Vol. 38 (2018)
No. 1

数 学 杂 志
J. of Math. (PRC)

A NEW BRANCH-AND-BOUND ALGORITHM FOR SOLVING MINIMAX LINEAR FRACTIONAL PROGRAMMING

WANG Chun-feng[1], JIANG Yan[2], SHEN Pei-ping[1]

(*1. College of Mathematics and Information, Henan Normal University, Xinxiang 453007, China*)

(*2. Foundation Department, Zhengzhou Tourism College, Zhengzhou 450009, China*)

Abstract: This paper considers minimax linear fractional programming (MLFP) problem, which has many applications in engineering, management, and so on. For solving problem MLFP, a new branch and bound algorithm is proposed. In this algorithm, a new linear relaxation technique is presented firstly, and then, the branch and bound algorithm is developed. The convergence of this algorithm is proved, and some experiments are provided to show its feasibility and efficiency.

Keywords: linear relaxation; global optimization; minimax linear fractional programming; branch and bound

2010 MR Subject Classification: 90C26; 90C30

Document code: A **Article ID:** 0255-7797(2018)01-0113-11

1 Introduction

This paper considers the following minimax linear fractional programming problem (MLFP)

$$\text{MLFP}\begin{cases} \min\max & \left\{\dfrac{n_1(x)}{d_1(x)}, \dfrac{n_2(x)}{d_2(x)}, \cdots, \dfrac{n_p(x)}{d_p(x)}\right\}, \\ \text{s.t.} & Ax \le b, \end{cases}$$

where $p \ge 2,\ A \in R^{m\times n},\ b \in R^m$ are arbitrary real numbers, $n_i(x) = \sum\limits_{j=1}^{n} c_{ij}x_j + d_i,\ d_i(x) = \sum\limits_{j=1}^{n} e_{ij}x_j + f_i$ are affine functions, $D = \{x \in R^n \mid Ax \le b\}$ is bounded with $\text{int}D \ne \emptyset$, and $\forall x \in D,\ n_i(x) \ge 0,\ d_i(x) > 0,\ i = 1, \cdots, p$. In fact, if we use the method of [1] to preprocess problem MLFP, we also only require $d_i(x) \ne 0,\ i = 1, \cdots, p$.

As an important branch of nonlinear optimization, fractional programming attracted the interest of practitioners and researchers since the 1970's. There are two main reasons. One

* **Received date:** 2016-03-11 **Accepted date:** 2016-06-28

Foundation item: Supported by NSFC (U1404105); the Key Scientific and Technological Project of Henan Province (142102210058); the Youth Science Foundation of Henan Normal University (2013qk02); Henan Normal University National Research Project to Cultivate the Funded Projects (01016400105); the Henan Normal University Youth Backbone Teacher Training.

Biography: Wang Chunfeng (1978–), male, born at Kaifeng, Henan, associate professor, major in optimization algorithm and its applications.

reason is that it frequently appears in various disciplines, including transportation planing [2, 3], government planing [4], finance and investment [5–7], and so on. Another reason is that, fractional programming is NP-hard [8, 9], that is, it generally posses multiple local optimal solutions that are not globally optimal, so it is of great challenge to solve this problem, and it is necessary to put forward some effective methods.

The problem MLFP is a special class of fractional programming, which also attracted the interest of practitioners and researchers during the past years [10–16]. To solve problem MLFP, many algorithms were proposed, including partial linearization algorithm [17], interior point algorithm [18], parametric programming method [19], cutting plane algorithm [20], monotonic optimization approach [21], branch and bound algorithms [1, 22–23], and so on. In addition, some theoretical results were obtained about the problem MLFP, and the readers can refer to the literature [1].

The aim of this paper is to present a new branch and bound algorithm for solving problem MLFP. Compared with other three branch and bound algorithms [1, 22–23], our algorithm is easier to implement. In their methods, to obtain a linear relaxation programming problem of problem MLFP, their procedures need twice linear relaxation. However, our method only need once linear relaxation. Comparison results show that the performance of our algorithm is superior to the other three methods in most case.

This paper is organized as follows. In Section 2, the new linear relaxation technique is presented, which can be used to obtain the linear relaxation programming problem LRP for problem MLFP. In Section 3, the global optimization algorithm is described, and the convergence of this algorithm is established. Numerical results are reported to show the feasibility and efficiency of our algorithm in Section 4.

