t \\ &= |\overline{s}_i - \underline{s}_i|\Big(\sum_{j=1}^{n} |d_{ij}| z_j + |g_i| t\Big).\end{aligned}$$

Since $\sum\limits_{j=1}^{n} d_{ij}z_j + g_i t$ is a bounded function, we have

$$\left|\Psi_i(s_i, t, z) - \underline{\Psi}_i(\underline{s}_i, \overline{s}_i, t, z)\right| \to 0 \text{ as } |\overline{s}_i - \underline{s}_i| \to 0.$$

Similarly, we demonstrate

$$\left|\overline{\Psi}_i(\underline{s}_i, \overline{s}_i, t, z) - \Psi_i(s_i, t, z)\right| \to 0 \text{ as } |\overline{s}_i - \underline{s}_i| \to 0,$$

and the proof of the theorem is accomplished. □

From Theorem 1, it follows that the functions $\underline{\Psi}_i(\underline{s}_i, \overline{s}_i, t, z)$ and $\overline{\Psi}_i(\underline{s}_i, \overline{s}_i, t, z)$ can infinitely approximate the function $\Psi_i(s_i, t, z)$ as $|\overline{s}_i - \underline{s}_i| \to 0$, which guarantees the global convergence of the outcome space branch-and-bound algorithm.

3. Algorithm and its computational complexity

An outcome space branch-and-bound method is provided in this part to resolve the SARP based on the previous ARP. By resolving a series of ARPs over the initial rectangle S^0 or a partitioned sub-rectangle of S^0, the algorithm is able to acquire a global optimal solution. The simplest standard rectangle bisection rule, which is provided as follows, is designated in the proposed algorithm.

(1) Let

$$\overline{s}_{\upsilon}^{k}-\underline{s}_{\upsilon}^{k}=\max\left\{\overline{s}_{i}^{k}-\underline{s}_{i}^{k}, i=1,2,\dots,p-1\right\}$$

and

$$s_{\upsilon}^{k}=\frac{l_{\upsilon}^{k}+u_{\upsilon}^{k}}{2}.$$

(2) Let

$$\widehat{s}=\left(s_1^k,s_2^k,\dots,s_{\upsilon-1}^k,s_{\upsilon}^k,s_{\upsilon+1}^k,\dots,s_{p-1}^k\right)^{\top}.$$

The rectangle S^k is split by the point $\widehat{s}$ into the following two rectangles:

$$S^{k1}=\prod_{i=1}^{\upsilon-1}\left[\underline{s}_i^k,\overline{s}_i^k\right]\times\left[\underline{s}_{\upsilon}^k,s_{\upsilon}^k\right]\times\prod_{i=\upsilon+1}^{p-1}\left[\underline{s}_i^k,\overline{s}_i^k\right]$$

and

$$S^{k2}=\prod_{i=1}^{\upsilon-1}\left[\underline{s}_i^k,\overline{s}_i^k\right]\times\left[s_{\upsilon}^k,\overline{s}_{\upsilon}^k\right]\times\prod_{i=\upsilon+1}^{p-1}\left[\underline{s}_i^k,\overline{s}_i^k\right].$$

3.1. Algorithm outline

The steps of the outcome space branch-and-bound algorithm for solving the SARP are described below:

Algorithm 1. Outcome space branch-and-bound algorithm.

Step 0. Given the convergence error $\epsilon \geq 0$ and the initial outer space rectangle

$$S^0=\{s\in\mathbb{R}^{p-1}\mid \underline{s}_i^0\leq s_i\leq\overline{s}_i^0,\ i=1,2,\dots,p-1\}.$$

Solve the problem (ARP) over S^0. If the problem (ARP) over S^0 is not feasible, then the problem (SARP) is not feasible, and the proposed algorithm stops.

Otherwise, we can obtain an optimal solution $(\hat{s}^0,t^0,z^0)$ and the optimal value $LB(S^0)$ of the problem (ARP) over S^0. Let

$$s_i^0=\frac{1}{\sum\limits_{j=1}^{n}d_{ij}z_j^0+g_it^0},\quad i=1,2,\dots,p-1,$$

then (s^0,t^0,z^0) is a feasible solution of the problem (EP$_3$). Let

$$LB_0=LB(S^0)$$

and

$$UB_0=\Phi(s^0,t^0,z^0).$$

If

$$UB_0-LB_0\leq\epsilon,$$

then the algorithm stops executing, and (s^0,t^0,z^0) and $\frac{z^0}{t^0}$ are the global ϵ-optimal solutions of the problems (EP$_3$) and (SARP), respectively.

Otherwise, denote by the set of all active nodes $\Theta_0 = \{S^0\}$, denote by the set of the initial feasible point

$$F = \{(s^0, t^0, z^0)\},$$

let $k = 0$, and proceed to Step 1.

Step 1. Set

$$UB_k = UB_{k-1}.$$

Using the former branching rule, subdivide S^k into two sub-rectangles S^{k1} and S^{k2}. Let

$$Q = \{S^{k1}, S^{k2}\}.$$

Step 2. For each

$$S^{k\tau}, \tau = 1, 2,$$

solve the problem (ARP) over $S^{k\tau}$. If the problem (ARP) over $S^{k,\tau}$ is not feasible, then delete the rectangle $S^{k\tau}$. Otherwise, we get the optimal solution $(\hat{s}(S^{k\tau}), t(S^{k\tau}), z(S^{k\tau}))$ and the optimal value $LB(S^{k\tau})$ of the problem (ARP) over $S^{k\tau}$.

