Research Article

Yingfeng Zhao*, Sanyang Liu, and Hongwei Jiao

A new branch and bound algorithm for minimax ratios problems

DOI 10.1515/math-2017-0072
Received December 5, 2016; accepted April 3, 2017.

Abstract: This study presents an efficient branch and bound algorithm for globally solving the minimax fractional programming problem (MFP). By introducing an auxiliary variable, an equivalent problem is firstly constructed and the convex relaxation programming problem is then established by utilizing convexity and concavity of functions in the problem. Other than usual branch and bound algorithm, an adapted partition skill and a practical reduction technique performed only in an unidimensional interval are incorporated into the algorithm scheme to significantly improve the computational performance. The global convergence is proved. Finally, some comparative experiments and a randomized numerical test are carried out to demonstrate the efficiency and robustness of the proposed algorithm.

Keywords: Fractional programming, Global optimization, Branch and bound, Adapted ratio partition

MSC: 90C26, 90C30

1 Introduction

The minimax fractional programming (MFP) problem we studied in this paper can be formulated as the following nonlinear optimization problem

\[
\text{MFP} : \begin{cases}
\min \max \left\{ \frac{n_1(x)}{d_1(x)}, \frac{n_2(x)}{d_2(x)}, \ldots, \frac{n_p(x)}{d_p(x)} \right\} \\
\text{s.t.} \quad g_j(x) \leq 0, \quad j = 1, 2, \ldots, m, \\
x \in X^0 = \{ x \in \mathbb{R}^n \mid l_i^0 \leq x_i \leq u_i^0, \quad i = 1, 2, \cdots, n \},
\end{cases}
\]

where \(n_i(x), -d_i(x), i = 1, 2, \cdots, p\), \(g_j(x), j = 1, 2, \cdots, m\), are all convex functions and we assume that each numerator and denominator in the objective function satisfy the following condition:

\[
n_i(x) \geq 0, \quad d_i(x) > 0, \quad \forall x \in X^0, \quad i = 1, 2, \cdots, p.
\]

The MFP problem, which we called generalized minimax convex-concave ratios problem, is one of the most topical and useful fields in fractional programming problems, it has attracted the interest of both practitioners and researchers for many years [1–4]. The reason for studying the MFP problem is twofold. The first one is that, from the practical point of view, the MFP problem has quite a lot of applications in various areas where several rates are to be optimized simultaneously, such as design of electronic circuits [5], biology to calculate malting efficient DNA [6], rural farming system [7], neural networks [8], management and finance [9] and system identification [10–12], to name but a few.

Another reason is that, from the research point of view, the MFP problem generalizes minimax linear ratios problem

*Corresponding Author: Yingfeng Zhao: School of Mathematics and Statistics, Xidian University, Xi’an 710071, China and School of Mathematical Science, Henan Institute of Science and Technology, Xinxiang 453003, China, E-mail: zhaoyingfeng6886@163.com
Sanyang Liu: School of Mathematics and Statistics, Xidian University, Xi’an 710071, China, E-mail: liusanyang@126.com
Hongwei Jiao: School of Mathematical Science, Henan Institute of Science and Technology, Xinxiang 453003, China
[13–15] and it is neither quasiconcave nor quasiconvex, so it may have multiple local optima most of which fail to be global optimal (Example 8. and Figure 1), thus it poses significant theoretical and computational challenges. For these reasons, developing new solution methods for the problem MFP is still quite necessary and meaningful.

