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

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

Abstract—**This paper presents a fresh global optimization algorithm specifically designed for the sum-of-linear-ratios problem. Initially, auxiliary variables are introduced to reformulate the original problem into an equivalent form, and the equivalence of the optimal solution in this new formulation is established. Additionally, the linear relaxation technique is applied to simplify the objective function and constraint conditions of the equivalent problem, thereby obtaining a linear relaxation programming problem. By resolving a progression of linear programming instances, the upper bound of the original problem undergoes continuous updates, while the linear relaxation problem offers a viable and efficient lower bound for the original problem. Subsequently, an improved branch-and-bound algorithm for tackling the original problem is introduced, incorporating hybrid division rules. The convergence of this algorithm is demonstrated. Ultimately, numerical experiments validate the feasibility and efficacy of the proposed algorithm.**

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

I. Introduction

FRACTIONAL programming represents a type of nonlinear optimization problem that is prevalent in the domain of optimization. Its objective is to determine the optimal fraction value under a set of specified constraints. The problem of sum-of-linear-ratios represents a particular instance, characterized by an objective function formulated as the summation of ratios. Given its classification as an NP-hard problem [1], multiple local optimum solutions may exist, so it is difficult in theoretical research and algorithm implementation. Sum-of-linear-ratios problems are widely used in financial economy [2], [3], transportation [4], and other problems. Due to its wide application in many fields and the growing problem demand, this field still attracts the interest of researchers and promotes its continuous development.

The primary focus of this paper is on the aforementioned sum-of-linear-ratios problem (SLR) :

Manuscript received June 5, 2024; revised November 5, 2024.

This work was supported by the National Natural Science Foundation of China (Project Number:22XGL019).

Xianfeng Ding is a professor at the School of Science, Southwest Petroleum University, Chengdu, 610500, China (Corresponding author: e-mail:dingxianfeng@swpu.edu.cn).

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

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

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

$$(SLR): \begin{cases} min \quad f(x) = \sum_{i=1}^{p} \frac{P_i(x)}{D_i(x)} \\ s.t. \qquad Ax \le b, x \ge 0. \end{cases}$$

in this context, $P_i(x)$ and $D_i(x)$ represent linear functions defined over the space R^n and

$$P_i(x) = p_i^T x + \alpha_i = \sum_{j=1}^{n} p_{ij} x_j + \alpha_i,$$

$$D_i(x) = {d_i}^T x + \beta_i = \sum_{j=1}^{n} d_{ij} x_j + \beta_i.$$

Let $S_0 = \{x \in R^n | Ax \leqslant b, x \geqslant 0\}$ be a nonempty bounded set, $A \in R^{m \times n}, b \in R^m$. And $p_i, d_i \in R^n, \alpha_i, \beta_i, p_{ij}, d_{ij} \in R, i = 1, 2, \cdots, p$, in general $p \ge 2$.

When $d_i^T x + \beta_i < 0$, we can set $\frac{p_i^T x + \alpha_i}{d_i^T x + \beta_i} = \frac{-\left(p_i^T x + \alpha_i\right)}{-\left(d_i^T x + \beta_i\right)}$, then the denominator becomes positive, and the original problem remains unchanged. Therefore, assuming $D_i(x) > 0$.

For any function $p_i^T x + \alpha_i, \frac{p_i^T x + \alpha_i + M\left(d_i^T x + \beta_i\right)}{d_i^T x + \beta_i}$ can be constructed, where M is a number large enough to satisfy $p_i^T x + \alpha_i + M\left(d_i^T x + \beta_i\right) > 0$.

So in this paper, we stipulate that for $\forall i = 1, 2, \cdots, p$, there are $P_i(x) \ge 0, D_i(x) > 0$.

Currently, a diverse range of global optimization algorithms exists for tackling sum-of-linear-ratios problems, such as parameter iteration method [5], image space method [6], various types of branch-and-bound methodologies [7]–[10], monotone optimization method [11], dual method [12], and region segmentation algorithm [13], etc.

In 2017, Li [14] introduced a branch-and-bound algorithm tailored for the programming problem involving the sum of linear fractions. Initially, the original problem underwent an equivalent transformation through the introduction of variables, and then relaxation technology was used to construct a relaxed linear programming of equivalent problems. Through the resolution of a sequence of linear programming problems, the upper and lower bounds of the optimal values of the original problem were constantly updated. Finally, the approximate optimal solution of the original problem is obtained. In 2019, aiming at the linear fraction multiproduct programming problem, Shen [15] introduced an algorithm to attain the global optimal solution of the original problem through Charnes-Cooper transformation to obtain the equivalent problem. In the same year, Shen [16] proposed a new equivalent transformation and contraction strategy, which obtained the solution of the original problem by tackling a series of standard geometric programming problems, and numerical experiments showed the traceability and effectiveness of the algorithm. In 2021, Wang [17] introduced a branch-and-bound method for solving the global problem by integrating convex relaxation techniques, adaptive branching rules, and acceleration strategies. This paper proposes an improved branch-and-bound algorithm, building upon both traditional algorithms and novel fractional programming techniques.

This paper introduces an innovative global optimization algorithm specifically designed for the sum-of-linear-ratios (SLR) problem. To achieve the global optimal solution, the first step involves transforming the (SLR) problem into an equivalent formulation. Next, leveraging the linear approximation properties of bilinear functions, the relaxation of the objective function and constraints for the equivalent problem is carried out to guarantee that it provides a lower bound for the equivalent problem. Additionally, we introduce a hybrid branch-and-bound algorithm that combines radiant subdivision and dichotomy, along with a convergence proof for the proposed method. Finally, numerical experiments demonstrate the algorithm's high efficiency, while two practical applications highlight its effectiveness.

The structure of the subsequent sections of this paper is organized as follows: The second and third sections focus on transforming the original problem into its equivalent form using techniques of transformation and relaxation. The fourth section presents an improved branch-and-bound algorithm, along with a proof of its convergence. In the fifth section, numerical experiments are conducted, and the results are subsequently compared. The sixth section provides two real-world examples to illustrate the efficacy of the introduced algorithm. The seventh section is the conclusion.

II. Equivalent problem

Prior to addressing the sum-of-linear-ratios problem, the initial problem, denoted as (SLR), undergoes a transformation process, which comprises two primary stages. The first stage involves converting the original (SLR) problem into an equivalent problem, termed (EP), through the application of transformation methodologies. The second stage entails transforming the equivalent problem (EP) into a linear relaxation problem, designated as (RLP), by utilizing relaxation techniques.