If

$$UB_k \leq LB(S^{k\tau}),$$

then let

$$Q = Q \setminus \{S^{k\tau}\}.$$

Otherwise, let

$$s_i(S^{k\tau}) = \frac{1}{\sum\limits_{j=1}^{n} d_{ij} z_j(S^{k\tau}) + g_i t(S^{k\tau})}, \quad i = 1, 2, \ldots, p-1,$$

update the feasible point set by

$$F = F \bigcup \{(s(\Omega^{k\tau}), t(S^{k\tau}), z(S^{k,\tau}))\},$$

let

$$UB_k = \min\{UB_k, \Phi(s(S^{k\tau}), t(S^{k\tau}), z(S^{k,\tau}))\},$$

and denote by the currently best feasible solution (s^k, t^k, z^k), which corresponds to UB_k.

Step 3. Set

$$\Theta_k = (\Theta_k \setminus S^k) \bigcup Q,$$

update

$$LB_k = \min\{LB(S) \mid S \in \Theta_k\},$$

and let S^k be the sub-rectangle which satisfies

$$LB_k = LB(S^k).$$

Step 4. If

$$UB_k - LB_k \leq \epsilon,$$

then the proposed algorithm stops, and $\frac{z^k}{t^k}$ and (s^k, t^k, z^k) are the ϵ-global optimal solutions of the problems (SARP) and (EP$_3$), respectively.

Otherwise, set $k = k + 1$, and go back to Step 1.

3.2. Convergence analysis

In this subsection, the global convergence of the proposed algorithm is discussed as follows:

Theorem 2. If Algorithm 1 ceases finite iterations, then the proposed algorithm will produce a globally optimal solution to the SARP. Otherwise, Algorithm 1 fails to terminate within a finite number of iterations, and it will produce an infinite sequence $\{x^k\}$ such that any accumulation point will be a global optimal solution of the SARP.

Proof. If Algorithm 1 ceases within a finite number of iterations, hypothesize that it is ceased at the k_{th} iteration, denoted LB_k as the current best lower bound, and denoted UB_k as the current best upper bound, for any ϵ, and we have that

$$UB_k - LB_k < \epsilon. \tag{4}$$

Assume that (s^k, t^k, z^k) is a feasible solution of the problem (EP$_3$) with that

$$s_i^k = \frac{1}{\sum\limits_{j=1}^{n} d_{ij} z_j^k + g_i t^k}.$$

Letting $\upsilon(\text{EP}_3)$ be the global optimal value of the problem (EP$_3$), we can get

$$\Phi(s^k, t^k, z^k) - LB_k = UB_k - LB_k \leq \epsilon \tag{5}$$

and

$$LB_k \leq \upsilon(\text{EP}_3). \tag{6}$$

However, (s^k, t^k, z^k) is the feasible solution of the problem (EP$_3$), and we can follow that

$$\Phi(s^k, t^k, z^k) \geq \upsilon(\text{EP}_3). \tag{7}$$

By the above inequalities (4)–(7), we can get

$$\upsilon(\text{EP}_3) \leq \Phi(s^k, t^k, z^k) \leq LB^k + \epsilon \leq \upsilon(\text{EP}_3) + \epsilon.$$

Therefore, (s^k, t^k, z^k) is an ϵ-global optimum solution of the problem (EP$_3$), and

$$x^k = \frac{z^k}{t^k}$$

is an ϵ-global optimum solution of the SARP.

If Algorithm 1 fails to terminate within a finite number of iterations, then it will produce an infinite feasible solution sequence $\{x^k\}$ of the SARP and an infinite feasible solution sequence (s^k, t^k, z^k) of the problem (EP$_3$) with

$$s_i^k = \frac{1}{\sum\limits_{j=1}^{n} d_{ij} z_j^k + g_i t^k}.$$

Letting (t^*, z^*) be an accumulation point of the sequence $\{(t^k, z^k)\}$, we can get

$$\lim_{k\to\infty}(t^k, z^k) = (t^*, z^*).$$

By the continuity of the function

$$s_i = \frac{1}{\sum\limits_{j=1}^{n} d_{ij}z_j + g_i t}, \quad i \in \{1, 2, \cdots, p-1\},$$

we can get that

$$\frac{1}{\sum\limits_{j=1}^{n} d_{ij}z_j^* + g_i t^*} = \lim_{k\to\infty} \frac{1}{\sum\limits_{j=1}^{n} d_{ij}z_j^k + g_i t^k} = \lim_{k\to\infty} s_i^k = s_i^*.$$

Thus, (s^*, t^*, z^*) is a feasible solution of the problem (EP$_3$).