Over the years, several global optimization methods have been available for solving special case of the problem (MFP), include parametric programming method [16], interior-point algorithm [17], monotonic optimization approach [18] and those kinds of dinkelbach-type algorithms [19, 20]. When the numerators and denominators in the objective function of the (MFP) are all affine, Feng presents two deterministic algorithm for solving the special case of the MFP problem [14, 15]. Recently, using two-step relaxation technique, Jiao and Liu have proposed a new global optimization algorithm for solving the MFP with linear ratios and constraints [13]. Jeyakumar et al. proposed a strong duality for robust minimax fractional programming problems, when data in both of the objective and constraints are uncertainty [21]. Lai and Huang established the sufficient optimality conditions for a minimax programming problem involving \( p \) fractional \( n \)--set functions under generalized convexity [22]. Progress for the MFP problem is far advanced than mentioned above – there are many other studies about (MFP) and its special cases. Despite these various contributions, most knowledge for MFP presented in recent works focus only on optimality conditions or duality theory for special case of the MFP problem, and existing algorithm can only find its local optimal solution, or can only solve the linear form of the MFP problems. For general (MFP) investigated in this paper, to our knowledge, very few global optimization algorithms have been developed.

In this paper, we put forward an unidimensional outcome interval branch-and-bound algorithm for globally solving the (MFP). First, a concise convex relaxation programming of the equivalent problem (EMFP) is established. Then some key operations in outcome branch and bound algorithm are described, especially the interval reduction skill can significantly improve the computational efficiency of the presented algorithm. Third, the proposed algorithm is developed by incorporating the accelerating technique as well as the adapted partition rule into the branch and bound scheme. The global convergence property is proved, numerical experimental results show that the proposed algorithm has higher computational efficiency and stronger robustness than Refs. [13] and [14, 15].

The remainder of this study is organized in the following way. In Section 2, by introducing an auxiliary variable, the MFP problem is converted into an equivalent problem (EMFP). Then the convex relaxation programming problem of the (EMFP) is established in Section 3, the linear relaxation programming for its special case, where the numerators and denominators in the objective function and the constraints are all linear, is also described in this section. Section 4 presents some key operations in branch and bound algorithm for globally solving the (MFP). An unidimensional outcome interval branch and bound algorithm for the (MFP) is introduced in Section 4. The numerical results of some test examples in recent works with the proposed algorithm are reported in Section 5, and some concluding remarks are given in the last section.

## 2 Equivalent problem

For solving the problem, we will first transform the problem (MFP) into an equivalent problem (EMFP) by associating all ratios in the objective function with an auxiliary variable. To this end, we first denote some notations as follows:

\[
\begin{align*}
    n_i & \leq n_i(x) \leq \pi_i, \quad d_i \leq d_i(x) \leq \pi_i, \quad t_{n+1}^0 = \min_{1 \leq i \leq p} \left\{ \frac{n_i}{d_i} \right\}, \quad u_{n+1}^0 = \max_{1 \leq i \leq p} \left\{ \frac{\pi_i}{d_i} \right\},
\end{align*}
\]

where \( n_i, d_i \) can be easily computed by utilizing the convexity and concavity of \( n_i(x) \) and \( d_i(x) \), respectively. Through introducing a positive auxiliary variable \( x_{n+1} \), we can derive the following equivalent problem (EMFP0) of the MFP problem.

\[
\begin{align*}
    \text{(EMFP0)} : \quad \begin{cases}
    \min \ x_{n+1} \\
    \text{s.t.} \quad n_i(x) - x_{n+1}d_i(x) \leq 0, \quad i = 1, 2, \cdots, p, \\
    \quad g_j(x) \leq 0, \quad j = 1, 2, \cdots, m, \\
    \quad x \in X^0 = \{ x \in \mathbb{R}^n \mid l_i^0 \leq x_i \leq u_i^0, \quad i = 1, 2, \cdots, n \}, \\
    \quad x_{n+1} \in D^0 = [t_{n+1}^0, u_{n+1}^0].
    \end{cases}
\end{align*}
\]
By utilizing variable equivalence replacement tactic and changing the notation, one can thus reformulate the (EMFP0) into the following form

\[
\text{(EMFP)} : \begin{cases} 
\min x_{n+1} \\
\text{s.t. } n_{i}(x) - x_{n+1}(d_{i}(x) - \underline{d}_{i}) - \overline{d}_{i}x_{n+1} \leq 0, \\
g_{j}(x) \leq 0, \quad j = 1, \ldots, m, \\
x \in X^{0}, \quad x_{n+1} \in D^{0}.
\end{cases}
\]