Without compromising generality, for $\forall i = 1, 2, \cdots p$, let

$$l_i^0 = min_{x\in S_0} P_i(x), u_i^0 = max_{x\in S_0} P_i(x),$$
$$L_i^0 = \frac{1}{max_{x\in S_0} D_i(x)}, U_i^0 = \frac{1}{min_{x\in S_0} D_i(x)}.$$

Because $P_i(x)$ and $D_i(x)$ are functions with finite bounds within the set S_0. By solving the linear programming problems $min_{x\in S_0} P_i(x)$, $max_{x\in S_0} P_i(x)$, $max_{x\in S_0} D_i(x)$, and $min_{x\in S_0} D_i(x)$, the values of l_i^0, u_i^0, L_i^0 and U_i^0 are obtained.

Consequently, the resulting (SLR) can be formulated into its equivalent problem, denoted as (EP), as outlined below:

$$(EP): \begin{cases} min & g(y,z) = \sum_{i=1}^{p} y_i z_i \\ s.t. & y_i = P_i(x), i = 1, 2, \cdots, p, \\ & z_i D_i(x) = 1, i = 1, 2, \cdots, p, \\ & x \in S_0, (y,z) \in \Omega^0. \end{cases}$$

where $\Omega^0 = \{(y,z) \in R^{2p} | l_i^0 \le y_i \le u_i^0, L_i^0 \le z_i \le U_i^0\}$, $i = 1, 2, \cdots, p$.

Theorem 1: If (x^*, y^*, z^*) represents the global optimal solution to the problem (EP), we define $y_i^* = P_i^*(x), z_i^* = \frac{1}{D_i^*(x)}, i = 1, 2, \cdots, p$. Under these conditions, x^* is the global optimal solution to the problem (SLR). Conversely, if x^* is the global optimal solution to the problem (SLR), then (x^*, y^*, z^*) is also the global optimal solution to (EP).

Proof: Let (x^*, y^*, z^*) be the global optimal solution to the problem (EP), then we will prove x^* to be the global optimal solution to the problem (SLR).

Utilizing a proof by contradiction, if x^* fails to be the globally optimal solution for the problem (SLR), then there necessarily exists a feasible solution $\overline{x}$ to (SLR) that fulfills the condition $f(\overline{x}) < f(x^*)$, whereby

$$\overline{y}_i = P(\overline{x}), \overline{z}_i = \frac{1}{D_i(\overline{x})}, i = 1, 2, \cdots, p.$$

Then $(\overline{x}, \overline{y}, \overline{z})$ constitutes a feasible solution to the problem denoted as (EP). Because $f(\overline{x}) < f(x^*)$, we can get

$$\sum_{i=1}^{p} \overline{y}_i \overline{z}_i = \sum_{i=1}^{p} \frac{P_i(\overline{x})}{D_i(\overline{x})} < \sum_{i=1}^{p} \frac{P_i(x^*)}{D_i(x^*)}.$$

Furthermore, given that (x^*, y^*, z^*) satisfies the requirements for the problem designated as (EP), there are

$$y_i^* = P_i(x^*), z_i^* = \frac{1}{D_i(x^*)}, i = 1, 2, \cdots, p.$$

So

$$\sum_{i=1}^{p} \overline{y}_i \overline{z}_i = \sum_{i=1}^{p} \frac{P_i(\overline{x})}{D_i(\overline{x})} < \sum_{i=1}^{p} \frac{P_i(x^*)}{D_i(x^*)} = \sum_{i=1}^{p} y_i^* z_i^*.$$

This is inconsistent with the notion that (x^*, y^*, z^*) represents the global optimal solution to the problem (EP). Consequently, x^* stands as the globally optimal solution for the problem designated as (SLR).

Conversely, consider the scenario where x^* serves as the global optimal solution to the problem (SLR), and let

$$y_i^* = P_i(x^*), z_i^* = \frac{1}{D_i(x^*)}, i = 1, 2, \cdots, p.$$

that is, (x^*, y^*, z^*) forms a feasible solution for the problem (EP). Suppose there exists another feasible solution $(\overline{x}, \overline{y}, \overline{z})$ for (EP) that satisfies

$$\sum_{i=1}^{p} \overline{y}_i \overline{z}_i < \sum_{i=1}^{p} y_i^* z_i^*.$$

Furthermore, let $\overline{x}$ be a feasible solution for the problem (SLR). It then follows that $f(\overline{x}) < f(x^*)$. This contradicts our initial assumption that x^* is the global optimal solution to problem (SLR). Proof complete. ■

III. Linear relaxation technique

By leveraging the unique characteristics of the objective and constraint functions in the equivalent problem (EP), we develop a linear relaxation program for (EP). This transformation results in the relaxation problem (RLP), where the optimal solution of (RLP) serves as a feasible solution for the problem (SLR).

Let Ω represent Ω^0 or a subrectangle of Ω^0 generated during a branch and bound search, in other words, $\Omega \subseteq \Omega^0$ and $\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_p$, where $\Omega_i = \left\{(y_i, z_i) \in R^2 | l_i \le y_i \le u_i, L_i \le z_i \le U_i\right\}$, $\left[l_i, u_i\right] \subseteq \left[l_i^0, u_i^0\right]$, $\left[L_i, U_i\right] \subseteq \left[L_i^0, U_i^0\right]$, $i = 1, 2, \cdots, p$.

First of all, in Ω_i, there is $y_i - l_i \ge 0$, $z_i - U_i \le 0$, then we can get $(y_i - l_i)(z_i - U_i) \le 0$, expansion gives: $y_i z_i - y_i U_i - l_i z_i + l_i U_i \le 0$, that is

$$y_i z_i \le y_i U_i + l_i z_i - l_i U_i, i = 1, 2, \cdots, p. \tag{1}$$

Similarly, we can get $y_i - u_i \le 0$, $z_i - L_i \ge 0$ in Ω_i, then $(y_i - u_i)(z_i - L_i) \le 0$ is true, and expand to get $y_i z_i \le y_i L_i + u_i z_i - u_i L_i, i = 1, 2, \cdots, p$.

Let $e_i^1 = y_i U_i + l_i z_i - l_i U_i$, $e_i^2 = y_i L_i + u_i z_i - u_i L_i$, then

$$e_i^u(y_i, z_i) = min\left\{e_i^1(y_i, z_i), e_i^2(y_i, z_i)\right\}. \tag{2}$$

On account of $y_i z_i \le e_i^1(y_i, z_i)$, $y_i z_i \le e_i^2(y_i, z_i)$, then we can get $y_i z_i \le e_i^u(y_i, z_i)$.