2 Equivalent Problem and its Linear Relaxation

To solve problem MLFP, we first convert it into an equivalent problem (EP). After that, for generate the linear relaxation problem of EP, we present a new linear relaxation technique.

2.1 Equivalent Problem

In order to derive the equivalent EP of MLFP, we first compute $l_j^0 = \min\limits_{x\in D} x_j$, $l_j^0 = \max\limits_{x\in D} x_j$, and construct the initial rectangle $X^0 = \{x \in R^n \mid l_j^0 \le x_j \le u_j^0,\ j = 1, \cdots, n\}$.

Then by introducing a new variable t, we can obtain the EP of problem MLFP as follows

$$\text{EP}\begin{cases} \min & t, \\ \text{s.t.} & \dfrac{n_i(x)}{d_i(x)} \le t,\ i = 1, \cdots, p, \\ & Ax \le b, \\ & x \in X^0 = \{x \in R^n \mid l_j^0 \le x_j \le u_j^0,\ j = 1, \cdots, n\}. \end{cases}$$

Theorem 1 If (x^*, t^*) is a global optimal solution of EP, then x^* is also a global optimal

solution of problem MLFP, and t^* is the optimal value of EP and MLFP.

Proof Readers can refer to [1].

By Theorem 1, in order to globally solve problem MLFP, we may globally solve EP instead. So, in the following, we only consider how to solve the EP.

2.2 Linear Relaxation Programming Problem

To solve EP, we present a branch and bound algorithm. In this algorithm, a principal process is to construct a linear relaxation programming problem for EP, which can provide a lower bound for the optimal value of EP over $X^k \subseteq X^0$.

Let $X^k = \{x \mid l \leq x \leq u\}$ be the initial box X^0 or modified box as defined for some partitioned subproblem in a branch and bound scheme. We will show how to construct the problem LRP for EP over X^k.

For convenience in expression, let

$$\underline{\xi}_i = \sum_{j=1}^{n} \min\{c_{ij}l_j, c_{ij}u_j\} + d_i, \quad \overline{\xi}_i = \sum_{j=1}^{n} \max\{c_{ij}l_j, c_{ij}u_j\} + d_i,$$
$$\underline{\eta}_i = \sum_{j=1}^{n} \min\{e_{ij}l_j, e_{ij}u_j\} + f_i, \quad \overline{\eta}_i = \sum_{j=1}^{n} \max\{e_{ij}l_j, e_{ij}u_j\} + f_i.$$

Obviously, we have $\underline{\xi}_i \leq n_i(x) \leq \overline{\xi}_i,\ \underline{\eta}_i \leq d_i(x) \leq \overline{\eta}_i,\ i = 1, \cdots, p$.

To derive the problem LRP of EP over X^k, we first consider the term $\frac{n_i(x)}{d_i(x)},\ i = 1, \cdots, p$.

From $\overline{\eta}_i n_i(x) - \underline{\xi}_i d_i(x) \geq 0,\ d_i(x) - \overline{\eta}_i \leq 0$, we have

$$\overline{\eta}_i^2 n_i(x) \geq \overline{\eta}_i n_i(x) d_i(x) - \underline{\xi}_i d_i^2(x) + \underline{\xi}_i \overline{\eta}_i d_i(x). \tag{1}$$

Since $\overline{\eta}_i \underline{\eta}_i d_i(x) > 0$, by dividing (1) with $\overline{\eta}_i \underline{\eta} d_i(x)$, simplifying and rearranging, we have

$$\frac{n_i(x)}{d_i(x)} \geq \frac{n_i(x)}{\overline{\eta}_i} - \frac{\underline{\xi}_i}{\overline{\eta}_i^2} d_i(x) + \frac{\underline{\xi}_i}{\overline{\eta}_i}. \tag{2}$$

Let $\Phi_i(x,t) = \frac{n_i(x)}{d_i(x)} - t$, from (2), we have the following relation

$$\Phi_i(x,t) = \frac{n_i(x)}{d_i(x)} - t \geq \frac{n_i(x)}{\overline{\eta}_i} - \frac{\underline{\xi}_i}{\overline{\eta}_i^2} d_i(x) + \frac{\underline{\xi}_i}{\overline{\eta}_i} - t = \Phi_i^l(x,t). \tag{3}$$

By (3), the linear relaxation programming problem LRP can be established as follows

$$\text{LRP} \begin{cases} \min & t, \\ \text{s.t.} & \Phi_i^l(x,t) \leq 0,\ i = 1, \cdots, p, \\ & Ax \leq b,\ x \in X^k. \end{cases}$$

Let $v(\text{LRP})$ and $v(\text{EP})$ be the optimal value of problems LRP and EP, respectively, from the above discussion, obviously, we have $v(\text{LRP}) \leq v(\text{EP})$ over X^k.