Note that, the reason why we convert the (EMFP0) into the form (EMFP) is that by isolating a linear part from the concave function \(d_{i}(x)\) we can make the convex relaxation programming more close to the initial problem (MFP), this operation will improve the algorithm significantly in efficiency which can be vindicated by numerical experiments. Moreover, in the EMFP problem, only the middle part in the first set of constraints is nonconvex. This brings a great convenience to us for building the convex relaxation problem of the initial problem. The equivalence between the problems (MFP) and (EMFP) can be obtained in the sense of the following theorem.

**Theorem 2.1.** If \((x^{*}, x_{n+1}^{*}) \in R^{n+1}\) is a global optimal solution for the (EMFP), then \(x^{*} \in R^{n}\) is a global optimal solution for the (MFP). Conversely, if \(x^{*}\) is a global optimal solution for the (MFP), then \((x^{*}, x_{n+1}^{*}) \in R^{n+1}\) is a global optimal solution for the (EMFP), where \(x_{n+1}^{*} = \max \left\{ \begin{array}{c} \frac{n_{1}(x^{*})}{d_{1}(x^{*})}, \frac{n_{2}(x^{*})}{d_{2}(x^{*})}, \ldots, \frac{n_{p}(x^{*})}{d_{p}(x^{*})} \end{array} \right\} \).

**Proof.** The proof of this theorem can be easily followed according to the definition of the problems (EMFP0) and (EMFP), therefore, it is omitted here. \(\square\)

To solve the MFP problem, based on Theorem 2.1, we only need to solve the EMFP problem instead. Hence from now on, we will assume that the initial problem (MFP) has been of the form like (EMFP), and then our main work will focus on how to globally solve the (EMFP).

### 3 Convex relaxation programming problem

In this section, we construct a convex programming relaxation of the (EMFP) only with relaxed auxiliary variable in an outcome interval \(D\). For the ease of presentation, we assume that \(D = [l_{n+1}, u_{n+1}]\) represents either the initial outcome interval \(D^{0}\) of (EMFP), or modified interval generated from the branching operation in the algorithm. Next, we describe the construction process of the relaxation convex programming (RMFP) of the (EMFP) corresponding to \(D\).

For this, we only need to consider the nonconvex constraints in (EMFP). As a matter of convenience, we denote these constraint functions with notation and number them as follows:

\[
h_{i}(x, x_{n+1}) = n_{i}(x) - x_{n+1}(d_{i}(x) - \underline{d}_{i}) - \overline{d}_{i}x_{n+1}
\]

Assume \(\tau, \upsilon\) denote the best upper bound and lower bound of the objective value known so far (if no such value is founded, set \(\tau := +\infty, \upsilon := -\infty\), respectively, and let \(M = \min\{\tau, u_{n+1}\}\), then we can obtain an underestimate function of \(h_{i}(x)\) which can be formulated as:

\[
\tilde{h}_{i}(x, x_{n+1}) = n_{i}(x) - M(d_{i}(x) - \underline{d}_{i}) - \overline{d}_{i}x_{n+1} = n_{i}(x) - Md_{i}(x) + M\underline{d}_{i} - \overline{d}_{i}x_{n+1},
\]

based on the definition of \(M\) and the convexity and concavity of \(n_{i}(x)\) and \(d_{i}(x)\), it is not hard to see that \(\tilde{h}_{i}(x)\) is convex and provides a very good lower approximation for \(h_{i}(x), i = 1, 2, \ldots, p\). Thus, according to the above discussion, for arbitrary region \(X^{k} \times D^{k} \subseteq X^{0} \times D^{0}\), we can describe the convex relaxation programming problem (RMFP) corresponding to outcome interval \(D\) as follows:

