An Outcome-space Branch-and-bound Algorithm for Sum-of-linear-ratios Problem

Hanbing Mei, Xianfeng Ding, Xiaolin Yi, Pengfei Wen, Yiyu Qin, Qianmei Luo

***Abstract*—In this research paper, a novel branch-and-bound algorithm within the outcome-space is introduced for the generalized sum-of-linear-ratios problem. Initially, an equivalent problem of the original problem is derived by applying constraints to the numerator and denominator of the objective function, leveraging the reciprocal characteristics of logarithmic and exponential functions. Subsequently, the linear relaxation programming problem for this equivalent problem is constructed through two distinct approaches, utilizing the geometric attributes of these functions. By integrating the branch-and-bound framework with the linear relaxation problem, a comprehensive branch-and-bound algorithm is proposed for globally solving the sum-of-linear-ratios problem, and its convergence is established. Finally, numerical experiments are conducted to confirm the viability and efficacy of the proposed algorithm.**

***Index Terms*—Sum-of-Linear-Ratios problem, Linear Relaxation, Branch and Bound, Outcome-Space.**

I. Introduction

THE primary focus of this paper is on addressing the sum-of-linear-ratios problem as outlined below:

$$(SLR): \begin{cases} \min & f(x) = \sum_{i=1}^{P} \frac{n_i(x)}{d_i(x)} \\ s.t. & Ax \le b, x \ge 0. \end{cases}$$

where $n_i(x)$ and $d_i(x)$ are linear functions on R^n,

$$n_i(x) = c_i^T x + d_i = \sum_{j=1}^{n} c_{ij} x_j + d_i,$$

$$d_i(x) = e_i^T x + f_i = \sum_{j=1}^{n} e_{ij} x_j + f_i,$$

$A \in R^{m\times n}$, $b \in R^m$, for each $i = 1, 2, \cdots, P$, $c_i, e_i \in R^n$, $d_i, f_i, c_{ij}, e_{ij} \in R$. X is defined as a non-empty bounded closed set: $\{x \in R^n \mid Ax \le b, x \ge 0\}$.

Manuscript received July 9, 2024; revised December 10, 2024.

This work was supported by the National Natural Science Foundation of China (Project Number: 22XGL019), Sichuan Science and Technology Program(Project Number: 24LHJJ0036).

Hanbing Mei is a postgraduate student from the School of Science, Southwest Petroleum University, Chengdu, 610500, China(e-mail: meihanbingS@163.com).

Xianfeng Ding is a professor at the School of Science, Southwest Petroleum University, Chengdu, 610500, China (Corresponding author: e-mail:fxxd@163.com).

Xiaolin Yi is a postgraduate student from the School of Science, Southwest Petroleum University, Chengdu, 610500, China(e-mail: 2083391532@qq.com).

Pengfei Wen is a postgraduate student from the School of Science, Southwest Petroleum University, Chengdu, 610500, China(e-mail: 2522303534@qq.com).

Yiyu Qin is a postgraduate student from the School of Science, Southwest Petroleum University, Chengdu, 610500, China(e-mail: 184724829@qq.com).

Qianmei Luo is a postgraduate student from the School of Science, Southwest Petroleum University, Chengdu, 610500, China(e-mail: lqmlqm1010@163.com).

If the condition $e_i^T x + f_i < 0$ holds, redefine the expression as $\frac{c_i^T x+d_i}{e_i^T x+f_i} = \frac{-(c_i^T x+d_i)}{-(e_i^T x+f_i)}$. This transformation ensures that the denominator becomes positive while preserving the essence of the original problem. Consequently, we assume that $d_i(x) > 0$.

Given the numerator $c_i^T x + d_i$, $\frac{c_i^T x+d_i+M(e_i^T x+f_i)}{e_i^T x+f_i}$ can be constructed, where M represents a sufficiently large constant to ensure that

$$c_i^T x + d_i + M\left(e_i^T x + f_i\right) > 0.$$

Therefore, in this paper, for all $x \in X$, it is given that $c_i^T x + d_i > 0$ and $e_i^T x + f_i > 0$ for indices $i = 1, 2, \cdots, P$.

As time progresses, optimization problems are gaining increasing attention. Among these, the sum-of-linear-ratios problem—a specialized type of optimization problem—has attracted considerable interest from industry experts and researchers due to its wide-ranging applications in portfolio optimization, transportation planning, data analysis, risk management, computer vision, and other fields [1]–[6].

The presence of multiple local optimal solutions in the sum-of-linear-ratios problem makes it challenging to identify the global optimum. The ongoing research endeavor has aimed at swiftly and efficiently identifying the global optimal solution. Until now, numerous global optimization algorithms have been devised to address the challenge posed by the sum-of-linear-ratios problem, including branch-and-bound series algorithms [7]–[9], image space methods [10], region segmentation algorithms [11], polynomial time approximation algorithms [12], [13], inner and outer approximation algorithms [14], proximity point algorithms [15], and so on. In 2019, Shen [16] introduced a compression technique specifically tailored for the generalized sum-of-polynomial-ratios problem. This method employs straightforward transformation and reduction strategies to convert the initial problem into a standard form of geometric programming. Compared with other methods, this method can more effectively solve the original problem. Subsequently, Ozkok [17], building on the definition of continuity in terms of (ϵ, δ), integrated the convergence conditions with the objective function of the sum-of-linear-ratios problem. He constructed an iterative constraint at a point within the feasible domain and analyzed the feasibility of the iterative algorithm by generating random large-scale test problems. In the same year, Shen [18] formulated an outer-space branch-and-bound approach for addressing the generalized linear multiplicative programming problem. Additionally, Shen established the global convergence properties of the algorithm and furnished an assessment of its computational complexity. In 2022, Jiao [19] introduced an efficient rectangular branch-and-bound algorithm in outer space to globally solve the general

sum-of-linear-ratios problem, based on the unique premise that the denominator does not equate to zero. The author presented and analyzed numerical results to demonstrate the algorithm's efficacy.

Drawing inspiration from Jiao [20], this paper introduces an outcome-space branch-and-bound algorithm specifically designed to address the sum-of-linear-ratios problem. During the construction of the relaxation problem, two relaxations were performed. Unlike Jiao [20], where both relaxations utilized the same construction method, our approach employs different methods for each relaxation. By integrating the branching process within the outcome-space, we incorporated an acceleration methodology and formulated a branch-and-bound algorithm tailored for the outcome-space, accompanied by a convergence proof. The final numerical experiments demonstrate the effectiveness of the proposed algorithm.