Theorem 2 For all $x \in X^k = [l, u]$, let $\Delta x = u - l$, consider the functions $\Phi_i^l(x,t)$ and $\Phi_i(x,t)$. Then we have

$$\lim_{\Delta x \to 0} (\Phi_i(x,t) - \Phi_i^l(x,t)) \to 0.$$

Proof From the definitions $\Phi_i(x,t)$ and $\Phi_i^l(x,t)$, we have

$$\begin{aligned}
|\ \Phi_i(x,t)-\Phi_i^l(x,t)\ | &= |\ \tfrac{n_i(x)}{d_i(x)}-(\tfrac{n_i(x)}{\overline{\eta}_i}-\tfrac{\xi_i}{\overline{\eta}_i^2}d_i(x)+\tfrac{\xi_i}{\overline{\eta}_i})\ | \\
&\le |\ \tfrac{n_i(x)}{d_i(x)}-(\tfrac{n_i(x)}{\overline{\eta}_i}-\tfrac{\underline{\xi}_i}{\overline{\eta}_i^2}d_i(x)+\tfrac{\underline{\xi}_i}{\overline{\eta}_i})\ | \\
&\le |\ n_i(x)(\tfrac{1}{d_i(x)}-\tfrac{1}{\overline{\eta}_i})\ |+|\ \tfrac{\underline{\xi}_i}{\overline{\eta}_i}(\tfrac{d_i(x)}{\overline{\eta}_i}-1)\ | \\
&= |\ n_i(x)\tfrac{\overline{\eta}_i-d_i(x)}{\overline{\eta}_i d_i(x)}\ |+|\ \tfrac{\underline{\xi}_i}{\overline{\eta}_i}\tfrac{d_i(x)-\overline{\eta}_i}{\overline{\eta}_i}\ | \\
&\le \overline{\xi}_i\tfrac{\overline{\eta}_i-\underline{\eta}_i}{\overline{\eta}_i\underline{\eta}_i}+\tfrac{\underline{\xi}_i}{\overline{\eta}_i}\tfrac{\overline{\eta}_i-\underline{\eta}_i}{\overline{\eta}_i}.
\end{aligned} \tag{4}$$

By the definitions of $\underline{\eta}_i$ and $\overline{\eta}_i$, we know that, $\Delta s \triangleq \overline{\eta}_i-\underline{\eta}_i \to 0$ as $\Delta x \to 0$. Thus from the above inequality, we have

$$\lim_{\Delta x\to 0}(\Phi_i(x,t)-\Phi_i^l(x,t))=0.$$

From Theorem 2, it follows that $\Phi_i^l(x,t)$ will approximate the function $\Phi_i(x,t)$ as $\Delta x \to 0$.

3 Algorithm and its Convergence

In this section, based on the former results, we present the branch and bound algorithm to solve EP.

3.1 Branching Rule

During each iteration of the algorithm, the branching process will generate a more refined partition that cannot yet be excluded from further consideration in searching for a global optimal solution for EP, which is a critical element in guaranteeing convergence. This paper chooses a simple and standard bisection rule, which is sufficient to ensure convergence since it drives the intervals shrinking to a singleton for all the variables along any infinite branch of the branch and bound tree.

Consider rectangle $X=\{x\in R^n \mid l_j\le x_j\le u_j,\ j=1,\cdots,n\}\subseteq X^0$, which is associated with a node subproblem. The branching rule is described as follows

(i) let $k=\text{argmax}\{u_j-l_j \mid j=1,\cdots,n\}$;

(ii) let $\tau=(l_k+u_k)/2$;

(iii) let

$$\begin{aligned}
X^1&=\{x\in R^n \mid l_j\le x_j\le u_j,\ j\ne k, l_k\le x_k\le \tau\},\\
X^2&=\{x\in R^n \mid l_j\le x_j\le u_j,\ j\ne k, \tau\le x_k\le u_k\}.
\end{aligned}$$

Through using this branching rule, the rectangle X is partitioned into two subrectangles X^1 and X^2.