Since $\left\{LB_k\right\}$ is a monotonic non-decreasing bounded sequence, we get that

$$LB_k = \Phi^R(s^k, t^k, z^k)$$

and

$$LB_k \le \upsilon(\text{EP}_3),$$

and also since $\{UB_k\}$ is a non-increasing and bounded sequence, we get that

$$UB_k = \Phi(s^k, t^k, z^k)$$

and

$$UB_k \ge \upsilon(\text{EP}_3).$$

So, we have

$$LB_k = \Phi^R(s^k, t^k, z^k) \le \upsilon(\text{EP}_3) \le \Phi(s^k, t^k, z^k) = UB_k.$$

Hence, from the termination condition of the algorithm, taking the limit on both side of the above inequalities, we get

$$\lim_{k\to\infty} LB_k = \upsilon(\text{EP}_3) = \lim_{k\to\infty} \Phi(s^k, t^k, z^k) = \Phi(s^*, t^*, z^*) = \lim_{k\to\infty} UB_k.$$

Thus, the accumulation point (s^*, t^*, z^*) of the sequence (s^k, t^k, z^k) is a global optimal solution of the problem (EP$_3$). At the same time, the accumulation point x^* of the sequence $\{x^k\}$ with

$$x^k = \frac{z^k}{t^k}$$

is a global optimal solution of the SARP, and the proof is complete. □

Remark 2. The algorithm proposed in this paper is a rectangular branch-and-bound global optimization algorithm. As is well-known, based on the convergence theory of branch-and-bound global optimization algorithms [33], the exhaustiveness of the branching process and the approximation of upper and lower bounds indicate that the proposed rectangular branch-and-bound global optimization algorithm must be convergent.

3.3. *Computational complexity of the algorithm*

Further, to examine the maximum number of iterations of Algorithm 1, we derive the computational complexity of Algorithm 1. First, we define the size Υ of the rectangle S as

$$\Upsilon(S) = \max\{\overline{s}_i - \underline{s}_i \mid i = 1, 2, \cdots, p-1\}.$$

Theorem 3. Given a termination error $\epsilon > 0$, if there exists a rectangle S formed by the proposed algorithm at the k_{th} iteration satisfying

$$\Upsilon(S) \leq \frac{\epsilon}{(p-1)\omega},$$

then we have that

$$UB_k - LB(S) \leq \epsilon,$$

where $LB(S)$ is the optimal value of the ARP over S, and UB_k is the currently known best upper bound of the optimum value of the problem (EP$_3$).

Proof. Without losing generality, suppose that (t^k, z^k) is the optimal solution of the ARP over S, and let

$$s_i^k = \frac{1}{\sum\limits_{j=1}^{n} d_{ij} z_j^k + g_i t^k}, \quad i = 1, 2, \ldots, p-1.$$

Then (s^k, t^k, z^k) is a feasible solution to the problem (EP$_3$). By the method of updating the upper bound and lower bound, and the construction process of the ARP, we have that

$$\Phi(s^k, t^k, z^k) \geq UB^k \geq LB(S) = \Phi^R(s^k, y^k, z^k).$$

By the definition of the size $\Upsilon(S)$ of the rectangle S and

$$\Upsilon(S) \leq \frac{\epsilon}{(p-1)\omega},$$

it follows that

$$\begin{aligned}
UB_k - LB(S) &\leq \Phi(s^k, t^k, z^k) - \Phi^R(s^k, t^k, z^k) \\
&= \Big| \sum_{i=1}^{p-1} s_i^k \Big(\sum_{j=1}^{n} c_{ij} z_j^k + f_i t^k \Big) + \sum_{j=1}^{n} c_{pj} z_j^k + f_p t^k \\
&\quad - \Big[\sum_{i=1}^{p-1} \Big(\sum_{j=1, c_{ij}>0}^{n} c_{ij} \underline{s}_i z_j^k + \sum_{j=1, c_{ij}<0}^{n} c_{ij} \overline{s}_i^k z_j^k + \sum_{j=1, f_i>0}^{n} f_i \underline{s}_i t^k + \sum_{j=1, f_i<0}^{n} f_i \overline{s}_i t^k + \sum_{j=1}^{n} c_{pj} z_j^k + f_p t^k \Big) \Big] \Big| \\
&= \Big| \sum_{i=1}^{p-1} (s_i - \underline{s}_i) \sum_{j=1, c_{ij}>0}^{n} c_{ij} z_j^k + \sum_{i=1}^{p-1} (s_i - \overline{s}_i) \sum_{j=1, c_{ij}<0}^{n} c_{ij} z_j^k + \sum_{i=1, f_i>0}^{p-1} (s_i - \underline{s}_i) f_i t^k + \sum_{i=1, f_i<0}^{p-1} (s_i - \overline{s}_i) f_i t^k \Big| \\
&\leq \sum_{i=1}^{p-1} (\overline{s}_i - \underline{s}_i) \times \Big(\sum_{j=1}^{n} |c_{ij}| z_j^k + \sum_{i=1}^{n} |f_i| t^k \Big) \\
&\leq (p-1)\omega \Upsilon(S) \\
&\leq \epsilon.
\end{aligned}$$

The proof of Theorem 3 is complete. □

Then we can examine the maximum iterations of Algorithm 1, see the following Theorem 4:

Theorem 4. Given the convergent tolerance $\epsilon \in (0,1)$, Algorithm 1 can acquire a global optimal solution of the SARP at most

$$(p-1)\cdot\left\lceil \log_2 \frac{(p-1)\omega\Upsilon(S^0)}{\epsilon}\right\rceil$$

iterations.