The framework of this paper is outlined as follows: In the second section, we derive an equivalent problem, denoted as (EP), from the sum-of-linear-ratios problem (SLR) by employing an equivalent transformation technique. We then introduce the relaxation problem (RLP) for the equivalent problem (EP), which is formulated using a two-step relaxation based on the characteristics of the equivalent function. In the third section, we detail the branch operation and acceleration methodology within the outcome-space, culminating in the proposal of the outcome-space branch-and-bound algorithm. In the fourth section provides a proof of global convergence for the algorithm, supported by two theorems. The fifth section is numerical experiment and comparison of numerical results. The sixth section is the conclusion.

II. Equivalent problem and relaxation programming

A. Equivalent problem

Prior to addressing the original problem (SLR), it undergoes an equivalent transformation:

For all $x \in X$, tackle the subsequent linear programming problems:

$$\underline{x}_i^0 = \begin{cases} \min & x_i \\ s.t. & Ax \le b, \\ & c_i^T x + d_i > 0, \quad i = 1, 2, \cdots, P, \\ & e_i^T x + e_i > 0, \quad i = 1, 2, \cdots, P, \end{cases}$$

$$\overline{x}_i^0 = \begin{cases} \max & x_i \\ s.t. & Ax \le b, \\ & c_i^T x + d_i > 0, \quad i = 1, 2, \cdots, P, \\ & e_i^T x + e_i > 0, \quad i = 1, 2, \cdots, P, \end{cases}$$

The initial rectangle $X^0 = \left[\underline{x}^0, \overline{x}^0\right] = \left[\underline{x}_i^0, \overline{x}_i^0\right]_{n\times 1}$ is obtained. Notably, X^0 encompasses all feasible solutions to the original problem (SLR).

For all $n_i(x) > 0$ and $d_i(x) > 0$, the equivalent transformation is as follows:

$$\begin{aligned} \frac{n_i(x)}{d_i(x)} &= e^{\ln \frac{n_i(x)}{d_i(x)}} \\ &= e^{[\ln n_i(x) - \ln d_i(x)]} \\ &= \exp\left[\ln n_i(x) - \ln d_i(x)\right]. \end{aligned}$$

For all $x \in X$, let

$l_i = \min_{x\in X} c_i^T x + d_i$, $u_i = \max_{x\in X} c_i^T x + d_i$,

$L_i = \min_{x\in X} e_i^T x + f_i$, $U_i = \max_{x\in X} e_i^T x + f_i$.

Subsequently, then problem (SLR) can be reformulated into the equivalent problem presented hereinafter:

$$(EP): \begin{cases} \min & h(x) = \sum_{i=1}^{P} \exp\left[\varphi_i(x)\right] \\ s.t. & \varphi_i(x) = \ln n_i(x) - \ln d_i(x), \\ & x \in X. \end{cases}$$

It is evident that the problems (SLR) and (EP) are equivalent, indicating that they share the same optimal solutions and corresponding values.

B. Relaxation programming problem

According to the properties of the equivalent problem, the relaxation problem of problem (EP) is established by two-step relaxation.

First relaxation:

Let $\varphi_i^l = \ln l_i - \ln U_i$, $\varphi_i^u = \ln u_i - \ln L_i$.

Based on the monotonicity of logarithmic functions, the following inequality is established:

$$\ln l_i \le \ln n_i(x) \le \ln u_i, \ln L_i \le \ln d_i(x) \le \ln U_i.$$

Then it can be concluded that

$$\ln n_i(x) - \ln d_i(x) \le \ln u_i - \ln L_i,$$

$$\ln l_i - \ln U_i \le \ln n_i(x) - \ln d_i(x).$$

So

$$\varphi_i^l \le \varphi_i(x) \le \varphi_i^u. \tag{1}$$

Second relaxation:

For all $x \in X$, let

$\varphi_i^{ll} = \min_{x\in X} \varphi_i^l$, $\varphi_i^{lu} = \max_{x\in X} \varphi_i^l$,

$\varphi_i^{ul} = \min_{x\in X} \varphi_i^u$, $\varphi_i^{uu} = \max_{x\in X} \varphi_i^u$,

$k_i^1 = \frac{\exp\left(\varphi_i^{lu}\right) - \exp\left(\varphi_i^{ll}\right)}{\varphi_i^{lu} - \varphi_i^{ll}}$, $k_i^2 = \frac{\exp\left(\varphi_i^{uu}\right) - \exp\left(\varphi_i^{ul}\right)}{\varphi_i^{uu} - \varphi_i^{ul}}$,

$g_i(x) = \exp\left[\varphi_i(x)\right]$, $g_i^l = k_i^1\left(1 + \varphi_i^l - lnk_i^1\right)$,

$g_i^u = k_i^2\left(\varphi_i^u - \varphi_i^{ul}\right) + \exp\left(\varphi_i^{ul}\right)$, $L = \exp\left(\varphi_i^l\right)$.

On the definition interval $\left[Y^l, Y^u\right]$ of Y, the following inequalities can be obtained from the geometric properties of $\exp(Y)$ functions:

$$A\left(1 + Y - \ln A\right) \le \exp(Y) \le A\left(Y - Y^l\right) + \exp\left(Y^l\right),$$

where $A = \frac{\exp\left(Y^u\right) - \exp\left(Y^l\right)}{Y^u - Y^l}$.

Drawing from the aforementioned, one can deduce that:

$$\exp\left(\varphi_i^u\right) \le k_i^2\left(\varphi_i^u - \varphi_i^{ul}\right) + \exp\left(\varphi_i^{ul}\right),$$

$$k_i^1\left(1 + \varphi_i^l - \ln k_i^1\right) \le \exp\left(\varphi_i^l\right) = L.$$

By combining (1), can be obtained

$$k_i^1\left[1 + \varphi_i^l(x) - \ln k_i^1\right] \le \exp\left(\varphi_i^l\right) \le \exp\left[\varphi_i(x)\right],$$

$$\exp\left[\varphi_i(x)\right] \le \exp\left(\varphi_i^u\right) \le k_i^2\left[\varphi_i^u(x) - \varphi_i^{ul}\right] + \exp\left(\varphi_i^{ul}\right).$$

That is

$$g_i^l \le L \le g_i(x). \tag{2}$$

Utilizing the information provided above, we can conclude that the relaxation programming problem for problem (EP) is:

$$(RLP):\begin{cases}\min & g^l(x)\\ s.t. & g^l(x)=\sum_{i=1}^{P} g_i^l,\\ & g_i(x)\le g_i^u,\\ & x\in X.\end{cases}$$

The construction of the relaxation problem explicitly demonstrates that feasible solutions for the equivalent problem (EP) are also viable for the relaxation problem (RLP). Furthermore, the optimal achieved by relaxation problem (RLP) is not greater than the optimal value of the equivalent problem (EP). Consequently, the optimal solution of the relaxation problem serves as a dependable lower estimate for the optimal solution of the equivalent problem.