In the same way, it can be established that $(y_i - l_i)(z_i - L_i) \ge 0$ is in Ω_i, and it can be expanded separately to get:

$$y_i z_i \ge y_i L_i + l_i z_i - l_i L_i,$$
$$y_i z_i \ge y_i U_i + u_i z_i - u_i U_i.$$

Let $e_i^3 = y_i L_i + l_i z_i - l_i L_i$, $e_i^4 = y_i U_i + u_i z_i - u_i U_i$,

$$e_i^l(y_i, z_i) = max\left\{e_i^3(y_i, z_i), e_i^4(y_i, z_i)\right\}, \tag{3}$$

because of $y_i z_i \ge e_i^3(y_i, z_i)$, $y_i z_i \ge e_i^4(y_i, z_i)$, then we can get $y_i z_i \ge e_i^l(y_i, z_i)$.

In summary can be obtained

$$e_i^l(y_i, z_i) \le y_i z_i \le e_i^u(y_i, z_i). \tag{4}$$

For $\forall z_i \in \left[L_i, U_i\right]$, because of $z_i = \frac{1}{D_i(x)}$, there is $\frac{1}{U_i} \le D_i(x) \le \frac{1}{L_i}$. For $\forall D_i(x) \in \left[\frac{1}{U_i}, \frac{1}{L_i}\right]$, let

$$\Phi_i(x, z_i) = z_i D_i(x) = 1.$$

A structure similar to formula (1) can be obtained

$$\Phi_i^u(x, z_i) = min\left\{U_i D_i(x) + \frac{z_i}{U_i} - 1, L_i D_i(x) + \frac{z_i}{L_i} - 1\right\},$$

$$\Phi_i^l(x, z_i) = max\left\{L_i D_i(x) + \frac{z_i}{U_i} - \frac{L_i}{U_i}, U_i D_i(x) + \frac{z_i}{L_i} - \frac{U_i}{L_i}\right\}.$$

Same thing as formula (4), for $\forall D_i(x) \in \left[\frac{1}{U_i}, \frac{1}{L_i}\right]$, $z_i \in \left[L_i, U_i\right]$, there is

$$\Phi_i^l(x, z_i) \le \Phi_i(x, z_i) \le \Phi_i^u(x, z_i). \tag{5}$$

From equations (4) and (5), for $\forall \Omega \subseteq \Omega^0$, the linear relaxation programming problem (RLP) is established on the region Ω as follows:

$$(RLP): \begin{cases} min & G(y, z) = \sum_{i=1}^{p} r_i \\ s.t. & r_i \ge y_i L_i + l_i z_i - l_i L_i, i = 1, 2, \cdots, p, \\ & r_i \ge y_i U_i + u_i z_i - u_i U_i, i = 1, 2, \cdots, p, \\ & y_i = P_i(x), i = 1, 2, \cdots, p, \\ & L_i D_i(x) + \frac{z_i}{U_i} - \frac{L_i}{U_i} \le 1, i = 1, 2, \cdots, p, \\ & U_i D_i(x) + \frac{z_i}{L_i} - \frac{U_i}{L_i} \le 1, i = 1, 2, \cdots, p, \\ & U_i D_i(x) + \frac{z_i}{U_i} - 1 \ge 1, i = 1, 2, \cdots, p, \\ & L_i D_i(x) + \frac{z_i}{L_i} - 1 \ge 1, i = 1, 2, \cdots, p, \\ & x \in S_0, \\ & (y, z) \in \Omega. \end{cases}$$

Based on the aforementioned methodology for constructing the linear relaxation programming problem, it is straightforward to deduce that for $\forall \Omega \subseteq \Omega^0$, every feasible solution of the problem (EP) is also a feasible solution of the problem (RLP). Furthermore, the optimal value of the problem (RLP) offers a valid lower bound on the region of the problem (EP).

IV. Algorithm and its convergence

A. Hybrid division rules

Radiant subdivision is an important subdivision technique of simplex subdivision [18]. Multiple subsimplexes will be produced in one subdivision, and radial subdivision will make full use of the optimal information obtained from the current calculation, which may converge to the optimal point more quickly.

Definition 1: Given a n-simplex (S), the vertex set is $v(S) = \left\{v_0, v_1, \cdots, v_n\right\}$. If we choose a point $\omega \in S$ and $\omega \notin v(S)$, it can be uniquely represented as

$$\omega = \sum_{i=0}^{n} \lambda_i v_i, \lambda_i \ge 0, (i = 0, 1, \cdots, n), \sum_{i=0}^{n} \lambda_i = 1.$$

For every i that makes $\lambda_i > 0$, replace the vertex v_i with ω and construct the simplex $S(i, \omega)$, that is

$$S(i, \omega) = co\left\{v_0, \cdots, v_{i-1}, \omega, v_{i+1}, \cdots, v_n\right\}.$$

This subdivision is called radial subdivision, also known as ω subdivision.

Definition 2: A n-simplex

$$S = \left[v_0, \cdots, v_i, \cdots, v_j, \cdots, v_n\right]$$

is defined by dividing it into two simplex S_1, S_2 based on the midpoint v of the longest side $\left[v_i, v_j\right]$ of S. Use point v to replace point v_i and point v_j, respectively. Among

$$S_1 = \left[v_0, \cdots, v, \cdots, v_j, \cdots, v_n\right],$$

$$S_2 = \left[v_0, \cdots, v_i, \cdots, v, \cdots, v_n\right],$$

$$v = \frac{1}{2}(v_i + v_j).$$

we call this subdivision dichotomy.

S_k is subdivided radially by point $\omega(S_k)$. For each simplex after division, check whether it meets the convergence condition of division.

For a simplex S_k^i that does not satisfy the convergence condition of division, it is divided on the basis of the midpoint of the longest edge of S_k^i until the convergence condition is satisfied.

By the above division rules, the simplex S completes the division.

Define Q_k as the collection of simplices potentially containing global optimal solutions during the k_{th} iteration. For each simplex $\overline{S} \in Q_k$, the optimal value of the relaxation problem (RLP) establishes a lower bound $G_k(y, z)$ for the optimal value of the problem (SLR). A simplex with a lesser optimal value is chosen and then divided using either ω subdivision or dichotomy. The corresponding solution for the problem is determined on each newly formed subsimplex, and this process is iterated until the convergence condition is satisfied.