\[
\text{(RMFP)} : \begin{cases} 
\min x_{n+1} \\
\text{s.t. } \tilde{h}_{i}(x, x_{n+1}) \leq 0, \quad i = 1, 2, \ldots, p, \\
g_{j}(x) \leq 0, \quad j = 1, 2, \ldots, m, \\
x \in X^{0}, \quad x_{n+1} \in D.
\end{cases}
\]
Remark 3.1. The relaxation operation we used is very simple and easy to implement, that is we only need to compare two real numbers in each iteration for constructing the relaxation problem. It is quite different from usual branch and bound algorithms most of which utilize concave envelope or linearity technique to construct relaxation, and this will greatly reduce the computation time for establishing underestimate function of non-convex functions in the initial 5 source problem, for establishing underestimate function of nonconvex functions in the initial source problem. Furthermore, since relaxation operation occurs only in the unidimensional outcome interval, we can only subdivide the outcome interval in branching process. This will significantly reduce the numbers of nods in the algorithm so that the outcome interval branch and bound algorithm we presented performs better than usual branch and bound algorithms which branches in the $n$-dimensional variable space.

Remark 3.2. Condition (1) can be relaxed to $d_i(x) \neq 0$, $\forall x \in X^0$, $i = 1, 2, \cdots, p$, when functions $n_i(x)$ and $d_i(x)$ are all linear, that is, the problem we considered in this paper generalized the minimax of linear ratios problem appeared in [13–15]. In fact, when $n_i(x)$ and $d_i(x)$ are all linear functions, suppose instead

$$d_i(x) = \sum_{j=1}^{n} a_{ij} x_j + \gamma_i = \sum_{j \in T^+} a_{ij} x_j + \sum_{j \in T^-} a_{ij} x_j + \gamma_i > 0, i = 1, 2, \cdots, p,$$

(4)

where $T^+ = \{j \mid a_{ij} > 0\}$, $T^- = \{j \mid a_{ij} < 0\}$. If not, we can substitute the ratio $\frac{n_i(x)}{d_i(x)}$ with $\frac{-n_i(x)}{d_i(x)}$, then it will turn into the case identified by (4). Sequentially, the underestimate function $\tilde{h}_i(x)$ can be constructed in the following way

$$\tilde{h}_i(x, x_{n+1}) = n_i(x) - M_{i1} \sum_{j \in T^+} a_{ij} x_j - M_{i2} \sum_{j \in T^-} a_{ij} x_j - \gamma_i x_{n+1},$$

(5)

where $M_{i1} = \min\{\gamma_i, u_{n+1}\}$, $M_{i2} = \max\{\gamma_i, l_{n+1}\}$. It is much simpler and more economical to use this relaxation technique in branch and bound algorithm than that in [14, 15] where the linear relaxation programming is constructed by utilizing convex and (or) concave envelopes of bilinear functions, and the numerical test will illustrate this.

Theorem 3.3. For any $x \in X^0$, $x_{n+1} \in D$, let $\Delta = |u_{n+1} - l_{n+1}|$, then the following conclusion will be established:

$$\tilde{h}_i(x) \leq h_i(x), \lim_{\Delta \to 0} |\tilde{h}_i(x) - h_i(x)| = 0.$$

Proof. The proof of this theorem can be easily verified by the expression of $\tilde{h}_i(x, x_{n+1})$ in (3), therefore, it is omitted here.

Theorem 2.1 and Theorem 3.3 ensure that $\tilde{h}_i(x, x_{n+1})$ are good underestimations for $h_i(x, x_{n+1})$ and will provide valid lower bounds for the optimal value of the initial problem (MFP).

4 Algorithm and its convergence

In this section, we will first present some key operations of the branch and bound algorithm, then the global optimization algorithm based on the former convex relaxation programming will be proposed, and finally, we will prove the global convergence of the present algorithm in theory.