III. Algorithm

In this paper, a branch and bound algorithm is introduced that is designed to attain the global optimal solution for the original problem (SLR). This is accomplished by the division of the rectangular space X^0 where x resides.

A. Branch operation

In this paper, the dichotomy of parameter ω is used to divide the rectangle. The purpose of the branch is to divide the initial rectangle X^0 into two smaller sub-rectangles, potentially housing the globally optimal solution of the equivalent problem (EP) within each.

Let $X^k=\left[\underline{x}^k,\overline{x}^k\right]\subseteq X^0$ represent the current rectangle to be divided, and for all $x^k\in X^k$, divide X^k in the following form:

Step 1: Calculate

$\omega=\max\left\{\left(x_i^k-\underline{x}_i^k\right)\left(\overline{x}_i^k-x_i^k\right):i=1,2,\cdots,n\right\}$,

If $\omega=0$, set $\overline{x}_u^k-\underline{x}_u^k=\max\left\{\overline{x}_i^k-\underline{x}_i^k:i=1,2,\cdots,n\right\}$, $x_u^k=\frac{\overline{x}_u^k+\underline{x}_u^k}{2}$.

Alternatively, identify the first $x_j^k\in arg\max\omega$, record it

$$x_u^k=x_j^k.$$

Step 2: Record

$\overline{x}=\left\{\underline{x}_1^k,\underline{x}_2^k,\cdots,\underline{x}_{i-1}^k,x_u^k,\underline{x}_{i+1}^k,\cdots,\underline{x}_n^k\right\}^T$,

$\overline{\overline{x}}=\left\{\overline{x}_1^k,\overline{x}_2^k,\cdots,\overline{x}_{i-1}^k,x_u^k,\overline{x}_{i+1}^k,\cdots,\overline{x}_n^k\right\}^T$.

The rectangle X^k is divided into two subrectangles by the line of $\overline{x}$ and $\overline{\overline{x}}$ or the hyperplane where the line is located:

$$X^{k1}=\left[\underline{x}^{k1},\overline{x}^{k1}\right]=\prod_{i=1}^{u-1}\left[\underline{x}_i^k,\overline{x}_i^k\right]\times\left[\underline{x}_u^k,x_u^k\right]\times\prod_{i=u+1}^{n}\left[\underline{x}_i^k,\overline{x}_i^k\right],$$

$$X^{k2}=\left[\underline{x}^{k2},\overline{x}^{k2}\right]=\prod_{i=1}^{u-1}\left[\underline{x}_i^k,\overline{x}_i^k\right]\times\left[x_u^k,\overline{x}_u^k\right]\times\prod_{i=u+1}^{n}\left[\underline{x}_i^k,\overline{x}_i^k\right].$$

B. Acceleration technique

Utilizing the aforementioned branching operations, we introduce a reduction strategy for each subrectangle, aimed at eliminating intervals devoid of the global optimal solution for the original problem, thereby enhancing the algorithm's efficiency. Specific acceleration techniques are detailed as follows:

Without loss of generality, since we are dividing in the outcome-space where x is, we can equivalently rewrite the objective function of problem (RLP) in the manner presented below:

$$g^l(x)=\sum_{i=1}^{n}\alpha_i x_i+\delta.$$

Define UB_k as the upper bound of the known global optimal value after k iterations of the problem(SLR), let

$$RLB=\sum_{i=1,\alpha_i>0}^{n}\alpha_i\underline{x}_i+\sum_{i=1,\alpha_i<0}^{n}\alpha_i\overline{x}_i+\delta,$$

$$\rho_q=\frac{UB_k-RLB+\min\left\{\alpha_q\overline{x}_q,\alpha_q\underline{x}_q\right\}}{\alpha_q},$$

$$X_i^1=\begin{cases}X_i,i\ne q,i=1,2,\cdots,n,\\ (\rho_q,\overline{x}_q]\bigcap X_q,i=q;\end{cases}$$

$$X_i^2=\begin{cases}X_i,i\ne q,i=1,2,\cdots,n,\\ \left[\underline{x}_q,\rho_q\right)\bigcap X_q,i=q.\end{cases}$$

Theorem 1: For any subrectangle $X\subseteq X^0$, the following conclusions holds:

(i) If $RLB>UB_k$, then the subrectangle X does not contain the global optimal solution of problem (SLR).

(ii) If $RLB\le UB_k$, then for all $q\in\left\{1,2,\cdots,n\right\}$,

When $\alpha_q>0$, the subrectangle $X^1=\left(X_i^1\right)_{n\times1}$ does not contain the global optimal solution of problem (SLR).

When $\alpha_q<0$, the subrectangle $X^2=\left(X_i^2\right)_{n\times1}$ does not contain the global optimal solution of problem (SLR).

Proof: (i) If $RLB>UB_k$, then for all $x\subseteq X$, there is $UB_k<RLB\le g^l(x)$, that is,

$$UB_k<RLB=\sum_{i=1,\alpha_i>0}^{n}\alpha_i\underline{x}_i+\sum_{i=1,\alpha_i<0}^{n}\alpha_i\overline{x}_i+\delta\le g^l(x)\le g(x).$$

Therefore, the subrectangle X lacks the global optimal solution to the problem (SLR).

(ii) If $RLB\le UB_k$, then for all $q\in\left\{1,2,\cdots,n\right\}$, when $\alpha_q>0$, for all $x\subseteq X^1$, we have $x_q>\rho_q$, that is

$$x_q>\frac{UB_k-RLB+\min\left\{\alpha_q\overline{x}_q,\alpha_q\underline{x}_q\right\}}{\alpha_q},$$

then

$$\begin{aligned}g(x)&\ge g^l(x)\\&\ge\sum_{i=1,i\ne q}^{n}\min\left\{\alpha_i\overline{x}_i,\alpha_i\underline{x}_i\right\}+\delta+\alpha_q x_q\\&\ge\sum_{i=1,i\ne q}^{n}\min\left\{\alpha_i\overline{x}_i,\alpha_i\underline{x}_i\right\}+\delta\\&\quad+\alpha_q\times\frac{UB_k-RLB+\min\left\{\alpha_q\overline{x}_q,\alpha_q\underline{x}_q\right\}}{\alpha_q}\\&=RLB+UB_k-RLB\\&=UB_k.\end{aligned}$$

Therefore, for all $x\subseteq X^1$, there is $g(x)\ge g^l(x)>UB_k$, indicating that the subrectangle X^1 cannot possess a global

optimal solution to the problem (SLR). Analogously, in the case where $\alpha_q < 0$, the subrectangle X^2 does not possess a global optimal solution to the problem (SLR).