B. Improved branch-and-bound algorithm

Presented below is the branch-and-bound algorithm employing a hybrid division approach:

$Step0$ (Initialization): Given the error tolerance $\epsilon \geq 0$, the optimal solution for solving the relaxation problem (RLP) on the feasible domain S_0 is $\left(x^0, y^0, z^0\right)$, and the optimal value is $G\left(y^0, z^0\right)$, where

$$y_i^0 = p_i^T x^0 + \alpha_i, z_i^0 = \frac{1}{d_i^T x^0 + \beta_i} \left(i = 1, 2, \cdots, p\right),$$

let $\mu_0 = G\left(y^0, z^0\right), \gamma_0 = f\left(x^0\right);$

$Step1$ (Termination): If $\gamma_0 - \mu_0 < \epsilon$, then the algorithm terminates and x^0 is the global optimal solution to the problem (SLR); Otherwise, let $Q_0 = \left\{S_0\right\}$, choose $S_k \in Q_0$ so that $\mu\left(S_k\right) = \mu_k$, and the corresponding optimal point $\omega\left(S_k\right)$, let $\gamma = \gamma_k$.

$Step2$: Initial iteration number $k = 1$, turn to step k;

$Stepk$: $k \geq 1$;

$Stepk_1$ (Branch): According to the division rules, the simplex S_k with a smaller optimal value is selected for dividing, and the S_k is subdivided radially according to the point $\omega\left(S_k\right)$, where

$$\omega\left(S_k\right) = \lambda_{k0} v_{k0} + \cdots + \lambda_{kj} v_{kj} + \cdots + \lambda_{kn} v_{kn}.$$

Divide S_k into t (the number of positive λ_{kj}) simplexes, where

$$S_{kj} = \left[v_{k0}, \cdots, v_{k(j-1)}, \omega\left(S_k\right), v_{k(j+1)}, \cdots, v_{kn}\right] \in \Gamma,$$

if and only if $\lambda_{kj} > 0$. For each simplex in Γ, solve for $\mu\left(S_{kj}\right)$ and the corresponding point $\omega\left(S_{kj}\right), j = 1, \cdots, t$, while updating the value of γ. Check whether each simplex in Γ satisfies the division convergence condition

$$\gamma - \mu\left(S_{kj}\right) < \left(1 - \epsilon\right)\left(\gamma_k - \mu_k\right).$$

For a simplex S_{kj} in Γ that does not satisfy the convergence condition, it is divided on the basis of the midpoint of the longest side of S_{kj} until $\gamma - \mu\left(S_{kj}\right) < \left(1 - \epsilon\right)\left(\gamma_k - \mu_k\right)$ is satisfied, Γ and γ are updated simultaneously. The set of new simplexes potentially containing global optimal solutions is denoted by $Q_k = Q_k \setminus S_k$.

$Stepk_2$ (Delimiting and cutting branches): For a subsimplex S_{kj} in Γ, if $S_{kj} \neq \emptyset$, then let

$$y_i^{kj} = p_i^T x^{kj} + \alpha_i, z_i^{kj} = \frac{1}{d_i^T x^{kj} + \beta_i}, i = 1, 2, \cdots, p.$$

Determine the optimal solution $\left(x^{kj}, y^{kj}, z^{kj}\right)$ and the optimal value $\mu_k = G\left(y^{kj}, z^{kj}\right)$ of the problem (RLP) on S_{kj}, and if $\gamma_k < \mu_k$, then $\Gamma = \Gamma \setminus \left\{S_{kj}\right\}$, there are two cases:

Case1: If $\Gamma = \emptyset$, go to step k;

Case2: If $\Gamma \neq \emptyset$, update Q_k by setting $Q_k = Q_k \cup \Gamma$, and revise the upper bound $\gamma_k = min\left\{\gamma_{k-1}, f\left(x^{kj}\right)\right\}$. Choose x^k such that $\gamma_k = f\left(x^k\right)$, go to step k_3;

$Stepk_3$ (Judgment rule): Let

$$Q_{k+1} = Q_k \setminus \left\{\overline{S} : \gamma_k\left(\overline{S}\right) - \mu_k \leq \epsilon, \overline{S} \in Q_k\right\}$$

if $Q_{k+1} = \emptyset$, then the algorithm terminates, with x^k being a global optimal solution to the problem (SLR) and γ_k being the global optimal value. Otherwise, if $Q_{k+1} \neq \emptyset$, let $k = k + 1$, select S_k to satisfy $S_k = arg\min_{\overline{S} \in Q_k} \mu_k\left(\overline{S}\right)$, and return to step k.

C. Algorithm and its convergence

Theorem 2: If the iteration process concludes within a finite number of steps, the global optimal solution to the problem (SLR) is attained at the iteration's termination. Conversely, if the algorithm produces an infinite sequence $\left\{x^k\right\}$ during the iterative process, every accumulation point arising from the sequence $\left\{x^k\right\}$ represents the global optimal solution to the problem (SLR), Furthermore, it holds that $\lim_{k\to\infty} \mu_k = \lim_{k\to\infty} f\left(x^k\right) = \lim_{k\to\infty} \gamma_k$.

Proof: If the algorithm's iteration concludes at a finite step, suppose this occurs at step $k\left(k \geq 1\right)$. Consider $\left(x^*, y^*, z^*\right)$ as the optimal solution to the relaxation problem (RLP). Additionally, let

$$T^0 = \left\{y \in R^p | l_i \leq y_i \leq u_i, i = 1, 2, \cdots, p\right\},$$
$$H^0 = \left\{z \in R^p | L_i \leq z_i \leq U_i, i = 1, 2, \cdots, p\right\}.$$

Given $T \subseteq T^0$ and $H \subseteq H^0$, consider the optimal solution expressed as:

$$x^*, y_i^* = \sum_{j=1}^{n} p_{ij} x_j^* + \alpha_i, z_i^* = \frac{1}{\sum_{j=1}^{n} d_{ij} x_j^* + \beta_i}, i = 1, 2, \cdots, p,$$

it is evident that the optimal solution x^* for the problem (RLP) constitutes a feasible solution of the problem (SLR). Moreover, if v represents an optimal value of the problem (SLR), then it necessarily follows that $f\left(x^*\right) \geq v$.