Proof. In general, we assume that the sub-rectangle S is selected for partitioning in Algorithm 1 at each iteration. After $k\cdot(p-1)$ iterations, we can follow that

$$\Upsilon(S) \le \frac{1}{2^k}\Upsilon(S^0).$$

Based on the former proof of Theorem 3, if

$$\frac{1}{2^k}\Upsilon(S^0) \le \frac{\epsilon}{(p-1)\omega},$$

i.e.,

$$k \ge \log_2 \frac{(p-1)\omega\Upsilon(S^0)}{\epsilon},$$

we can follow that

$$UB_k - LB(S) \le \epsilon.$$

Therefore, after at most

$$(p-1)\cdot\left\lceil \log_2 \frac{(p-1)\omega\Upsilon(S^0)}{\epsilon}\right\rceil$$

iterations, we can get

$$\begin{aligned}0 &\le \Phi(s^k,t^k,z^k) - \Phi(s^*,t^*,z^*)\\ &\le \Phi(s^k,t^k,z^k) - LB(S)\\ &= UB_k - LB(S)\\ &\le \epsilon,\end{aligned}$$

where (s^*,t^*,z^*) is the optimal solution of the problem (EP$_3$), and (s^k,t^k,z^k) is the best currently know feasible solution for the problem (EP$_3$). This implies that (s^k,t^k,z^k) is an ϵ-global optimal solution of the problem (EP$_3$) when Algorithm 1 terminates. At the same time, we can follow that $\frac{z^k}{t^k}$ is an ϵ-global optimal solution of the SARP, and we complete the proof. □

Remark 3. From the above complexity analysis of the algorithm in Theorem 4, letting

$$\Gamma = (p-1)\cdot\left\lceil \log_2 \frac{(p-1)\omega\Upsilon(S^0)}{\epsilon}\right\rceil,$$

the running time of Algorithm 1 is bounded by

$$2(\Gamma-1)\times T(m+2p+1,n+1)$$

for finding an ϵ-global optimal solution of the SARP, where $T(m+2p+1,n+1)$ denotes the time taken to solve a linear programming problem with $n+1$ variables and $m+2p+1$ linear constraints.

4. Numerical experiments

In this section, we numerically compare Algorithm 1 with some known extant branch-and-bound algorithms. All tests were implemented in MATLAB R2018a and run on a microcomputer with Intel(R) Core(TM) i5-10400 CPU @ 2.90GHz processor and 8 GB RAM. For all test problems, we provide their computational results. The maximum time limit was set to $10000s$ for all algorithms.

First of all, some small-size certainty examples (see the Appendix Examples 1–12) were tested with Algorithm 1 for comparison with the known extant algorithms [14, 21, 34–40], and a numerical comparison between some existing algorithms and Algorithm 1 on Examples 1–12 were reported in Table 1 with the given convergence tolerance, where some notations have been used for column headers: Opt. val.: global optimal value; Iter.: number of iterations of the algorithm; Time: the CPU execution time of the algorithm in seconds.

Table 1. Numerical comparison between some existing algorithms and Algorithm 1 on test Examples 1–12.

No.	Algorithms	Opt. val.	Optimal solution	Iter.	Time	ϵ
1	Algorithm 1	-4.83976	(0.1117, 2.3603)	16	0.254	10^{-2}
	Benson [34]	-4.84151	(0.1000, 2.3750)	4	0.190	10^{-2}
	Jiao and Liu [39]	-4.84151	(0.1000, 2.3750)	200	4.257	10^{-2}
2	Algorithm 1	-2.47143	(1.0000, 0.0000, 0.0000)	11	0.192	10^{-2}
	Shen et al. [35]	-2.47143	(1.0000, 0.0000, 0.0000)	2	0.015	10^{-2}
	Jiao and Liu [39]	-2.47124	(1.0001, 0.0000, 0.0001)	54	1.135	10^{-2}
3	Algorithm 1	-1.90000	(0.0000, 3.3333, 0.0000)	424	6.907	10^{-6}
	Shen and Wang [36]	-1.90000	(0.0000, 3.3333, 0.0000)	8	0.926	10^{-6}
4	Algorithm 1	1.62318	(0.0000, 0.2841)	39	0.566	10^{-2}
	Jiao and Liu [39]	1.62319	(0.0000, 0.2861)	93	2.485	10^{-2}
5	Algorithm 1	2.86190	(5.0000, 0.0000, 0.0000)	199	2.776	10^{-3}
	Jiao and Liu [39]	2.86241	(4.8302, 0.0000, 0.0666)	4008	128.0	10^{-3}
	Shen and Lu [37]	2.86191	(5.0000, 0.0000, 0.0000)	16	0.125	10^{-3}
	Gao and Jin [38]	2.86190	(5.0000, 0.0000, 0.0000)	12	28.29	10^{-3}
6	Algorithm 1	-4.09070	(1.1111, 0.0000, 0.0000)	28	0.359	10^{-2}
	Jiao and Liu [39]	-4.09062	(1.1106, 0.0000, 0.0015)	619	16.62	10^{-2}
	Shen and Lu [37]	-4.08741	(1.0715, 0.0000, 0.0000)	17	3.251	10^{-2}
7	Algorithm 1	3.71092	(0.0000, 1.6667, 0.0000)	89	1.124	10^{-4}
	Jiao and Liu [39]	3.71093	(0.0000, 1.6667, 0.0000)	2747	94.64	10^{-4}
	Gao and Jin [38]	3.7087	(0.0000, 1.6667, 0.0000)	5	4.190	10^{-4}
8	Algorithm 1	-3.00225	(0.0000, 2.8455, 0.0000)	36	0.566	10^{-2}
	Jiao and Liu [39]	-3.00292	(0.0000, 3.3333, 0.0000)	1072	31.746	10^{-2}
9	Algorithm 1	4.91267	(1.5015, 1.5024)	30	0.422	10^{-3}
	Shen and Lu [37]	4.91259	(1.5000, 1.5000)	56	1.087	10^{-3}
10	Algorithm 1	-4.09070	(1.1111, 0.0000, 0.0000)	33	0.492	10^{-2}
	Jiao et al. [40]	-4.09070	(1.1111, 0.0000, 0.0000)	2	0.008	10^{-6}
	Jiao and Liu [39]	-4.09065	(1.1109, 0.0000, 0.0005)	977	32.41	10^{-6}
11	Algorithm 1	3.29167	(3.0000, 4.0000)	138	1.902	10^{-6}
	Shen and Wang [36]	3.29167	(3.0000, 4.0000)	9	0.489	10^{-6}
12	Algorithm 1	4.42857	(5.0000, 0.0000, 0.0000)	67	0.931	10^{-4}
	Jiao and Liu [39]	4.42794	(4.9930, 0.0000, 0.0000)	128	4.213	10^{-4}
	Shi [21]	4.42857	(5.0000, 0.0000, 0.0000)	58	2.968	10^{-4}