In summary, the theorem has been established. ■

C. Outcome-space branch-and-bound algorithm

Utilizing the aforementioned branch operation, we will now present the branch-and-bound algorithm pertaining to the outcome space:

Step 0 (Initialization): With a specified error tolerance $\epsilon > 0$, determine the optimal solution x^0 and its corresponding optimal value $LB^0 = g^l\left(x^0\right)$ for the (RLP) problem on $X = X^0$, and set $UB^0 = h\left(x^0\right)$.

Step 1 (Termination criteria): If $UB^0 - LB^0 \leq \epsilon$, the calculation terminates, and x^0 is the global optimal solution to problem (SLR). Otherwise, initialize $Q^0 = \left\{X^0\right\}$, set the iteration number $k = 1$, and proceed to step 2.

Step 2: $k \geq 1$.

Step 3 (Branch of outcome-space): According to the dividing rule in section III-A, select the rectangle X^k with a smaller optimal value and divide it into two parts X^{k1} and X^{k2}. $X^{k1} \bigcap X^{k2}$ is an empty set, and combined with the acceleration technique in section III-B, delete the subrectangles that do not contain the global optimal solution. Denote Λ represent the collection of all discarded sub-rectangles. So the remaining partition set is $Q^k = \left\{X \mid X \in Q^{k-1} \cup \left\{X^{k1}, X^{k2}\right\}, X \notin \Lambda\right\}$, and let $R = \left\{X^{k1}, X^{k2}\right\}$.

Step 4 (Delimiting and cutting branches): For subrectangles X^{k1} and X^{k2}, if X^{kr} is not an empty set and $r = 1, 2$, determine the optimal solution x^{kr} and the optimal value $LB^k = g^l\left(x^{kr}\right)$ for problem (RLP) within X^{k1} and X^{k2}. If $UB^k < LB^k$, then $R = R \setminus \left\{X^{kr}\right\}$, leading to two possible cases:

Case 1: If R is an empty set, go to step 2;

Case 2: If R is not an empty set, get Q^k and update the upper bound $UB^k = min\left\{UB^{k-1}, h\left(x^{kr}\right)\right\}$. Select x^k to satisfy $UB^k = h\left(x^k\right)$, and proceed to step 5.

Step 5 (Judgment rule): Let
$Q^{k+1} = Q^k \setminus \left\{\overline{X} : UB^k\left(\overline{X}\right) - LB^k \leq \epsilon, \overline{X} \in Q^k\right\}$.

If Q^{k+1} is an empty set, the calculation stops, x^k is identified as the global optimal solution for problem (SLR) with UB^k being the corresponding global optimal value. Alternatively, if Q^{k+1} is not an empty set, let $k = k + 1$, select X^k to satisfy $X^k = arg\min_{\overline{X} \in Q^k} LB\left(\overline{X}\right)$, and revert to step 2.

IV. Convergence analysis

The convergence of this algorithm on a global scale is established in Theorems 2 and 3 presented hereinafter.

Theorem 2: For all $x \in X^0$, during the running of the algorithm, for the rectangle X^k to be branched, $\left|h\left(x\right) - g^l\left(x\right)\right| \to 0$ when $\triangle = |\overline{x} - \underline{x}| \to 0$.

Proof: $\left|h\left(x\right) - g^l\left(x\right)\right| = \left|\sum_{i=1}^{P} g_i\left(x\right) - \sum_{i=1}^{P} g_i^l\right| = \sum_{i=1}^{P}\left|g_i\left(x\right) - g_i^l\right|$, From equation (2), it can be transformed into:

$$\begin{aligned}\left|h\left(x\right) - g^l\left(x\right)\right| &= \sum_{i=1}^{P}\left|g_i\left(x\right) - g_i^l\right| \\ &= \sum_{i=1}^{P}\left|g_i\left(x\right) - L + L - g_i^l\right| \\ &\leq \sum_{i=1}^{P}\left|g_i\left(x\right) - L\right| + \sum_{i=1}^{P}\left|L - g_i^l\right|.\end{aligned} \tag{3}$$

Record $\triangle_1 = g_i\left(x\right) - L$ and $\triangle_2 = L - g_i^l$, and from equation (2), it can be concluded that $\triangle_1 \geq 0$ and $\triangle_2 \geq 0$.

Let's first prove the part $\triangle_1$:

$$\begin{aligned}\triangle_1 &= g_i\left(x\right) - L \\ &= \exp\left[\varphi_i\left(x\right)\right] - \exp\left(\varphi_i^l\right) \\ &\leq \left|\varphi_i\left(x\right) - \varphi_i^l\right| \sup_{\theta_i \in L\left(\varphi_i^l, \varphi_i(x)\right)} \exp\left(\theta_i\right),\end{aligned}$$

where $L\left(\varphi_i^l, \varphi_i\left(x\right)\right) = \beta\varphi_i^l + \left(1 - \beta\right)\varphi_i\left(x\right)$ and $\beta \in [0, 1]$.

For $\triangle_1$, we mainly focus on the part of $\left|\varphi_i\left(x\right) - \varphi_i^l\right|$:

$$\begin{aligned}\varphi_i\left(x\right) - \varphi_i^l &= \ln n_i\left(x\right) - \ln d_i\left(x\right) - \left(\ln l_i - \ln U_i\right) \\ &= \ln n_i\left(x\right) - \ln l_i + \ln U_i - \ln d_i\left(x\right) \\ &= \frac{1}{t_1}\left[n_i\left(x\right) - l_i\right] + \frac{1}{t_2}\left[U_i - d_i\left(x\right)\right],\end{aligned} \tag{4}$$

where $t_1 \in \left(l_i, n_i\left(x\right)\right)$, $t_2 \in \left(d_i\left(x\right), U_i\right)$.

At this time

$$\begin{aligned}n_i\left(x\right) - l_i &= c_i^T x + d_i - \min\left(c_i^T x + d_i\right) \\ &= c_i^T\left[x - \min\left(x\right)\right] \\ &\leq c_i^T\left(\overline{x} - \underline{x}\right).\end{aligned}$$

So when $\triangle \to 0$, $\left[n_i\left(x\right) - l_i\right] \to 0$. Similarly, it can be proven that when $\triangle \to 0$, $\left[U_i - d_i\left(x\right)\right] \to 0$.