4.1 Key operations

There are several key operations in reduced outcome interval branch and bound algorithm. The focus of these operations is to find better upper and lower bounds of the objective value at a faster pace. In our algorithm, there are three base operations: branching, reduction and bounding.
Choosing a suitable partition rule is a critical element in guaranteeing convergence to a global optimal solution, here we choose a simple and adapted bisecion rule which is sufficient to ensure convergence. For any subproblem identified by hyper-rectangle

\[ X^0 \times D = \{(x, x_{n+1}) \mid x \in X^0, l_{n+1} \leq x_{n+1} \leq u_{n+1} \} \subset X^0 \times D^0, \]

let \( q = (l_{n+1} + u_{n+1})/2 \), then partition \( X^0 \times D \) will be subdividing into two regions \( X^0 \times \tilde{D} \) and \( X^0 \times \tilde{D} \), where \( \tilde{D} = [l_{n+1}, q] \), \( \tilde{D} = [q, u_{n+1}] \). Note that we only partition the outcome unidimensional interval \( D \), the \( n \)-dimensional variable space is never partitioned in the algorithm, thus the number of partition set is much smaller than in case of those branch and bound algorithms using pure variable space partition rule. What’s more, the adapted partition technique corresponding to the current objective value will also improve the performance of the algorithm.

Moreover, the partition set generated from the branching operation can be further reduced by the special structure of the (EMFP). Actually, based on Theorem 2.1, we know that the EMFP problem has the same optimal value of those branch and bound algorithms using pure variable space partition rule. What’s more, the adapted partition technique corresponding to the current objective value will also improve the performance of the algorithm.

4.2 Algorithm statement and convergence property

Based upon the results and key operations given in the above sections, the new branch and bound algorithm for solving the EMFP problem, and hence the MFP problem, can be described as follows.

**Step 0.** (Initialization) Choose convergence tolerance \( \epsilon > 0 \), set iteration counter \( k := 0 \) and the initial active nod as \( \Omega_0 = X^0 \times D^0 \).

Solve the initial convex relaxation problem (RMFP) over region \( X^0 \times D^0 \), if the (RMFP) is not feasible then there is no feasible solution for the (MFP). Else, denote the optimal value and solution as \( f_0 \) and \( x_{opt}^0 \), respectively, then we can obtain the initial upper and lower bound of the optimal value for the MFP problem, that is, \( \bar{\tau} := f(x_{opt}^0) \), and \( \underline{\tau} := f_0 \), where \( f(x) = \max \{ m_i(x) \mid i = 1, 2, \ldots, p \} \). Then and, if \( \bar{\tau} - \underline{\tau} < \epsilon \), the algorithm can stop, and \( x_{opt}^0 \) is the optimal solution of the (MFP), otherwise proceed to step 1.

**Step 1.** (Reduction) Substitute the interval end points \( l_{n+1}^k \) and \( u_{n+1}^k \) by utilizing the reduction technique (6).

**Step 2.** (Branching) Partition \( D^k \) into two new sub-intervals according to the adapted rule described in section 4.1. Add the new nods into the active nods set and denote the set of new partitioned intervals as \( D^k \).

**Step 3.** (Bouding) For each subregion still of interest \( X^0 \times D^k \subseteq X^0 \times D, \mu = 1, 2, \ldots, s_k \), obtain the optimal solution and value for the RMFP problem corresponding outcome interval \( D^k \mu \) by solving some convex programming, if \( LB(D^k \mu) > \bar{\tau} \), or \( f(x^k \mu) < \underline{\tau} \) then delete \( D^k \mu \) from \( \tilde{D}^k \). Otherwise, let \( LB^k = \min \{ LB(D^k \mu) \mid \mu = 1, 2, \ldots, s_k \} \) and \( UB^k = \min \{ f(x^k \mu), \bar{\tau} \} \), then we can update the lower and upper bounds as follows

\[ \underline{\tau} := \max \{ \underline{\tau}, LB^k \}, \quad \bar{\tau} := \min \{ \bar{\tau}, UB^k \}, \]