3.2 Branch and Bound Algorithm

Based upon the results and operations given above, this subsection summarizes the basic steps of the proposed algorithm.

Let $LB(X^k)$ and $(x(X^k), t(X^k))$ be the optimal function value of problem LRP and an element of the corresponding argmin over the subrectangle X^k, respectively.

Algorithm statement

Step 1 Set the convergence tolerance $\epsilon \geq 0$; the feasible error $\epsilon_1 \geq 0$; the upper bound $UB_0 = +\infty$; the set of feasible points $F = \emptyset$.

Find $LB_0 = LB(X^0)$ and $(x(X^0), t(X^0))$ by solving the problem LRP over X^0. With the feasible error ϵ_1, if $(x(X^0), t(X^0))$ is feasible to EP, set $(x^0, t^0) = (x(X^0), t(X^0))$, $F = F \bigcup \{(x^0, t^0)\}$ and update UB_0. If $UB_0 - LB_0 \leq \epsilon$, then stop: (x^0, t^0) is an ϵ-optimal solution of EP. Otherwise, set $Q_0 = \{X^0\}$, $k = 1$, and go to Step 2.

Step 2 Set $UB_k = UB_{k-1}$. Subdivide X^{k-1} into two subrectangles $X^{k,1}$, $X^{k,2}$ via the branching rule. Let $\overline{X} = \{X^{k,1}, X^{k,2}\}$.

Step 3 For each $X^{k,t} \in \overline{X}$ $(t = 1, 2)$, find the lower bound $LB(X^{k,t})$ and $(x(X^{k,t}), t(X^{k,t}))$ by solving the LRP over $X^{k,t}$. If $LB(X^{k,t}) > UB_k$, set $\overline{X} = \overline{X} \setminus X^{k,t}$; else if $(x(X^{k,t}), t(X^{k,t}))$ is feasible to EP with feasible error ϵ_1, then set

$$F = F \bigcup \{(x(X), t(X))\}, \quad UB_k = \min\{UB_k, t(X^{k,t})\}.$$

If $UB_k = t(X^{k,t})$, set $(x^k, t^k) = (x(X^{k,t}), t(X^{k,t}))$.

Step 4 Set $Q_k = (Q_{k-1} \setminus X^{k-1}) \bigcup \overline{X}$.

Step 5 Set $LB_k = \min\{LB(X) \mid X \in Q_k\}$. Let X^k be the subrectangle which satisfies that $LB_k = LB(X^k)$. If $UB_k - LB_k \leq \epsilon$, then stop: (x^k, t^k) is a global ϵ-optimal solution of problem EP. Otherwise, set $k = k + 1$, and go to Step 2.

3.3 Convergence Analysis

The following theorem gives the global convergence properties of the above algorithm.

Theorem 3 If the algorithm terminates finitely, then upon termination, x^k is a global ϵ-optimal solution for problem MLFP; else, it will generate an infinite sequence $\{x^k\}$ of iterations such that along any infinite branch of the branch and bound tree, and any accumulation point will be a global optimal solution of problem MLFP.

Proof When the algorithm terminates finitely, the conclusion is obvious. When the algorithm terminates infinitely, as stated in [25], a sufficient condition for the algorithm to be convergent to a global optimum is that the bounding operation must be consistent and the selection operation is bound improving.

A bounding operation is called consistent if at every step any unfathomed partition can be further refined, and if any infinitely decreasing sequence of successively refined partition elements satisfies

$$\lim_{k \to \infty} (UB_k - LB_k) = 0, \tag{5}$$

where UB_k is a computed upper bound in stage k and LB_k is the best lower bound at iteration k not necessarily occurring inside the same subrectangle with UB_k. In the following, we will show (5) holds.

Since the employed subdivision process is the bisection, the process is exhaustive. Consequently, from Theorem 2 and the relationship $v(\text{LRP}) \leq v(\text{EP})$, formulation (5) holds, this implies that the employed bounding operation is consistent.

A selection operation is called bound improving if at least one partition element where the actual upper bound is attained is selected for further partition after a finite number of refinements. Clearly, the employed selection operation is bound improving because the partition element where the actual upper bound is attained is selected for further partition in the immediately following iteration.

Based on the above discussion, we know that the bounding operation is consistent and that selection operation is bound improving. Therefore, according to [25], the employed algorithm is convergent to the global optimum of MFLP.