Since $\left[n_i\left(x\right) - l_i\right] \to 0$, $\left[U_i - d_i\left(x\right)\right] \to 0$, and t_1, t_2 are bounded quantities, $\left|\varphi_i\left(x\right) - \varphi_i^l\right| \to 0$ can be obtained. Similarly, it can be proven that when $\left|\varphi_i\left(x\right) - \varphi_i^l\right| \to 0$, $\triangle_1 \to 0$, so when $\triangle \to 0$, $\triangle_1 \to 0$.

Next, we will prove that part $\triangle_2$:

$$\begin{aligned}\triangle_2 &= L - g_i^l \\ &= \exp\left(\varphi_i^l\right) - k_i^1\left[1 + \varphi_i^l\left(x\right) - \ln k_i^1\right] \\ &= \exp\left(\varphi_i^l\right) - \exp\left(\varphi_i^l\right) \times \frac{\exp\left(w_i\right) - 1}{w_i} \\ &\times \left(1 + \varphi_i^l - \ln\frac{\exp\left(\varphi_i^l\right)\left[\exp\left(w_i\right) - 1\right]}{w_i}\right) \\ &= \exp\left(\varphi_i^l\right)\left(1 - K_i + K_i \ln K_i\right),\end{aligned}$$

where $w_i = \varphi_i^{lu} - \varphi_i^{ll}$, $K_i = \frac{\exp(w_i) - 1}{w_i}$.

Similar to the proof of equation (4), it can be obtained that when $\triangle \to 0$, $\left|\varphi_i^{lu} - \varphi_i^{ll}\right| \to 0$. When $\left|\varphi_i^{lu} - \varphi_i^{ll}\right| \to 0$, $K_i \to 1$, then $\triangle_2 \to 0$.

Therefore, when $\triangle = |\overline{x} - \underline{x}| \to 0$, $\triangle_1 \to 0$, $\triangle_2 \to 0$, combined with equation (3), $\left|h\left(x\right) - g^l\left(x\right)\right| \to 0$. This theorem is proven. ■

Theorem 3: In the case where the algorithm's iterative process concludes after a finite number of steps, the global optimal solution to problem (SLR) is attainable upon termination; Conversely, if the algorithm does not terminate within a finite number of steps, it will produce an infinite sequence $\left\{x^k\right\}$, where each accumulation point of this sequence corresponds to the global optimal solution of problem (SLR).

Proof: When the algorithm terminates after a finite number of steps, it can be configured to conclude at step k. Define x^* as the optimal solution obtained by solving problem (RLP). Notably, x^* is also the feasible solution for problem (SLR). Consequently, it follows that $g^l\left(x^k\right) \leq h\left(x^*\right) \leq h\left(x^k\right)$.

According to the termination criteria:

$$\left|h\left(x^k\right) - g^l\left(x^k\right)\right| \leq \epsilon.$$

Combine the above two equations:

$$h\left(x^k\right) \leq g^l\left(x^k\right) + \epsilon \leq h\left(x^*\right) + \epsilon \leq h\left(x^k\right) + \epsilon.$$

When the algorithm terminates within a finite number of steps, the global ϵ-optimal solution to problem (SLR) can be obtained.

If the algorithm cannot terminate within a finite number of steps, an infinite sequence of feasible solutions $\left\{x^k\right\}$ for the original problem (SLR) can be generated by solving problem (RLP). As the rectangular space of x continues to subdivide, there is $\lim_{k\to\infty} x^k = x^*$ and $LB^k = g^l\left(x^k\right) \leq h\left(x^*\right) \leq h\left(x^k\right) = UB^k$, where x^* is the accumulation point of the sequence $\left\{x^k\right\}$.

According to the algorithm, $\left\{LB^k\right\}$ and $\left\{UB^k\right\}$ are both convergent sequences,then

$$\lim_{k\to\infty} LB^k \leq h\left(x^*\right) \leq \lim_{k\to\infty} UB^k.$$

The rectangle is continuously subdivided by the branch operation in section III-A, and $h\left(x\right)$ is a continuous function, so

$\lim_{k\to\infty} LB^k = \lim_{k\to\infty} g^l\left(x^k\right) = h\left(x^*\right) = \lim_{k\to\infty} h\left(x^k\right) = \lim_{k\to\infty} UB^k$.

The theorem is proven. ■

V. Numerical experiments

To showcase the effectiveness of the algorithm proposed in this paper, numerical experiments will be presented to demonstrate the superiority and inferiority of the algorithm by comparing it with results from known references.

The following numerical experiments are compiled and run on $Matlab\,(2018b)$. All calculation processes are performed on a personal computer with $AMD\ Ryzen\ 77840HS\ w/Radeon\ 780M\ Graphics\ 3.80\,GHz$ processor $32GB$ of memory and $Win11$ operating system.

Where Iter represents the number of iterations; Time (seconds) represents the algorithm's running time measured in seconds; Additionally, p stands for the count of linear fractions; m for the number of linear constraints; n is the number of variables; Ave.NT represents the mean iteration count, while Ave.time indicates the average running time; Std.NT represents the standard deviation of the number of iterations; Std.time represents the standard deviation of running time. Except for iterations that are precise to one decimal place, all other decimals are rounded to four decimal places.

Example 1: (See [12], [21])

$$\min \frac{-x_1 + 2x_2 + 2}{3x_1 - 4x_2 + 5} + \frac{4x_1 - 3x_2 + 4}{-2x_1 + x_2 + 3}$$

$$\begin{aligned} s.t. \quad & x_1 + x_2 \leq 1.5, \\ & x_1 - x_2 \leq 0, \\ & 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1. \end{aligned}$$

Example 2: (See [22], [23])

$$\min \frac{3x_1 + 5x_2 + 3x_3 + 50}{3x_1 + 4x_2 + 5x_3 + 50} + \frac{3x_1 + 4x_2 + 50}{4x_1 + 3x_2 + 2x_3 + 50} + \frac{4x_1 + 2x_2 + 4x_3 + 50}{5x_1 + 4x_2 + 3x_3 + 50}$$

$$\begin{aligned} s.t. \quad & 2x_1 + x_2 + 5x_3 \leq 10, \\ & x_1 + 6x_2 + 2x_3 \leq 10, \\ & 9x_1 + 7x_2 + 3x_3 \geq 10, \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

Example 3: (See [11], [21])

$$\min \frac{4x_1 + 3x_2 + 3x_3 + 50}{3x_2 + 3x_3 + 50} + \frac{3x_1 + 4x_3 + 50}{4x_1 + 4x_2 + 5x_3 + 50} + \frac{x_1 + 2x_2 + 4x_3 + 50}{x_1 + 5x_2 + 5x_3 + 50} + \frac{x_1 + 2x_2 + 4x_3 + 50}{5x_2 + 4x_3 + 50}$$

$$\begin{aligned} s.t. \quad & 2x_1 + x_2 + 5x_3 \leq 10, \\ & x_1 + 6x_2 + 2x_3 \leq 10, \\ & 9x_1 + 7x_2 + 3x_3 \geq 10, \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