and let

\[ \Omega_k = (\Omega_k \setminus X) \cup \tilde{D}^k \]
Step 4. (Termination) Let
\[
\Omega_{k+1} = \Omega_k \setminus \{X \mid f(X) - LB(X) \leq \epsilon, \ X \in \Omega_k\}.
\]
If \(\Omega_{k+1} = \emptyset\), the algorithm can be stopped, \(\pi\) is the global optimal value for (MFP). Otherwise, set \(k := k + 1\), select \(X^k\) from \(\Omega_k\) with \(X^k = \operatorname{argmin}_{X \in \Omega_k} LB(X)\), then return to Step 3.

**Theorem 4.1.** The proposed algorithm either terminates within finite iterations with an optimal solution for the MFP is found, or generates an infinite sequence of iterations such that along any infinite branches of the branch-and-bound tree, any accumulation point of the sequence \(\{x^k\}\) will be the global optimal solution of the (MFP).

**Proof.** (1) If the proposed algorithm is finite, it will terminate in some iteration \(k, \ k \geq 0\). And it can be known, by the termination criteria, that
\[
\pi - v \leq \epsilon.
\]
From Step 0 and Step 3, it implies that
\[
f(x^k) - LB_k \leq \epsilon.
\]
Let \(v_{opt}\) be the optimal value of the MFP problem. Then by section 3 and section 4.1, we known that
\[
f(x^k) \geq v_{opt} \geq LB_k.
\]
Hence, taken together, it implies that
\[
v_{opt} + \epsilon \geq LB_k + \epsilon \geq f(x^k) \geq v_{opt}.
\]
And thus the proof of part (1) is completed.

(2) If the algorithm is infinite and generates an infinite feasible solution sequence \(\{(x^k, x^{k+1})\}\) via solving the (RMFP). Let \(x^{k+1} = f(x^k)\), then \(\{(x^k, x^{k+1})\}\) is a feasible solution sequence for the (EMFP). Since the sequence \(\{x^k\}\) is bounded, it must have accumulations, without loss of generality, assume \(\lim_{k \to \infty} x^k = x^*\). On the other hand, by the continuity of \(n_i(x)\) and \(d_i(x)\), we can get
\[
\lim_{k \to \infty} x^k_{n+1} = \lim_{k \to \infty} f(x^k) = f(x^*). \quad (7)
\]
Also, according to the branching regulation described before, we can see that
\[
\lim_{k \to \infty} l^k_{n+1} = \lim_{k \to \infty} u^k_{n+1} = x^*_{n+1}. \quad (8)
\]
What’s more, note that \(l^k_{n+1} \leq f(x^k) \leq u^k_{n+1}\), taken (7) and (8) together, we can come to the conclusion that
\[
x^*_{n+1} = \lim_{k \to \infty} f(x^k) = f(x^*) = \lim_{k \to \infty} x^k_{n+1}.
\]
Therefore \((x^*, x^*_{n+1})\) is also a feasible solution for the (EMFP). Further more, since the lower bound sequence \(LB^k\) for the optimal value is increasing and lower bounded by the optimal value \(v_{opt}\) in the algorithm, combining the continuity of \(x_{n+1}\), we have
\[
\lim_{k \to \infty} LB^k = x^*_{n+1} \leq v_{opt} \leq \lim_{k \to \infty} x^k_{n+1} = x^*_{n+1}.
\]
That is, \((x^*, x^*_{n+1})\) is an optimal solution for the (EMFP), and of course \(x^*\) is an optimal solution for the (MFP) according to the equivalence of the problems (MFP) and (EMFP), the proof is completed. \(\square\)
5 Numerical experiments and results

The purpose of this section is to demonstrate the computational performance of our algorithm for the MFP problem. To this end, some numerical experiments in recent works have been carried out on a personal computer containing an Intel Core i5 processor of 2.40 GHz and 4GB of RAM. The code base is written in Matlab2014a and interfaces LINPROG for the linear relaxation problems. The comparison results are listed in Table 1. In addition, to assess the feasibility and stability of the proposed algorithm, a randomly generated test problem is given at the end of this section, all problems are solved with relative/absolute optimality tolerance of $1 \times 10^{-6}$, and we use Iter, Nods and Time to denote the number of iterations, maximum number of nodes stored in memory and the CPU time required in the solving process, respectively.