4 Numerical Experiments

In this section, to verify the performance of the proposed algorithm, some numerical experiments are carried out and compared with three latest algorithms [1, 22–23]. The algorithm is implemented by Matlab 7.1, and all test problems are carried out on a Pentium IV (3.06 GHZ) microcomputer. The simplex method is applied to solve the linear relaxation programming problems.

For test problems 1–8, the convergence tolerance ϵ is set to 5×10^{-8}, the feasible error ϵ_1 are set 0.005, 0.001, 0.001, 0.005, 0.001, 0.001, 0.001, 0.001, which agree with the feasible error used in [1].

The results of problems 1–8 are summarized in Table 1, where the following notations have been used in row headers: *Iter*: number of algorithm iterations; $L_{\max}$: the maximal number of algorithm active nodes necessary.

Example 9 is a random test problem. Table 2 summarizes our computational results of Example 9. For this test problem, the convergence tolerance $\epsilon = 5e - 8$, and the feasible error $\epsilon_1 = 0.001$. In Table 2, *Ave.Iter* denotes the average number of iterations; *Ave.Time* represents the average CPU time of the algorithm in seconds, which are obtained by solving 10 different random instances for each size.

Example 1 [1, 21, 22]

$$
\begin{aligned}
\min \max \quad & \left\{ \frac{3x_1 + x_2 - 2x_3 + 0.8}{2x_1 - x_2 + x_3},\ \frac{4x_1 - 2x_2 + x_3}{7x_1 + 3x_2 - x_3} \right\}, \\
\text{s.t.} \quad & x_1 + x_2 - x_3 \leq 1, \\
& -x_1 + x_2 - x_3 \leq -1, \\
& 12x_1 + 5x_2 + 12x_3 \leq 34.8, \\
& 12x_1 + 12x_2 + 7x_3 \leq 29.1, \\
& -6x_1 + x_2 + x_3 \leq -4.1, \\
& 1.0 \leq x_1 \leq 1.1,\ 0.55 \leq x_2 \leq 0.65,\ 1.35 \leq x_3 \leq 1.45.
\end{aligned}
$$

Example 2 [1, 21, 22]

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

Example 3 [1, 22]

$$\begin{aligned} \min\max \quad & \left\{ \frac{2x_1 + 2x_2 - x_3 + 0.9}{x_1 - x_2 + x_3}, \ \frac{3x_1 - x_2 + x_3}{8x_1 + 4x_2 - x_3} \right\}, \\ \text{s.t.} \quad & x_1 + x_2 - x_3 \le 1, \\ & -x_1 + x_2 - x_3 \le -1, \\ & 12x_1 + 5x_2 + 12x_3 \le 34.8, \\ & 12x_1 + 12x_2 + 7x_3 \le 29.1, \\ & -6x_1 + x_2 + x_3 \le -4.1, \\ & 1.0 \le x_1 \le 1.2, \ 0.55 \le x_2 \le 0.65, \ 1.35 \le x_3 \le 1.45. \end{aligned}$$

Example 4 [1,22]

$$\begin{aligned} \min\max \quad & \left\{ \frac{3x_1 + x_2 - 2x_3 + 0.8}{2x_1 - x_2 + x_3}, \frac{4x_1 - 2x_2 + x_3}{7x_1 + 3x_2 - x_3}, \frac{3x_1 + 2x_2 - x_3 + 1.9}{x_1 - x_2 + x_3}, \frac{4x_1 - x_2 + x_3}{8x_1 + 4x_2 - x_3} \right\}, \\ \text{s.t.} \quad & x_1 + x_2 - x_3 \le 1, \\ & -x_1 + x_2 - x_3 \le -1, \\ & 12x_1 + 5x_2 + 12x_3 \le 34.8, \\ & 12x_1 + 12x_2 + 7x_3 \le 29.1, \\ & -6x_1 + x_2 + x_3 \le -4.1, \\ & 1.0 \le x_1 \le 1.2, \ 0.55 \le x_2 \le 0.65, \ 1.35 \le x_3 \le 1.45. \end{aligned}$$