Example 4: (See [21], [24])

$$\min -\frac{3x_1 + 5x_2 + 3x_3 + 50}{3x_1 + 4x_2 + 5x_3 + 50} - \frac{3x_1 + 4x_2 + 50}{4x_1 + 3x_2 + 2x_3 + 50} - \frac{4x_1 + 2x_2 + 4x_3 + 50}{5x_1 + 4x_2 + 3x_3 + 50}$$

$$\begin{aligned} s.t. \quad & 6x_1 + 3x_2 + 3x_3 \leq 10, \\ & 10x_1 + 3x_2 + 8x_3 \leq 10, \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

Example 5: (See [8], [23])

$$\min \frac{37x_1 + 73x_2 + 13}{13x_1 + 13x_2 + 13} + \frac{63x_1 - 18x_2 + 39}{13x_1 + 26x_2 + 13}$$

$$\begin{aligned} s.t. \quad & 5x_1 - 3x_2 = 3, \\ & 1.5 \leq x_1 \leq 3. \end{aligned}$$

Example 6: (See [21], [25])

$$\max \frac{4x_1 + 3x_2 + 3x_3 + 50}{3x_2 + 3x_3 + 50} + \frac{3x_1 + 4x_2 + 50}{4x_1 + 4x_2 + 5x_3 + 50} + \frac{x_1 + 2x_2 + 5x_3 + 50}{x_1 + 5x_2 + 5x_3 + 50} + \frac{x_1 + 2x_2 + 4x_3 + 50}{5x_2 + 4x_3 + 50}$$

$$\begin{aligned} s.t. \quad & 2x_1 + x_2 + 5x_3 \leq 10, \\ & x_1 + 6x_2 + 3x_3 \leq 10, \\ & 5x_1 + 9x_2 + 2x_3 \leq 10, \\ & 9x_1 + 7x_2 + 3x_3 \leq 10, \\ & x_1 \geq 0, x_2 \geq 0, x_3 \geq 0. \end{aligned}$$

Example 7: (See [8], [26])

$$\max \frac{37x_1 + 73x_2 + 13}{13x_1 + 13x_2 + 13} + \frac{63x_1 - 18x_2 + 39}{-13x_1 - 26x_2 - 13} + \frac{13x_1 + 13x_2 + 13}{63x_1 - 18x_2 + 39} + \frac{13x_1 + 26x_2 + 13}{-37x_1 - 73x_2 - 13}$$

$$s.t. \quad 5x_1 - 3x_2 = 3,$$
$$1.5 \leq x_1 \leq 3.$$

Example 8: (See [27])

$$\min \sum_{i=1}^{p} \frac{\sum_{j=1}^{n} d_{ij}x_j + C}{\sum_{j=1}^{n} c_{ij}x_j + C}$$

$$s.t. \quad \sum_{j=1}^{n} a_{kj}x_j \leq 10, k = 1, 2, \cdots, m,$$
$$x_j \geq 0, j = 1, 2, \cdots, n.$$

Among them, c_{ij}, d_{ij}, and a_{kj} are randomly produced within $[0, 10]$, $C = 10$; $k = 1, 2, \cdots, m$; $j = 1, 2, \cdots, n$.

The outcomes of the computations for Example 1-Example 7 are displayed in Table I. The table clearly demonstrates that our algorithm's optimal solutions and values are not significantly different from those in other references, with some remaining consistent, such as in Example 3, Example 4, Example 6, and Example 7. Regarding the number of iterations, the algorithm proposed in this paper exhibits the fewest iterations in Examples 1 and 5, while having more iterations than the algorithms from other references in the remaining examples. From the perspective of running time, the algorithm proposed in this paper outperforms other algorithms in Example 1, Example 2, and Example 5, while the difference in running time compared to other examples is not significant.

From Table I, it is clear that the algorithm presented in this paper is both feasible and effective. Next, we will exemplify the performance of the algorithm when running random cases through Example 8. For Example 8, the generation method and error tolerance of the relevant data in reference [27] were consistent. The algorithm was run 10 times, and the obtained data results were compared with those in reference [27], as shown in Table II.

As shown in Table II, the algorithm proposed in this paper demonstrates a clear advantage in the number of iterations. Through equivalent transformation and linear relaxation, we have transformed the sum-of-linear-ratios problems that are difficult to solve into linear programming problems, greatly reducing the number of iterations. Regarding running time, the change of the running time of the algorithm in this paper conforms to the basic fact. Although the initial time used is longer than that of the algorithm in the reference [27], with the increase of the complexity of the example, the algorithm in this paper is significantly better than that in the reference [27], and the time used is significantly less.

From Table II, it is apparent that as p and n increase, there are significant changes in the number of iterations and running time used by the algorithm. Consequently, we will conduct large-scale examples to delve deeper into the computational efficiency of the algorithm. Increase p and n, randomly run the algorithm 10 times, observe the changes in iteration times and running time, and the specific results are shown in Table III.

From Table III, it is evident that the algorithm in this paper is feasible and effective in running large-scale examples. Moreover, it can be seen that among the two factors that affect the number of iterations and running time, the impact of n is significantly greater than that of p. As p and n increase, the number of iterations and running time of the algorithm also gradually rise, with a slower increase in iteration times and a more rapid change in running time. Notably, when n increases to 2000, the running time of the algorithm in this paper increases sharply. As shown in Table III, the time is all above 400s. Therefore, when dealing with large-scale numerical cases with variable dimensions less than 2000, the algorithm demonstrates a distinct advantage in terms of time efficiency.

In summary, from Table I concludes that the algorithm proposed in this paper is practical, with the generated optimal solutions and values aligning closely with those from other references, and in some cases, outperforming them. Table II highlights the algorithm's performance with random cases, demonstrating its advantage in terms of the number of iterations. Finally, Table III explores two factors affecting the algorithm's efficiency and identifies the optimal operating range for the proposed algorithm.

VI. Concluding remarks

This paper introduces an equivalent transformation tailored to the specific attributes of the objective function for sum-of-linear-ratios problems. Two different relaxation methods are employed to derive the relaxation problem. By integrating the branch-and-bound framework with an acceleration strategy, an outcome-space branch-and-bound algorithm is proposed. Examples 1-7 showcase the viability of the proposed algorithm. Due to the sufficient relaxation transformation of the objective function, our algorithm uses fewer iterations to solve numerical cases, offering certain advantages over algorithms presented in other references. When solving large-scale numerical cases, our algorithm has less running time when solving numerical cases with variable dimensions less than 2000. As the variable dimension continues to increase, the time used will increase rapidly. So the future research topic is how to reduce the running time of large-scale numerical cases with high variable dimensions.