Upon satisfying the condition $\gamma_k - \mu_k \leq \epsilon$, the algorithm halts its execution. The upper limit of the problem (SLR) is revised by updating the functional value associated with a feasible solution of (SLR). Subsequently, the algorithm is employed to derive:

$$f\left(x^*\right) - \mu_k \leq \epsilon, \mu_k \leq v.$$

So in sum, we derive the inequality $v \leq f\left(x^*\right) \leq \mu_k + \epsilon$, which further implies $\mu_k + \epsilon \leq v + \epsilon$. Combining these, we obtain:

$$v \leq f\left(x^*\right) \leq v + \epsilon.$$

If the algorithm's iteration proceeds indefinitely, we assume it generates an infinite sequence of branch-and-bound trees denoted by $\left\{\left(x^k, y^k, z^k\right)\right\}$. For each $k \geq 1$, $\left(x^k, y^k, z^k\right)$ can be obtained by solving the relaxation problem (RLP), for $T^k \subseteq T^0, H^k \subseteq H^0$. Specifically, the optimal solution x^k belongs to S_0, with $y_i^k = \sum_{j=1}^{n} p_{ij} x_j^k + \alpha_i$ and $z_i^k = \frac{1}{\sum_{j=1}^{n} d_{ij} x_j^k + \beta_i}, i = 1, 2, \cdots, p$. It is obvious that sequences $\left\{x^k\right\}$ constitute feasible solutions to the problem (SLR). Assuming $\overline{x}$ is the accumulation point of $\left\{x^k\right\}$ such that $\lim_{k\to\infty} x^k = \overline{x}$. Clearly, $\overline{x}$ is also a feasible solution to the problem (SLR). Consequently, $f\left(\overline{x}\right) \geq v$. Given that S_0 is a compact set, it necessarily follows that $\overline{x} \in S_0$.

For every k of the sequence $\left\{x^k\right\}$, there exists a relationship such that $T^{k+1} \subseteq T^k \subseteq T^0$ and $H^{k+1} \subseteq H^k \subseteq H^0$, where T^k and H^k are defined as follows:

$$T^k = \left\{y \in R^p | l_i^k \leq y_i \leq u_i^k, i = 1, 2, \cdots, p\right\},$$
$$H^k = \left\{z \in R^p | L_i^k \leq z_i \leq U_i^k, i = 1, 2, \cdots, p\right\}.$$

Then for some point $\overline{y} \in R^p, \overline{z} \in R^p$, the expression $\lim_{k\to\infty} T^k = \bigcap\limits_k T^k = \{\overline{y}\}$, $\lim_{k\to\infty} H^k = \bigcap\limits_k H^k = \{\overline{z}\}$ is true.

For each k, $\{\mu_k\}$ obtained through the algorithmic step is finite and $\lim_{k\to\infty} \mu_k \le v$. For $T^k \subseteq T^0, H^k \subseteq H^0$, x^k is the optimal solution to the problem $RLP\big(T^k, H^k\big)$, with μ_k being its optimal value. Consequently, as k approaches infinity, the limits of the lower and upper bounds converge:

$$\lim_{k\to\infty} l^k = \lim_{k\to\infty} u^k = \{\overline{y}\},$$

$$\lim_{k\to\infty} L^k = \lim_{k\to\infty} U^k = \{\overline{z}\}.$$

To sum up:

$$\lim_{k\to\infty} \mu_k \le v \le f(\overline{x}).$$

Through the application of the division rule during branching, we derive:

$$\lim_{k\to\infty} \mu_k = v = f(\overline{x}).$$

Thus, $\overline{x}$ is identified as a global optimal solution of the problem (SLR). During the iterative process of the algorithm, $\gamma_k = f\left(x^k\right)$ when iterating at step k. And $f(x)$ is a continuous function, so $x^k \to \overline{x}$ when $k \to \infty$, then $\lim_{k\to\infty} f\left(x^k\right) = f(\overline{x})$. So $\lim_{k\to\infty}\gamma_k = \lim_{k\to\infty} f\left(x^k\right) = f(\overline{x}) = v$, that is $\lim_{k\to\infty}\gamma_k = v$.

To sum up, the theorem is proved. ■

V. Numerical experiments and results

To assess the viability of the aforementioned algorithm, some examples are presented to evaluate its efficacy and performance. The above algorithm is compiled and the code is run on $Matlab\,(2018b)$. All calculations are performed on a $Ryzen$ 7 $7840HS\ w/Raden$ $780M\ Graphics$ 3.80 GHz processor with $32GB\ RAM$ and a $Win11$ operating system PC.

Example 1: (See [19])

$$max \frac{-0.9x_1 + 1.8x_2 + 1.8}{3x_1 - 4x_2 + 5} + \frac{-0.4x_1 + 0.3x_2 - 0.4}{-2x_1 + x_2 + 3}$$

$$\begin{aligned} s.t.\quad & x_1 + x_2 \le 1.50, \\ & x_1 - x_2 \le 0, \\ & 0 \le x_1, x_2 \le 1. \end{aligned}$$

Example 2: (See [14], [20])

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

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

Example 3: (See [14], [20])

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

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

Example 4: (See [8], [13], [19], [21])

$$max \frac{3x_1 + x_2 - 2x_3 + 0.8}{2x_1 - x_2 + x_3} + \frac{4x_1 - 2x_2 + x_3}{7x_1 + 3x_2 - x_3}$$

$$\begin{aligned} s.t.\quad & x_1 + x_2 - x_3 \le 1, \\ & -x_1 + x_2 - x_3 \le -1, \\ & 12x_1 + 5x_2 + 12x_3 \le 34.8, \\ & 12x_1 + 12x_2 + 7x_3 \le 29.1, \\ & -6x_1 + x_2 + x_3 \le -4.1, \\ & x_1, x_2, x_3 \ge 0. \end{aligned}$$

Example 5: (See [7], [10], [13], [22])

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

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

Example 6: (See [13], [23], [24])

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

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

Example 7: (See [10], [22], [25])

$$max \frac{37x_1 + 73x_2 + 13}{13x_1 + 13x_2 + 13} + \frac{63x_1 - 18x_2 + 39}{-13x_1 - 26x_2 - 13} + \frac{13x_1 + 13x_2 + 13}{63x_1 - 18x_2 + 39} + \frac{x_1 + 2x_2 + 4x_3 + 50}{5x_2 + 4x_3 + 50}$$

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

Example 8: (See [10])

$$min \sum_{j=1}^{p} \frac{\sum_{i=1}^{n} u_j^i x_i + C}{\sum_{i=1}^{n} v_j^i x_i + C}$$

$$\begin{aligned} s.t.\quad & \sum_{i=1}^{n} a_{qi} x_i \le b_q, q = 1, 2, \cdots, m, \\ & 0 \le x_i \le \overline{x}_i, i = 1, 2, \cdots, n. \end{aligned}$$