From the numerical results in Table 1, for Examples 2, 5, 6, 10 and 12, we can follow that Algorithm 1 can obtain the better global optimal solutions and optimal values than the existing algorithms of Jiao and Liu [39] and Shen and Lu [37]. Algorithm 1 performs better than the method of Jiao and Liu [39] for finding the optimal solution in less time and fewer iterations. Therefore, in terms of test Examples 1–12, their experimental results verify that Algorithm 1 is valid and feasible.

Next, we chose two large-scale test problems generated randomly to verify the proposed algorithm further, see Problems 1 and 2 for details. With the given approximation error $\epsilon = 10^{-2}$, we first tested Problem 1 with large-size variables, numerical comparisons among Algorithm 1, the algorithm of Jiao and Liu [39], and the algorithm of Li et al. [41], which are reported in Table 2.

Table 2. Numerical comparisons among Algorithm 1, the algorithm of Jiao and Liu [39], and the algorithm of Li et al. [41] on Problem 1.

(p, m, n)	Algorithms	Iter.			Time		
		min.	ave.	max.	min.	ave.	max.
(2,100,1000)	Jiao and Liu [39]	34	111.5	342	6.78	22.82	71.53
	Li et al. [41]	9	21.8	30	2.21	4.03	5.46
	Algorithm 1	**6**	**14.8**	**23**	**1.56**	**2.73**	**3.41**
(2,100,3000)	Jiao and Liu [39]	55	99.8	159	82.16	147.45	235.12
	Li et al. [41]	9	23.4	27	11.26	18.12	24.12
	Algorithm 1	**6**	**12.4**	**19**	**11.23**	**17.89**	**23.45**
(2,100,5000)	Jiao and Liu [39]	41	83.9	152	171.34	339.68	618.89
	Li et al. [41]	12	18.1	30	56.56	75.24	96.18
	Algorithm 1	**8**	**12.3**	**25**	**38.51**	**50.45**	**68.32**
(2,100,7000)	Jiao and Liu [39]	49	78.7	124	392.12	634.51	987.42
	Li et al. [41]	10	16.9	27	99.84	126.53	185.62
	Algorithm 1	**7**	**11.7**	**21**	**68.87**	**89.45**	**129.23**
(2,100,10000)	Jiao and Liu [39]	13	73.1	129	230.12	1188.43	2145.25
	Li et al. [41]	12	18.5	25	241.56	289.25	324.56
	Algorithm 1	**8**	**12.8**	**17**	**166.56**	**204.43**	**245.52**
(3,100,1000)	Jiao and Liu [39]	215	814.4	1756	43.12	175.68	398.53
	Li et al. [41]	75	309.4	628	10.89	40.53	72.47
	Algorithm 1	**53**	**238.2**	**465**	**7.45**	**28.54**	**48.98**
(3,100,3000)	Jiao and Liu [39]	*	*	*	*	*	*
	Li et al. [41]	92	257.8	378	123.56	254.58	367.92
	Algorithm 1	**63**	**198.8**	**296**	**81.47**	**187.45**	**284.32**
(3,100,5000)	Jiao and Liu [39]	*	*	*	*	*	*
	Li et al. [41]	75	273.8	512	278.56	625.48	985.26
	Algorithm 1	**54**	**191.7**	**348**	**208.45**	**452.78**	**712.45**
(3,100,7000)	Jiao and Liu [39]	*	*	*	*	*	*
	Li et al. [41]	89	243.5	325	645.38	1201.48	1572.48
	Algorithm 1	**61**	**163.4**	**242**	**439.43**	**805.39**	**1168.71**
(3,100,10000)	Jiao and Liu [39]	*	*	*	*	*	*
	Li et al. [41]	130	249.8	426	1580.22	2678.56	4582.58
	Algorithm 1	**89**	**165.2**	**287**	**1087.24**	**1786.13**	**3090.56**
(4,100,1000)	Jiao and Liu [39]	*	*	*	*	*	*
	Li et al. [41]	485	3436.6	9856	46.53	323.23	1169.14
	Algorithm 1	**317**	**2332.8**	**7279**	**43.62**	**278.85**	**985.93**