References

[1] I. Stancu-Minasian, *Fractional Programming: Theory, Methods and Applications*. Kluwer Academic Publishers, 1997.

[2] N. T. H. P. Tuy, "A unified monotonic approach to generalized linear fractional programming," *J Glob Optim*, no. 3, pp. 229–259, 2003.

[3] Y. Liu, Xia;Gao, "A new global optimization algorithm for a class of linear fractional programming," *Mathematics*, vol. 7, no. 9, p. 867, 2019.

[4] W. Jiao, Hongwei;Wang, "Outer space branch-reduction-bound algorithm for solving generalized affine multiplicative problems," *J Comput Appl Math*, vol. 419, p. 114784, 2023.

[5] B. E, *Linear-fractional programming: theory, methods, applications and software*. Dordrecht: Kluwer Academic Publishers, 2003.

[6] J. Schaible, Siegfried;Shi, "Fractional programming the sum-of-ratios case," *Optim Methods Softw*, vol. 18, no. 2, pp. 219–229, 2003.

[7] Z. B. Liu X, Gao Y L, "A new global optimization algorithm for a class of linear fractional programming," *Mathematics*, vol. 7, no. 9, p. 867, 2019.

[8] G. L. Liu S, "An outcome space algorithm for minimizing a class of linear ratio optimization problems," *Comput Appl Math*, vol. 40, no. 6, pp. 1–17, 2021.

[9] Y. G. Bo Z, "A new deterministic algorithm for solving the global solution of the generalized quadratic ratio problem," *Mathematica Applicata*, vol. 32, no. 04, pp. 767–777, 2019.

TABLE I
COMPARISON OF THE RESULTS OF EXAMPLE 1-7

Ex.	Refs.	ϵ	Optimal solution	Optimal value	Iter	Time(seconds)
1	ours	10^{-2}	$(0, 0.2839)$	1.6232	10	0.2792
1	[12]	10^{-2}	$(0, 0.2816)$	1.6232	17	0.4600
1	[21]	10^{-2}	$(0, 0.2847)$	1.6231	22	0.4248
2	ours	10^{-4}	$(4.9998, 0, 0)$	2.8619	16	0.4246
2	[22]	10^{-4}	$(0, 3.3333, 0)$	2.8619	80	8.5663
2	[23]	10^{-4}	$(5, 0, 0)$	2.8619	12	28.2943
3	ours	10^{-6}	$(0, 1.6667, 0)$	3.7109	18	0.4301
3	[11]	10^{-6}	$(0, 1.6667, 0)$	3.7192	7	0.1057
3	[21]	10^{-4}	$(0, 1.6667, 0)$	3.7109	32	0.9698
4	ours	10^{-6}	$(0, 0.3333, 0)$	-3.0029	24	0.4992
4	[21]	10^{-6}	$(0, 0.3333, 0)$	-3.0029	18	1.1676
4	[24]	10^{-3}	$(0, 0.3333, 0)$	-3.0029	17	0.1290
5	ours	10^{-4}	$(1.5000, 1.5000)$	4.9126	24	0.6655
5	[8]	10^{-4}	$(1.5000, 1.5000)$	4.9159	49	1.1896
5	[23]	10^{-4}	$(1.5000, 1.5000)$	4.9125	113	201.6260
6	ours	10^{-9}	$(1.1111, 0, 0)$	4.0907	16	0.4207
6	[21]	10^{-9}	$(1.1111, 0, 0)$	4.0907	18	0.4782
6	[25]	10^{-6}	$(1.1111, 0, 0)$	4.0907	2	0.0020
7	ours	10^{-6}	$(3, 4)$	3.2917	32	0.9204
7	[8]	10^{-6}	$(3, 4)$	3.2967	7	0.1903
7	[26]	10^{-6}	$(3, 4)$	3.2917	78	0

TABLE II
COMPARISON OF NUMERICAL RESULTS OF EXAMPLE 8 SOLVED BY THE ALGORITHM IN THIS PAPER AND THE ALGORITHM IN REFERENCE [27]

	[27]		Ours	
(p, m, n)	Avg(Std).NT	Avg(Std).Time	Avg(Std).NT	Avg(Std).Time
$(2, 100, 100)$	8.6 (1.0198)	0.5727 (0.0723)	3.3 (1.2774)	4.5041 (0.3073)
$(2, 100, 300)$	8.2 (1.2489)	1.1530 (0.1898)	3.6 (0.2530)	23.8519 (4.3734)
$(2, 100, 500)$	8.6 (1.1135)	2.0478 (0.2603)	3.9 (0.2400)	58.9631 (7.5078)
$(3, 50, 50)$	25.1 (14.0689)	0.4202 (0.2423)	4.5 (0.2713)	0.9660 (0.0925)
$(3, 50, 100)$	35.7 (14.9669)	0.9662 (0.4331)	3.8 (0.6450)	2.5945 (0.1582)
$(3, 50, 200)$	55.6 (24.1669)	2.3498 (1.0436)	3.6 (0.6155)	6.6605 (1.4126)
$(3, 50, 500)$	58.6 (15.9824)	5.9055 (1.5506)	3.9 (0.4630)	41.1246 (8.7633)
$(3, 100, 50)$	39.2 (19.8927)	1.4321 (0.7536)	1.4 (0.2040)	0.3076 (0.0429)
$(3, 100, 100)$	46.4 (19.1739)	3.3694 (1.3991)	4.9 (0.5276)	3.7378 (0.6909)
$(3, 100, 200)$	50.6 (16.8178)	5.5343 (1.9166)	5.0 (0.8015)	12.7616 (1.1501)
$(3, 100, 300)$	51.9 (17.8742)	8.2884 (2.8661)	5.4 (0.3250)	22.4608 (2.1761)
$(3, 100, 400)$	52.0 (14.9922)	14.3724 (3.1495)	5.5 (0.5367)	40.3700 (6.5027)
$(3, 100, 500)$	51.9 (9.7103)	14.3911 (2.5609)	5.8 (0.2332)	43.4990 (12.7230)
$(4, 100, 100)$	225.3 (127.7756)	17.2119 (9.8605)	5.1 (0.2713)	3.6343 (0.5928)
$(4, 100, 200)$	579.4 (549.9580)	71.4312 (67.9846)	5.3 (0.7263)	15.0623 (0.5928)
$(4, 100, 300)$	681.5 (282.7013)	118.6573 (45.9829)	5.7 (0.0980)	20.4332 (1.3415)
$(4, 100, 400)$	582.2 (229.4980)	180.0370 (68.8793)	5.7 (0.3072)	39.2040 (4.2542)
$(4, 100, 500)$	505.2 (224.8082)	150.0132 (60.0158)	5.9 (0.2598)	51.9276 (9.1107)
$(5, 100, 100)$	1527.4 (1035.6756)	123.2237 (84.5176)	5.1 (0.4833)	3.8532 (0.4892)
$(5, 100, 300)$	2902.1 (2399.7190)	515.4426 (496.4284)	5.6 (0.4079)	28.9320 (2.2488)
$(6, 100, 100)$	8865.7 (10071.6115)	819.3571 (995.1982)	5.3 (0.8352)	3.8293 (0.6991)
$(6, 100, 300)$	11612.4 (7441.3452)	2220.1240 (1397.8028)	6.1 (0.6765)	26.1969 (8.3079)
$(6, 100, 500)$	10913.7 (6417.2996)	4725.6321 (1439.3208)	6.2 (0.5122)	71.6661 (3.2750)