TABLE I
COMPARISON OF NUMERICAL RESULTS OF EXAMPLE 1-7

Ex.	Refs.	ϵ	x^*	$f(x^*)$	Iter	Time(seconds)
1	ours	10^{-6}	$(0,1)$	3.5750	2	0.8285
1	[19]	10^{-6}	$(0,1)$	3.5750	1	-
2	ours	10^{-4}	$(1.1111,0,0)$	4.0907	2	0.1586
2	[14]	10^{-4}	$(5,0,0)$	4.4286	3	0.4050
2	[20]	10^{-4}	$(0,0.6250,1.8750)$	4.0000	58	2.9684
3	ours	10^{-4}	$(0,1.4267,0)$	3.0007	4	0.9318
3	[14]	10^{-4}	$(0,1.6667,0)$	3.0009	64	7.4100
3	[20]	10^{-4}	$(0,3.3333,0)$	3.0029	80	8.5665
4	ours	10^{-9}	$(1,0,0)$	2.4714	1	0.0928
4	[8]	10^{-9}	$(1,0,0)$	2.4714	2	0.1013
4	[13]	10^{-2}	$(1,0,0)$	2.4714	19	0.5269
4	[19]	10^{-2}	$(1,0,0)$	2.4712	2	0.0109
4	[21]	10^{-2}	$(1,0,0)$	2.4714	3	0.0128
5	ours	10^{-6}	$(1.1111,0,0)$	-4.0907	2	0.1110
5	[7]	10^{-6}	$(1.1111,0,0)$	-4.0907	72	2.5632
5	[10]	10^{-6}	$(1.1111,0,0)$	-4.0907	154	4.9857
5	[13]	10^{-2}	$(1.1111,0,0)$	-4.0907	179	2.1459
5	[22]	10^{-6}	$(1.1111,0,0)$	-4.0907	769	11.5349
6	ours	10^{-4}	$(1.5000,1.5000)$	4.9125	2	0.2664
6	[13]	10^{-3}	$(1.5000,1.5000)$	4.9126	56	1.0781
6	[23]	10^{-4}	$(1.5000,1.5000)$	5.0000	32	1.0895
6	[24]	10^{-4}	$(1.5000,1.5000)$	4.9125	112	201.6260
7	ours	10^{-6}	$(3,4.1919)$	3.4140	1	0.2502
7	[10]	10^{-6}	$(3,4)$	3.2917	5	0.2207
7	[22]	10^{-6}	$(3,4)$	3.2917	2	0.0023
7	[25]	10^{-6}	$(3,4)$	3.2917	9	0

Where p is a positive integer, u_j^i, v_j^i, a_{qi}, b_q, and $\overline{x}_i$ in this example are all randomly generated on the interval $\left[0.01,\delta\right]$ (parameter δ is 1, 10 or 100), except that the value of C is taken as 100.

The calculation results of Example 1-7 have been shown in TableI. According to different error tolerance ϵ, the optimal solution and optimal value of Example 1-7 within the accuracy range have been determined. Additionally, Table I provides a summary of the iteration times and running time of the algorithm for reference purposes.

The "-" in the above table represents that the corresponding results are not given in the references. It can be seen from TableI that: for Example1, Example4, Example5 and Example6, the algorithm in this paper is consistent with the optimal solutions and values obtained by other algorithms, and basically all use less iteration times and running time; In the case of Example2 and Example3, the optimal solution and value derived by this algorithm align closely with those obtained by other methods. However, our algorithm demonstrates superiority in terms of iteration times and running time. In Example 3, our algorithm requires significantly fewer iterations compared to the other two methods. In contrast, for Example 7, the numerical results and iteration times presented in this paper surpass those of other algorithms, albeit with a marginally longer running time. In general, the algorithm in this paper is feasible and effective, and its computing power is comparable to that of Example1, Example4, Example5, and Example6, and at least better than that of Example7.

The practical application of the proposed algorithm is illustrated through the examples provided in Examples 1 to 7. Additionally, TableI clearly shows that the iteration times and the running time align with the fundamental observations. To better demonstrate the merits of the introduced algorithm, two ways are utilized to showcase the performance of the proposed algorithm for Example8. Method 1 compares the proposed algorithm with reference [10], which is consistent with the iteration times and error tolerance in reference [10]. Run the algorithm at random 100 times, set $\delta = 1$, and the error tolerance of 10^{-6}. The average iteration time and average iteration times obtained are shown in TableII.

It can be seen from the data in TableII that with the change of parameter p, m, n, the iteration times in this paper is all smaller than that found in the reference [10], taking an absolute advantage.

However, the algorithm presented in this paper exhibits a relatively prolonged running time, which escalates markedly with a steep increase in n. Operating within the R^{2p}-dimensional division space, the algorithm's iterations and running time are increasingly influenced by variations in p. Therefore, the impact of the change of δ and n on the performance of the algorithm will be investigated next. Method 2 takes reference [10] as the basis and adjusts δ to 10 or 100. When the tolerance is 10^{-3}, the algorithm proposed is contrasted with the large-scale solution in the reference [10]. For each group (p, m, n), 10 examples are randomly generated, and the relevant numerical results are obtained, as shown in TableIII.

As shown in TableIII, when $\delta = 10$, the iteration times and the running time of this paper exceed those of reference [10]; however, when δ increases to 100, the iteration times and the running time of this algorithm are less than that of the reference [10], and the changes are relatively stable. The hybrid division rule is applied in this algorithm, and multiple subsimplexes will be generated during the dividing. By making full use of the known information obtained from the calculation, the algorithm in this paper demonstrates certain advantages when applied to large-scale numerical experiments.