In Tables 2 and 3, Avg.Iter denotes the average number of iterations of the algorithm, Avg.Time denotes the average execution CPU time of the algorithm in seconds, "$*$" denotes the situation that the proposed algorithm failed to terminate in $10000s$ for some of the arbitrary 50 independently generated test examples. For random Problems 1 and 2, we solved 50 independently generated test instances and recorded their average results among these 50 tests, and we highlighted in bold the winner of average results in numerical comparisons.

Table 3. Numerical comparisons between Algorithm 1 and the algorithm of Li et al. [41] on Problem 2.

(p, m, n)	Algorithms	Iter.			Time		
		min.	ave.	max.	min.	ave.	max.
(10,100,300)	Li et al. [41]	9	13.6	19	5.28	8.87	12.7
	Algorithm 1	**7**	**9.8**	**15**	**4.12**	**6.45**	**10.2**
(10,100,400)	Li et al. [41]	10	16	25	6.90	12.65	20.66
	Algorithm 1	**8**	**12.8**	**19**	**5.61**	**9.68**	**16.92**
(10,100,500)	Li et al. [41]	10	17.4	30	8.07	15.89	26.52
	Algorithm 1	**9**	**14.2**	**26**	**6.45**	**12.75**	**21.45**
(15,100,400)	Li et al. [41]	50	121.6	201	46.78	118.75	201.66
	Algorithm 1	**38**	**95.6**	**189**	**39.98**	**95.68**	**179.85**
(15,100,500)	Li et al. [41]	49	118.1	258	54.57	137.92	303.49
	Algorithm 1	**41**	**98.7**	**202**	**41.38**	**99.56**	**201.24**
(20,100,300)	Li et al. [41]	157	321.2	861	126.19	255.46	694.20
	Algorithm 1	**118**	**278.6**	**598**	**89.16**	**202.46**	**587.45**
(20,100,400)	Li et al. [41]	99	399.9	1134	99.06	425.77	1199.2
	Algorithm 1	**87**	**312.7**	**985**	**87.45**	**364.10**	**950.26**

Problem 1. (Li et al. [41])

$$\begin{cases} \min \ \sum_{i=1}^{p} \dfrac{\bar{c}_i^\top x + \bar{f}_i}{\bar{d}_i^\top x + \bar{g}_i}, \\ \text{s.t.} \quad \bar{A}x \leq \bar{b}, \\ \qquad\ x \geq 0, \end{cases}$$

where $\bar{c}_i \in \mathbb{R}^n$, $\bar{d}_i \in \mathbb{R}^n$, $\bar{A} \in \mathbb{R}^{m\times n}$, $\bar{b} \in \mathbb{R}^m$, $\bar{f}_i \in \mathbb{R}, \bar{g}_i \in \mathbb{R}, i = 1, 2, \ldots, p$; each element of $\bar{c}_i$, $\bar{d}_i$, and $\bar{A}$ is randomly generated from the interval $[0, 10]$; each element of $\bar{b}$ is equal to 10, and each element of $\bar{f}_i$ and $\bar{g}_i$ is randomly generated from the interval $[0, 1]$.

Problem 2. (Li et al. [41])

$$\begin{cases} \min \ \sum_{i=1}^{p} \dfrac{\sum_{j=1}^{n} \tilde{\gamma}_{ij} x_j + \tilde{\xi}_i}{\sum_{j=1}^{n} \tilde{\delta}_{ij} x_j + \tilde{\eta}_i}, \\ \text{s.t.} \quad \tilde{A}x \leq \tilde{b}, \quad x \geq 0, \end{cases}$$

where $\tilde{\gamma}_{ij}, \tilde{\xi}_i, \tilde{\delta}_{ij}, \tilde{\eta}_i \in \mathbb{R}$, $i = 1, 2, \ldots, p$, $j = 1, 2, \ldots, n$; $\tilde{A} \in \mathbb{R}^{m\times n}$, $\tilde{b} \in \mathbb{R}^m$; all $\tilde{\gamma}_{ij}$ and $\tilde{\delta}_{ij}$ are randomly generated from $[-0.1, 0.1]$; all elements of $\tilde{A}$ are randomly generated from $[0.01, 1]$; all elements of $\tilde{b}$

are equal to 10; and all $\tilde{\xi}_i$ and $\tilde{\eta}_i$ satisfy

$$\sum_{j=1}^{n} \tilde{\gamma}_{ij} x_j + \tilde{\xi}_i > 0$$

and

$$\sum_{j=1}^{n} \tilde{\delta}_{ij} x_j + \tilde{\eta}_i > 0.$$

From the results in Table 2, for Problem 1 with the large-size number of variables, we first get the observation that the algorithm proposed in Jiao and Liu [39] is more time-consuming than Algorithm 1. Especially, when $p = 3, m = 100, n = 3000$; $p = 3, m = 100, n = 5000$; $p = 3, m = 100, n = 7000$; $p = 3, m = 100, n = 10000$; $p = 4, m = 100, n = 1000$; the algorithm of Jiao & Liu [39] failed to solve all 50 independently generated instances in $10000s$, but Algorithm 1 can obtain the global optimal solution of test Problem 1 with higher computational efficiency. Second, the computational efficiency of Algorithm 1 is superior to the algorithm of Li et al. [41] in all cases.