Example 5 [1,23]

$$\begin{aligned} \min\max \quad & \left\{ \frac{2.1x_1 + 2.2x_2 - x_3 + 0.8}{1.1x_1 - x_2 + 1.2x_3}, \ \frac{3.1x_1 - x_2 + 1.3x_3}{8.2x_1 + 4.1x_2 - x_3} \right\}, \\ \text{s.t.} \quad & x_1 + x_2 - x_3 \le 1, \\ & -x_1 + x_2 - x_3 \le -1, \\ & 12x_1 + 5x_2 + 12x_3 \le 40, \\ & 12x_1 + 12x_2 + 7x_3 \le 50, \\ & -6x_1 + x_2 + x_3 \le -2, \\ & 1.0 \le x_1 \le 1.2, \ 0.55 \le x_2 \le 0.65, \ 1.35 \le x_3 \le 1.45. \end{aligned}$$

Example 6 [1,23]

$$\begin{aligned}
\min\max\ & \left\{\frac{3x_1+4x_2-x_3+0.5}{2x_1-x_2+x_3+0.5}, \frac{3x_1-x_2+3x_3+0.5}{9x_1+5x_2-x_3+0.5}, \frac{4x_1-x_2+5x_3+0.5}{11x_1+6x_2-x_3}, \frac{5x_1-x_2+6x_3+0.5}{12x_1+7x_2-x_3+0.9}\right\},\\
\text{s.t.}\ & x_1+x_2-x_3\le 1,\\
& -x_1+x_2-x_3\le -1,\\
& 12x_1+5x_2+12x_3\le 42,\\
& 12x_1+12x_2+7x_3\le 55,\\
& -6x_1+x_2+x_3\le -3,\\
& 1.0\le x_1\le 2,\ 0.5\le x_2\le 2,\ 0.5\le x_3\le 2.
\end{aligned}$$

Example 7 [1,23]

$$\begin{aligned}
\min\max\ & \left\{\frac{3x_1+4x_2-x_3+0.9}{2x_1-x_2+x_3+0.5}, \frac{3x_1-x_2+3x_3+0.5}{9x_1+5x_2-x_3+0.5}, \frac{4x_1-x_2+5x_3+0.5}{11x_1+6x_2-x_3+0.9}, \frac{5x_1-x_2+6x_3+0.5}{12x_1+7x_2-x_3+0.9}, \frac{6x_1-x_2+7x_3+0.6}{11x_1+6x_2-x_3+0.9}\right\},\\
\text{s.t.}\ & 2x_1+x_2-x_3\le 2,\\
& -2x_1+x_2-2x_3\le -1,\\
& 11x_1+6x_2+12x_3\le 45,\\
& 11x_1+13x_2+6x_3\le 52,\\
& -7x_1+x_2+x_3\le -2,\\
& 1.0\le x_1\le 2,\ 0.35\le x_2\le 0.9,\ 1\le x_3\le 1.55.
\end{aligned}$$

Example 8 [1,23]

$$\begin{aligned}
\min\max\ & \left\{\frac{5x_1+4x_2-x_3+0.9}{3x_1-x_2+2x_3+0.5}, \frac{3x_1-x_2+4x_3+0.5}{9x_1+3x_2-x_3+0.5}, \frac{4x_1-x_2+6x_3+0.5}{12x_1+7x_2-x_3+0.9}, \frac{7x_1-x_2+7x_3+0.5}{11x_1+9x_2-x_3+0.9}, \frac{7x_1-x_2+7x_3+0.7}{11x_1+7x_2-x_3+0.8}\right\},\\
\text{s.t.}\ & 2x_1+2x_2-x_3\le 3,\\
& -2x_1+x_2-3x_3\le -1,\\
& 11x_1+7x_2+12x_3\le 47,\\
& 13x_1+13x_2+6x_3\le 56,\\
& -6x_1+x_2+3x_3\le -1,\\
& 1.0\le x_1\le 2,\ 0.35\le x_2\le 0.9,\ 1\le x_3\le 1.55.
\end{aligned}$$

Example 9

$$\begin{aligned}
\min\max\ & \left\{\frac{\sum_{j=1}^{n} c_{1j}x_j+d_1}{\sum_{j=1}^{n} e_{1j}x_j+f_1}, \frac{\sum_{j=1}^{n} c_{2j}x_j+d_2}{\sum_{j=1}^{n} e_{2j}x_j+d_2}, \cdots, \frac{\sum_{j=1}^{n} c_{pj}x_j+d_p}{\sum_{j=1}^{n} e_{pj}x_j+d_p}\right\},\\
\text{s.t.}\ & Ax\le b,\\
& 0\le x_i\le 3,\ i=1,\cdots,n,
\end{aligned}$$

where all elements of $c_{ij},\ d_{ij},\ i=1,\cdots,p, j=1,\cdots,n$ are randomly generated in [0, 1]; all elements of $d_i,\ f_i$ are randomly generated in $[0, p]$; all elements of A and b are randomly generated in [0, 1].