In summary, the numerical cases demonstrate the robust-

TABLE II
COMPARISON OF NUMERICAL RESULTS OF EXAMPLE 8 SOLVED BY THE ALGORITHM IN THIS PAPER AND THE ALGORITHM IN REFERENCE [10] WHEN $\delta = 1$

	Ours		[10]	
(p, m, n)	Avg.Iter	Avg.Time	Avg.Iter	Avg.Time
$(2, 20, 20)$	2.93	0.4261	3.55	0.1025
$(2, 30, 30)$	3.13	0.5407	3.76	0.1254
$(2, 40, 40)$	3, 27	0.6509	3.71	0.1461
$(2, 50, 50)$	3.40	0.6938	4.04	0.1807
$(2, 10, 100)$	3.70	1.3176	4.89	0.2032
$(2, 30, 100)$	4.07	1.7881	4.81	0.2432
$(2, 10, 200)$	4.26	2.6666	4.79	0.2914
$(3, 20, 20)$	3.01	0.4432	5.11	0.1658
$(3, 30, 30)$	3.20	0.5916	5.34	0.2058
$(3, 40, 40)$	3.56	0.6487	5.26	0.2349
$(3, 50, 50)$	3.35	0.7251	5.40	0.2782
$(3, 10, 100)$	4.01	1.4484	6.75	0.3321
$(3, 30, 100)$	4.40	1.7959	6.47	0.3768
$(3, 10, 200)$	4.42	3.1653	7.28	0.4990
$(4, 20, 20)$	3.68	0.5313	7.43	0.2615
$(4, 30, 30)$	3.76	0.6262	7.21	0.3347
$(4, 40, 40)$	4.36	0.7219	7.03	0.3283
$(4, 50, 50)$	4.40	0.9021	7.69	0.4101
$(4, 10, 100)$	4.65	1.6038	8.58	0.4321
$(4, 30, 100)$	4.76	2.0465	8.67	0.5470
$(4, 10, 200)$	4.73	2.7660	8.60	0.6344
$(5, 10, 100)$	4.72	2.1204	11.34	0.6146
$(5, 10, 200)$	4.80	3.4643	13.84	1.0281
$(5, 10, 300)$	4.80	7.1198	15.99	1.5051
$(6, 10, 100)$	4.80	2.2450	15.25	0.8519
$(6, 10, 200)$	5.07	3.6898	19.75	1.5526
$(6, 10, 300)$	5.20	7.2014	20.69	1.8879
$(7, 10, 100)$	5.25	2.4445	19.26	1.1569
$(7, 10, 200)$	5.41	4.0131	21.85	1.8085
$(7, 10, 300)$	6.20	13.1893	29.78	3.0462

ness and effectiveness of the proposed algorithm in this paper, and compared to some other algorithms, it offers notable advantages.

VI. PRACTICAL APPLICATIONS

A. Transportation problem

In our analysis, we examine the power transportation issue outlined in [26]. Assume a power company possesses three power stations tasked with fulfilling the electrical demands of four municipalities. For each conceivable route of electricity transmission, we introduce the variable x_{ij} to denote the quantity of electricity dispatched from the i-th plant to the j-th city. Utilizing the data provided in [26], we have obtained the mathematical model of the power transmission problem as detailed below:

$$max\frac{5x_{11}+4x_{12}+4x_{13}+3x_{14}+6x_{21}+2x_{22}}{8x_{11}+6x_{12}+10x_{13}+9x_{14}+9x_{21}+12x_{22}}\cdots$$
$$\frac{+3x_{23}+4x_{24}+10x_{31}+5x_{32}+6x_{33}+2x_{34}}{+13x_{23}+7x_{24}+14x_{31}+9x_{32}+16x_{33}+5x_{34}}$$

$$\begin{aligned} s.t. \quad & x_{11}+x_{21}+x_{31} \geq 45, \\ & x_{12}+x_{22}+x_{32} \geq 20, \\ & x_{13}+x_{23}+x_{33} \geq 30, \\ & x_{14}+x_{24}+x_{34} \geq 30, \\ & x_{11}+x_{12}+x_{13}+x_{14} \leq 35, \\ & x_{21}+x_{22}+x_{23}+x_{24} \leq 50, \\ & x_{31}+x_{32}+x_{33}+x_{34} \leq 40, \\ & x_{ij} \geq 0, i=1,2,3, j=1,2,3,4. \end{aligned}$$

When $\epsilon = 10^{-6}$, after 4 iterations, it takes 0.2435 s to solve the problem by the given algorithm, and the optimal solution is $x^* = (0, 17, 18, 0, 21, 0, 4, 25, 24, 3, 8, 5)^T$, the optimal value is 0.5691.

B. The production planning problem

We delve into the production planning problem outlined in [27], and present its corresponding mathematical model in the following manner:

$$max\frac{\sum_{i=1}^{n} c_i x_i + c_0}{\sum_{i=1}^{n} d_i x_i + d_0}$$

$$s.t. \quad Ax \leq b, x \geq 0.$$

The elements of A and b consist solely of randomly generated integers within the range of [0,10]. All d_i and c_i are integers, d_i randomly taken from [0,10] and c_i randomly taken from [-10, 0]. For each instance that is randomly generated, the proposed algorithm is employed to solve the problem 5 times, and the resultant average numerical outcomes are presented in Table IV. An examination of Table IV reveals that the proposed algorithm is capable of effectively addressing this type of problem.

VII. CONCLUDING REMARKS

In this paper, for a class of sum-of-linear-ratios problems, an improved branch-and-bound algorithm is obtained by combining relaxation techniques and hybrid division rules.

TABLE III
COMPARISON OF NUMERICAL RESULTS OF EXAMPLE8 BETWEEN THE ALGORITHM IN THIS PAPER AND THE ALGORITHM IN REFERENCE [10] WHEN $\delta = 10$ AND $\delta = 100$