[10] P. S. W. Falk J E, "Image space analysis of generalized fractional programs," *J Glob Optim*, vol. 4, no. 1, pp. 63–88, 1994.

[11] L. T. Shen P P, "Regional division and reduction algorithm for minimizing the sum of linear fractional functions," *J Inequal Appl*, vol. 2018, no. 1, pp. 1–19, 2018.

[12] W. C. F. Shen P P, Zhang T L, "Solving a class of generalized fractional programming problems using the feasibility of linear programs," *J Inequal Appl*, vol. 2017, no. 1, p. 147, 2017.

[13] W. L. F. Shen P P, Huang B D, "Range division and linearization algorithm for a class of linear ratios optimization problems," *J Comput Appl Math*, vol. 350, no. 1, pp. 324–342, 2019.

[14] T. T. S. Yamada, "An outer approximation algorithm guaranteeing feasibility of solutions and approximate accuracy of optimality," *J Global Optim*, vol. 14, no. 3, pp. 267–281, 1994.

[15] R. A. Addoune S, Haffari M, "A proximal point algorithm for generalized fractional programs," *Optimization*, vol. 66, no. 9, pp. 1–23, 2017.

[16] C. X. Shen P, Zhu Z, "A practicable contraction approach for the sum of the generalized polynomial ratios problem," *Eur J Oper Res*, vol. 278, no. 1, pp. 36–48, 2019.

[17] B. A. Ozkok, "An iterative algorithm to solve a linear fractional programming problem," *Comput Ind Eng*, vol. 140, p. 106234, 2020.

[18] P. S. W. Lu, "Outer space branch and bound algorithm for solving linear multiplicative programming problems," *J Global Optim*, vol. 78, no. 3, pp. 453–482, 2020.

[19] H. J. Ma, "An efficient algorithm and complexity result for solving the sum of general affine ratios problem," *Chaos Soliton Fract*, vol. 164, p. 112701, 2022,.

[20] Y. Jiao, Hongwei;Shang, "Two-level linear relaxation method for generalized linear fractional programming," *J Oper Res Soc China*, vol. 11, no. 3, pp. 569–594, 2022.

[21] G. Y. L. B, "An outcome-space branch and bound algorithm for the sum-of-linear-ratios programming problem," *Mathematica Numerica Sinica*, vol. 42, no. 2, pp. 207–222, 2020.

[22] S. Chun-Feng, W;Pei-Ping, "A global optimization algorithm for linear fractional programming," *Appl Math Comput*, vol. 204, no. 1, pp. 281–287, 2008.

[23] S. Gao, Y;Jin, "A global optimization algorithm for sum of linear ratios problem," *J Appl Math*, vol. 2013, pp. 276 245(1–7), 2013.

[24] Y. Liu, Xia;Gao, "A new global optimization algorithm for a class of

TABLE III
THE NUMERICAL RESULTS OF SOLVING LARGE-SCALE EXAMPLE [8] USING THE ALGORITHM IN THIS PAPER

(p, m, n)	Avg(Std).NT	Avg(Std).Time
$(2, 100, 1000)$	$5.7\,(0.1608)$	$139.1415\,(6.8699)$
$(2, 100, 1500)$	$8.5\,(3.6172)$	$154.5562\,(4.9665)$
$(2, 100, 2000)$	$10.4\,(8.1572)$	$434.0015\,(28.4081)$
$(2, 100, 2500)$	$13.9\,(10.5893)$	$562.7714\,(22.3976)$
$(2, 100, 3000)$	$16.6\,(14.7656)$	$654.3368\,(31.8562)$
$(3, 100, 1000)$	$10.1\,(10.2975)$	$135.0691\,(5.0266)$
$(3, 100, 1500)$	$12.1\,(9.3897)$	$162.7040\,(2.6869)$
$(3, 100, 2000)$	$15.8\,(8.3232)$	$479.8664\,(21.4855)$
$(3, 100, 3000)$	$19.4\,(15.6370)$	$627.1043\,(15.4494)$
$(4, 100, 1000)$	$13.6\,(20.8094)$	$143.7965\,(10.6357)$
$(4, 100, 1500)$	$14.2\,(10.9592)$	$176.5548\,(12.1084)$
$(4, 100, 2000)$	$17.7\,(23.5633)$	$488.2862\,(17.0485)$
$(4, 100, 3000)$	$20.3\,(20.0725)$	$686.5297\,(13.4271)$
$(5, 100, 1000)$	$19.8\,(22.4519)$	$195.1658\,(27.2763)$
$(5, 100, 1500)$	$22.4\,(7.0843)$	$209.5611\,(36.1787)$
$(5, 100, 2000)$	$24.6\,(12.2354)$	$498.7044\,(25.8275)$
$(5, 100, 3000)$	$31.4\,(21.9280)$	$550.0822\,(20.8717)$
$(6, 100, 1000)$	$23.2\,(18.7658)$	$243.5865\,(6.7969)$
$(7, 100, 1000)$	$27.5\,(20.4421)$	$285.9020\,(24.4101)$

linear fractional programming," *Mathematics*, vol. 7, no. 9, p. 867, 2019,.

[25] S. Jiao, HW ;Liu, "Outcome space range reduction method for global optimization of sum of affine ratios problem," *Open Mathematics*, vol. 14, no. 1, pp. 736–746, 2016.

[26] S. Y, "Global optimization for sum of ratios problems," Ph.D. dissertation, Henan Normal University, 2011.

[27] Y. G. Bo Z, "An out-space branch-and-bound algorithm for finding the global solution of the sum-of-linear-ratios problem," *Mathematica Numerica Sinica*, vol. 44, no. 2, pp. 233–256, 2022.