From Table 1, it can be seen that, except for Examples 2 and 8, the performance of our algorithm is superior to the other three methods. From Table 2, we can see that the

Table 1: Computational results of Examples 1–8

Example	Methods	Optimal solution	Optimal value	$Iter$	$L_{\max}$	Time
1	[1]	(1.01557,0.59185,1.40157)	0.57139	1	2	0.00692539
	[21]	(1.01569,0.59049,1.40367)	0.573102	1	-	0.06
	[22]	(1.01567, 0.59067,1.40339)	0.572810738	6	5	0.017700826
	ours	(1.01568,0.59063,1.40345)	0.57032	1	1	0.005185
2	[1]	(1.5,1.5)	1.49661806	3	4	0.004879
	[21]	(1.5,1.5)	1.489510	1	-	0
	[22]	(1.5,1.5)	1.49072061	6	7	0.00680581
	ours	(1.5,1.5)	1.49124130	6	5	0.006917
3	[1]	(1.016666667,0.55,1.45)	1.344502171	4	4	0.0144057
	[22]	(1.016666667, 0.55,1.45)	1.346854863	8	8	0.0228725
	ours	(1.016667,0.55, 1.45)	1.347325	3	2	0.025291
4	[1]	(1.008333333,0.5,1.45)	2.280126353	3	4	0.0197122
	[22]	(1.008333333,0.5,1.45)	2.284427051	7	8	0.0288736
	ours	(1.016667,0.55, 1.45)	2.39605	1	1	0.003642
5	[1]	(1.0,0.55,1.45)	1.160998779	6	7	0.0104895
	[23]	(1.0,0.55,1.45)	1.160759760	113	106	0.235226
	ours	(1.0,0.55,1.45)	1.16093	5	4	0.01027
6	[1]	(1.345382850,0.50,1.946283817)	0.989117392	21	20	0.0557114
	[23]	(1.339843750,0.50,1.943285553)	0.985599329	580	420	1.19123
	ours	(1.34375, 0.50, 1.94637)	0.98939	20	19	0.049236
7	[1]	(1.504885652,0.350,1.550)	1.117070767	20	20	0.0819911
	[23]	(1.504882813,0.350, 1.550)	1.117065399	747	638	1.68613
	ours	(1.50436, 0.350, 1.550)	1.11772	19	5	0.079462
8	[1]	(1.752859889,0.350,1.550)	1.117793086	26	22	0.148766
	[23]	(1.753906250,0.350, 1.550)	1.117416325	2901	2534	6.52166
	ours	(1.75181, 0.350, 1.550)	1.11788	32	27	0.260751

Table 2: Computational results of Example 9

(p, m, n)	$Ave.Time(s)$	$Ave.Iter$
(2,3,5)	1.3399	25.5
(3,5,5)	1.9204	44.9
(4,10,5)	2.0625	47.9
(10,5,5)	2.5284	50.0
(10,20,5)	2.9531	58.3
(10,50,5)	7.1952	64.8
(20,20,5)	4.5687	63.0
(30,30,5)	9.8783	80.3
(10,10,10)	12.5283	243.2

CPU time and the number of iterations of our algorithm are not sensitive to the size of the problems p and m.

Test results show that our algorithm is competitive and can be used to solve the problem MLFP.

References

[1] Jiao H W, Liu S Y. A new linearization technique for minimax linear fractional programming[J]. Intern. J. Comp. Math., 2014, 91(8): 1730–1743.

[2] Almogy Y, Levin O. Parametric analysis of a multistage stochastic shipping problem (Edited by Lawrence J)[M]. Oper. Res. 69, London, England: Tavistock Publications, 1970.

[3] Falk J E, Palocsay S W. Optimizing the sum of linear fractional functions, recent advances in global optimization (edited by Floudas C A and Pardalos P M)[M]. Princeton, New Jersey: Princeton University Press, 1992.

[4] Colantoni C S, Manes R P, Whinston A. Programming, profit rates and pricing decisions[J]. Account. Rev., 1969, 44: 467–481.