δ	10				100			
	Ours		[10]		Ours		[10]	
(p, m, n)	iter	time	iter	time	iter	time	iter	time
$(4, 10, 200)$	3.80	4.4059	2.55	0.2364	5.40	3.0238	9.80	2.5156
$(5, 10, 100)$	4.01	3.9256	2.50	0.1711	6.80	4.8521	27.54	4.0338
$(5, 10, 200)$	3.80	5.0328	4.23	0.2885	7.94	6.2125	34.87	8.8408
$(5, 10, 300)$	4.16	7.1789	2.19	0.2384	11.83	7.9897	28.66	11.6785
$(5, 20, 300)$	4.51	7.9154	1.34	0.2328	15.41	9.1213	18.60	9.7752
$(5, 20, 500)$	4.08	17.4266	1.60	0.3188	20.06	22.0931	15.83	14.3958
$(6, 20, 500)$	4.28	18.0535	1.45	0.3732	17.13	20.3645	35.54	27.4491
$(7, 20, 500)$	4.40	19.6236	1.74	0.4363	18.54	20.6411	51.06	48.7854
$(8, 20, 500)$	4.40	20.0412	1.74	0.5976	20.81	23.4046	182.33	183.1540
$(9, 20, 500)$	4.45	22.9562	1.79	0.6248	12.16	12.9536	220.52	22.0193
$(10, 20, 500)$	4.56	25.9861	1.95	0.6453	13.08	13.7458	196.01	29.2540
$(6, 20, 600)$	3.80	27.3811	1.40	0.4479	17.56	25.5235	102.56	9.6076
$(6, 20, 700)$	3.85	32.2583	1.81	0.4988	19.92	33.3258	79.03	117.1986
$(6, 20, 800)$	4.11	40.7466	1.80	0.5481	22.04	38.0006	36.54	69.9123
$(7, 20, 800)$	3.92	40.9808	1.80	0.6278	23.46	38.7024	103.70	185.3989
$(7, 20, 1000)$	3.21	42.3243	1.81	0.7798	18.89	35.8677	184.96	379.9043
$(2, 20, 1000)$	3.16	32.7626	1.24	0.3555	9.88	32.1476	3.47	6.8478
$(3, 20, 1000)$	3.16	32.0842	1.80	0.3639	10.14	33.8058	10.39	20.4984
$(4, 20, 1000)$	3.16	33.7799	1.70	0.4624	10.37	34.9667	33.57	87.6891
$(5, 20, 1000)$	3.21	40.3524	1.70	0.5558	11.21	35.9563	57.88	119.4336
$(6, 20, 1000)$	3.21	44.0178	1.87	1.3932	10.60	36.9854	127.20	261.2227
$(3, 20, 2000)$	3.80	47.9786	1.69	1.2717	13.58	49.1715	17.51	130.2921

TABLE IV
RESULTS OF THE PROPOSED ALGORITHM FOR PRODUCTION PLANNING PROBLEM

(m, n)	Time(seconds)	Iter
$(10, 10)$	0.3970	2
$(100, 100)$	1.4967	1
$(500, 500)$	2.7019	1.57
$(1000, 1000)$	16.6234	3.85
$(2000, 2000)$	50.2186	4.92

The proposed algorithm has a faster convergence speed and more advantages in terms of the iteration times and running time. The results of the numerical experiments demonstrate that, in the first 7 examples, the optimal solution and its corresponding value derived in this paper closely resemble those reported in the references. Notably, the iteration times and the running time required are significantly reduced, thereby suggesting the feasibility and efficacy of the proposed algorithm. In Example8, for large-scale random cases, with the increase of δ, the algorithm in this paper has more advantages in iteration times and is more efficient in solving problems. Finally, the practicality of the proposed algorithm was demonstrated through transportation problem and product planning problem. In future research, further consideration will be given to reducing branch space and proposing acceleration methods to improve the speed of algorithm operation.

REFERENCES

[1] M. T, "Np-hardness of linear multiplicative programming and related problems," *J Global Optim*, vol. 9, pp. 113–119, 1996.

[2] W. H, "Bond portfolio optimization problems and their applications to index tracking: a partial optimization approach," *J Oper Res Soc Jpn*, vol. 39, no. 3, pp. 295–306, 1996.

[3] F. C. A. Maranas C D, Androulakis I P, "Solving long-term financial planning problems via global optimization," *J Econ Dyn Control*, vol. 21, no. 8-9, pp. 1405–1425, 1997.

[4] P. S. W. FALKJE, "Optimizing the sum of linear fractional functions," *Rec Adv Global Optim*, pp. 221–258, 1992.

[5] S. A. S. Gruzdeva T V, "On solving the sum-of-ratios problem," *Appl Math Comput*, vol. 318, pp. 260–269, 2018.

[6] P. S. W. Falk J E, "Image space analysis of generalized fractional programs," *J Global Optim*, vol. 4, pp. 63–88, 1994.

[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, 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.

[10] B. Z. YueLin G, "An outcome-space branch and bound algorithm for the sum-of-linear-ratios programming problem," *Mathematica Numerica Sinica*, vol. 42, no. 2, p. 207, 2020.

[11] T. H. Thi Hoai Phuong N, "A unified monotonic approach to generalized linear fractional programming," *Global Optim*, vol. 26, pp. 229–259, 2003.

[12] B. H. P, "A simplicial branch and bound duality-bounds algorithm for the linear sum-of-ratios problem," *Eur J Oper Res*, vol. 182, no. 2, pp. 597–611, 2007.

[13] 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.

[14] L. D. H, "Global optimization algorithms of fractional programming problems," Ph.D. dissertation, Henan Normal University, 2017.

[15] S. P. P. Shen Z H, “A fully polynomial time approximation algorithm for linear fractional multiplicative programming problems,” *Mathematica Numerica Sinica*, vol. 41, no. 02, pp. 212–218, 2019.
[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] W. K. M, “Global optimization algorithms for solving two classes of nonconvex programming problems,” Ph.D. dissertation, Henan Normal University, 2021.
[18] L. B. Yang T T, Tian Z Y, “A global optimization algorithm of lipschitz function,” *J. Qingdao Univ. (Nat. Sci. Ed.)*, vol. 25, no. 2, pp. 21–24, 2012.
[19] L. S. Y. Jiao H W, “A practicable branch and bound algorithm for sum of linear ratios problem,” *Eur J Oper Res*, vol. 243, no. 3, pp. 723–730, 2015.
[20] R. M. R, “Cluster analysis and mathematical programming,” *J Am Stat Assoc*, vol. 66, no. 335, pp. 622–626, 1971.
[21] W. L. Shen P, Huang B, “Range division and linearization algorithm for a class of linear ratios optimization problems,” *J Comput Appl Math*, vol. 350, pp. 324–342, 2019.
[22] Y. J. Jiao H, Liu S, “Outcome space range reduction method for global optimization of sum of affine ratios problem,” *Open Math*, vol. 14, no. 1, pp. 736–746, 2016.
[23] S. P. P. Wang C F, “A global optimization algorithm for linear fractional programming,” *Appl Math Comput*, vol. 204, no. 1, pp. 281–287, 2008.
[24] J. S. Gao Y, “A global optimization algorithm for sum of linear ratios problem,” *J Appl Math*, vol. 2013, 2013.
[25] W. C. F. Shen P P, “Global optimization for sum of linear ratios problem with coefficients,” *Appl Math Comput*, vol. 176, no. 1, pp. 219–229, 2006.
[26] L. B, “Linear-fractional programming: Theory, methods, applications and software,” *Interfaces*, vol. 36, no. 5, pp. 473–474, 2006.
[27] O. B, “An iterative algorithm to solve a linear fractional programming problem,” *Comput Ind Eng*, vol. 140, p. 106234, 2020”.