From the numerical comparisons for Problem 2 in Table 3, the computational efficiency of Algorithm 1 is superior to the algorithm of Li et al. [41] in all cases.

From the numerical comparisons in Tables 1–3, we can get that Algorithm 1 can globally solve the sum of affine ratios problem to obtain their global optimal solutions and optimal values with higher computational efficiency.

5. Conclusions

This paper studies the sum of affine ratios problem and presents an outcome space branch-and-bound algorithm. In this algorithm, we proposed a novel linearization technique for constructing the affine relaxation problem of the equivalent problem. Moreover, the computational complexity of the algorithm is analyzed, and the maximum number of iterations of the algorithm is derived. Algorithm 1 can find an ϵ-global optimal solution in at most

$$(p-1) \cdot \left\lceil \log_2 \frac{(p-1)\omega\Upsilon(S^0)}{\epsilon} \right\rceil$$

iterations. Numerical comparisons show the effectiveness and superiority of Algorithm 1. Future work will extend Algorithm 1 to solve the sum of nonlinear ratios problem.

Author contributions

Yan Shi: formal analysis, investigation, resources, methodology, writing-original draft, validation, and data curation; Qunzhen Zheng: formal analysis, invesigation, writing-review & editing, software, data curation, conceptualization, supervision, project administration; 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 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 in Henan Province (202102210147, 192102210114).

Conflict of interest

The authors declare no conflicts of interest.

References

1. J. Majhi, R. Janardan, J. Schwerdt, M. Smid, P. Gupta, Minimizing support structures and trapped area in two-dimensional layered manufacturing, *Comput. Geom.*, **12** (1999), 241–267. https://doi.org/10.1016/S0925-7721(99)00003-6

2. J. Majhi, R. Janardan, M. Smid, P. Gupta, On some geometric optimization problems in layered manufacturing, *Comput. Geom.*, **12** (1999), 219–239. https://doi.org/10.1016/S0925-7721(99)00002-4

3. 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

4. C. S. Colantoni, R. P. Manes, A. Whinston, Programming, profit rates and pricing decisions, *Account. Rev.*, **44** (1969), 467–481.

5. H. Konno, H. Watanabe, Bond portfolio optimization problems and their applications to index tracking: a partial optimization approach, *J. Oper. Res. Soc. Jpn.*, **39** (1996), 295–306. https://doi.org/10.15807/jorsj.39.295

6. I. M. Stancu-Minasian, *Fractional programming theory, methods and applications*, Kluwer Academic Publishers, 1997. https://doi.org/10.1007/978-94-009-0035-6

7. E. B. Bajalinov, *Linear-fractional programming theory, methods, applications and software*, Springer Science and Business Media, 2003. https://doi.org/10.1007/978-1-4419-9174-4

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

9. B. Sawik, Downside risk approach for multi-objective portfolio optimization, In: D. Klatte, L. Hans-Jakob, K. Schmedders, *Operations research proceedings 2011*, Springer Science and Business Media, 2012. https://doi.org/10.1007/978-3-642-29210-1_31

10. V. Milenkovic, K. Daniels, Z. Li, *Placement and compaction of nonconvex polygons for clothing manufacture*, Memorial University of Newfoundland, 1992.

11. E. M. Arkin, Y. J. Chiang, M. Held, J. S. B. Mitchell, V. Sacristan, S. S. Skiena, et al., On minimum-area hulls, *Algorithmica*, **21** (1998), 119–136. https://doi.org/10.1007/PL00009204

12. A. Charnes, W. W. Cooper, Programming with linear fractional functionals, *Nav. Res. Logist. Q.*, **9** (1962), 181–186. https://doi.org/10.1002/nav.3800090303

13. H. Konno, Y. Yajima, T. Matsui, Parametric simplex algorithms for solving a special class of nonconvex minimization problems, *J. Global Optim.*, **1** (1991), 65–81. https://doi.org/10.1007/BF00120666

14. N. T. H. 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

15. Y. E. Nesterov, A. S. Nemirovskii, An interior-point method for generalized linear-fractional programming, *Math. Program.*, **69** (1995), 177–204. https://doi.org/10.1007/BF01585557

16. R. W. Freund, F. Jarre, Solving the sum-of-ratios problem by an interior-point method, *J. Global Optim.*, **19** (2001), 83–102. https://doi.org/10.1023/A:1008316327038

17. E. F. James, W. P. Susan, Image space analysis of generalized fractional programs, *J. Global Optim.*, **4** (1994), 63–88. https://doi.org/10.1007/BF01096535