[5] Maranas C D, Androulakis I P, Floudas C A, et al. Solving long-term financial planning problems via global optimization[J]. J. Econ. Dyn. Contr., 1997, 21: 1405–1425.

[6] Konno H, Watanabe H. Bond portfolio optimization problems and their applications to index tracking: a partial optimization approach[J]. J. Oper. Res. Soc. Japan, 1996, 39: 295–306.

[7] Schaible S. Fractional programming[M]. Horst R, Pardalos P M. Handbook of Global Optimization, Dordrecht: KluwerAcademic, 1995.

[8] Freund R W, Jarre F. Solving the sum-of-ratios problem by an interior point method[J]. J. Glob. Optim., 2001, 19: 83–102.

[9] Matsui T. NP-hardness of linear multiplicative programming and related problems[J]. J. Glob. Optim., 1996, 9: 113–119.

[10] Ahmad I, Husain Z. Duality in nondifferentiable minimax fractional programming with generalized convexity[J]. Appl. Math. Comput., 2006, 176: 545–551.

[11] Bajona-Xandri C, Martinez-Legaz J E. Lower subdifferentiability in minimax fractional programming[J]. Optim., 1999, 45: 1–12.

[12] Balasubramaniam P, Lakshmanan S. Delay-interval-dependent robust-stability criteria for neutral stochastic neural networks with polytopic and linear fractional uncertainties[J]. Intern. J. Comput. Math., 2011, 88(10): 2001–2015.

[13] Ding F. Decomposition based fast least squares algorithm for output error systems[J]. Sig. Proc., 2013, 93: 1235–1242.

[14] Ding F. Two-stage least squares based iterative estimation algorithm for CARARMA system modeling[J]. Appl. Math. Model., 2013, 37: 4798–4808.

[15] Zhu J, Liu H, Hao B. A new semismooth newton method for NCPs based on the penalized KK function[J]. Intern. J. Comput. Math., 2012, 89(4): 543–560.

[16] Liu Y J, Ding R. Consistency of the extended gradient identification algorithm for multi-input multi-output systems with moving average noises[J]. Intern. J. Comput. Math., 2013, 90(9): 1840–1852.

[17] Benadada Y, Fedand J A. Partial linearization for generalized fractional programming[J]. Zeitschrift Für Oper. Res., 1988, 32: 101–106.

[18] Freund R W, Jarre F. An interior-point method for fractional programs with convex constraints[J]. Math. Prog., 1994, 67: 407–440.

[19] Crouzeix J P, Ferland J A, Schaible S. An algorithm for generalized fractional programs[J]. J. Optim. The. Appl., 1985, 47: 135–149.

[20] Barros A I, Frenk J B G, Generalized fractional programming and cutting plane algorithms[J]. J. Optim. The. Appl., 1995, 87: 103–120.

[21] Phuong N T H, Tuy H. A unified monotonic approach to generalized linear fractional programming[J]. J. Glob. Optim., 2003, 26: 229–259.

[22] Feng Q, Jiao H, Mao H, Chen Y. A deterministic algorithm for min-max and max-min linear fractional programmingproblems[J]. Intern. J. Comput. Intel. Sys., 2011, 4: 134–141.

[23] Feng Q, Mao H, Jiao H. A feasible method for a class of mathematical problems in manufacturing system[J]. Key Engin. Mater., 2011, 460-461: 806–809.

[24] Li X A, Gu M N. Global optimization method for a class of new fractional programming problems[J]. J. Math., 2012, 36(2): 1011–1020.

[25] Horst R, Tuy H. Global optimization, deterministic approaches[M]. Berlin: Springer-Verlag, 1990.

求极小极大分式规划问题的一个新的分支定界算法

汪春峰[1], 蒋 妍[2], 申培萍[1]

(1.河南师范大学数学与信息科学学院, 河南 新乡 453007)

(2. 郑州旅游职业学院基础部, 河南 郑州 450009)

摘要: 本文研究在工程、管理等领域应用广泛的极小极大线性分式规划问题(MLFP). 为求解MLFP问题, 提出一个新的分支定界算法. 在算法中, 首先给出一个新的线性松弛化技巧; 然后, 构造了一个新的分支定界算法. 算法的收敛性得以证明. 数值实验结果表明了算法的可行性与有效性.

关键词: 线性松弛; 全局优化; 极小极大线性分式规划; 分支定界

MR(2010)**主题分类号**: 90C26; 90C30 **中图分类号**: O221