18. T. Kuno, A revision of the trapezoidal branch-and-bound algorithm for linear sum-of-ratios problem, *J. Global Optim.*, **33** (2005), 215–234. https://doi.org/10.1007/s10898-004-1952-z

19. H. Jiao, A branch and bound algorithm for globally solving a class of nonconvex programming problems, *Nonlinear Anal.*, **70** (2009), 1113–1123. https://doi.org/10.1016/j.na.2008.02.005

20. K. S. Bazaraa, H. D. Sherali, C. M. Shetty, *Nonlinear programming: theory and algorithms*, John Wiley & Sons, Inc., 2006. https://doi.org/10.1002/0471787779

21. Y. Shi, *Global optimization for sum of ratios problems*, MA thesis, Henan Normal University, 2011.

22. H. Jiao, J. Ma, An efficient algorithm and complexity result for solving the sum of general affine ratios problem, *Chaos Solitons Fract.*, **164** (2022), 112701. https://doi.org/10.1016/j.chaos.2022.112701

23. 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

24. H. Jiao, J. Ma, P. Shen, Y. Qiu, Effective algorithm and computational complexity for solving sum of linear ratios problem, *J. Ind. Manage. Optim.*, **19** (2023), 4410–4427. https://doi.org/10.3934/jimo.2022135

25. H. Jiao, Y. Shang, Image space branch-reduction-bound algorithm for globally solving the sum of affine ratios problem, *J. Comput. Math.*, 2024. https://doi.org/10.4208/jcm.2203-m2021-0085

26. Y. Pei, D. Zhu, Global optimization method for maximizing the sum of difference of convex functions ratios over nonconvex region, *J. Appl. Math. Comput.*, **41** (2013), 153–169. https://doi.org/10.1007/s12190-012-0602-8

27. T. Kuno, A revision of the trapezoidal branch-and-bound algorithm for linear sum-of-ratios problems, *J. Global Optim.*, **33** (2005), 215–234. https://doi.org/10.1007/s10898-004-1952-z

28. P. P. Shen, W. M. Li, Y. C. Liang, Branch-reduction-bound algorithm for linear sum-of-ratios fractional programs, *Pac. J. Optim.*, **11** (2015), 79–99.

29. H. Jiao, S. Liu, Range division and compression algorithm for quadratically constrained sum of quadratic ratios, *Comput. Appl. Math.*, **36** (2017), 225–247. https://doi.org/10.1007/s40314-015-0224-5

30. H. Jiao, B. Li, W. Yang, A criterion-space branch-reduction-bound algorithm for solving generalized multiplicative problems, *J. Global Optim.*, **89** (2024), 597–632. https://doi.org/10.1007/s10898-023-01358-w

31. 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

32. A. Charnes, W. W. Cooper, Programming with linear fractional functionals, *Nav. Res. Log. Q.*, **9** (1962), 181–186. https://doi.org/10.1002/nav.3800150308

33. R. Horst, H. Tuy, *Global optimization, deterministic approaches*, Springer-Verlag, 1990. https://doi.org/10.1007/978-3-662-02598-7

34. 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

35. P. Shen, B. Huang, L. Wang, Range division and linearization algorithm for a class of linear ratios optimization problems, *J. Comput. Appl. Math.*, **350** (2019), 324–342. https://doi.org/10.1016/j.cam.2018.10.038

36. 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

37. P. P. Shen, T. Lu, Regional division and reduction algorithm for minimizing the sum of linear fractional functions, *J. Inequal. Appl.*, **2018** (2018), 63. https://doi.org/10.1186/s13660-018-1651-9

38. Y. Gao, S. Jin, A global optimization algorithm for sum of linear ratios problem, *J. Appl. Math.*, 2013, 276245. https://doi.org/10.1155/2013/276245

39. H. W. Jiao, S. Y. 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

40. H. Jiao, S. Liu, J. Yin, Y. Zhao, Outcome space range reduction method for global optimization of sum of affine ratios problem, *Open Math.*, **14** (2016), 736–746. https://doi.org/10.1515/math-2016-0058

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

Appendix

Test Examples 1–12 are given as follows:

Example 1. (Benson [34])

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

Example 2. (Phuong and Tuy [14] and Shen et al. [35])

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

Example 3. (Shen et al. [35], Shen and Wang [36])

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

Example 4. (Shen et al. [35])

$$\begin{cases} \min & \frac{-x_1+2x_2+2}{3x_1-4x_2+5} + \frac{4x_1-3x_2+4}{-2x_1+x_2+3}, \\ \text{s.t.} & x_1 + x_2 \leq 1.5, \\ & x_1 - x_2 \leq 0, \\ & 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1. \end{cases}$$

Example 5. (Shen and Lu [36], Gao and Jin [38])

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

Example 6. (Shen and Lu [37])

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

Example 7. (Shen and Lu [37], Gao and Jin [38])

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

Example 8. (Shen and Lu [37])

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

Example 9. (Shen and Lu [37], Gao and Jin [38])

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

Example 10. (Jiao and Liu [39], Jiao et al. [40], Shen Wang [36])

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

Example 11. (Shen and Wang [36], Shi [21])

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

Example 12. (Shi [21])

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