Example 1 ([14]).

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

Example 2 ([15]).

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

Example 3 ([13, 15]).

$$
\begin{aligned}
& \min \ \max \left\{ \frac{3x_1 + 4x_2 - x_1 + 0.5}{2x_1 - x_2 + x_3 + 0.5}, \frac{3x_1 - x_1 + 3x_3 + 0.5}{9x_1 + 3x_2 - x_3 + 0.5}, \frac{4x_1 - x_3 + 5x_3 + 0.5}{11x_1 + 6x_2 - x_3}, \frac{5x_1 - x_3 + 6x_3 + 0.5}{12x_1 + 7x_2 - x_3 + 0.9} \right\} \\
\text{s.t.} & \quad x_1 + x_2 - x_3 \leq 1, \\
& \quad -x_1 + x_2 - x_3 \leq -1, \\
& \quad 12x_1 + 5x_2 + 12x_3 \leq 42, \\
& \quad 12x_1 + 12x_2 + 7x_3 \leq 55, \\
& \quad -6x_1 + x_2 + x_3 \leq -3, \\
& \quad 1.0 \leq x_1 \leq 2, 0.50 \leq x_2 \leq 2, 0.5 \leq x_3 \leq 2.
\end{aligned}
$$

Example 4 ([13, 15]).

$$
\begin{aligned}
& \min \ \max \left\{ \frac{3x_1 + 4x_2 - x_3 + 0.9}{2x_1 - x_2 + x_3 + 0.5}, \frac{3x_1 - x_3 + 3x_3 + 0.5}{9x_1 + 5x_2 - x_3 + 0.5}, \frac{4x_1 - x_3 + 5x_3 + 0.5}{11x_1 + 6x_2 - x_3 + 0.9}, \frac{5x_1 - x_3 + 6x_3 + 0.5}{12x_1 + 7x_2 - x_3 + 0.9}, \frac{6x_1 - x_2 + x_3 + 0.6}{11x_1 + 6x_2 - x_3 + 0.9} \right\} \\
\text{s.t.} & \quad 2x_1 + x_2 - x_3 \leq 2, \\
& \quad -2x_1 + x_2 - 2x_3 \leq -1, \\
& \quad 11x_1 + 6x_2 + 12x_3 \leq 45, \\
& \quad 11x_1 + 13x_2 + 6x_3 \leq 52, \\
& \quad -7x_1 + x_2 + x_3 \leq -2, \\
& \quad 1.0 \leq x_1 \leq 2, 0.35 \leq x_2 \leq 0.9, 1.0 \leq x_3 \leq 1.55.
\end{aligned}
$$
Example 5 ([13, 15]).
\[
\begin{align*}
\begin{cases}
\min & \max \left\{ \frac{5x_1+4x_2-x_3+0.9}{3x_1-x_2+2x_3+0.5}, \frac{3x_1-x_2+4x_3+0.5}{9x_1+3x_2-x_3+0.5}, \frac{4x_1-x_2+6x_3+0.5}{7x_1-x_2+7x_3+0.5}, \frac{7x_1-x_2+7x_3+0.7}{12x_1+7x_2-x_3+0.9}, \frac{7x_1-x_2+7x_3+0.9}{TTx_1+9x_2-x_3+0.9}, \frac{7x_1-x_2+7x_3+0.7}{TTx_1+7x_2-x_3+0.9} \right\} \\
\text{s.t.} & \quad 2x_1 + 2x_2 - x_3 \leq 3, \\
& \quad -2x_1 + x_2 - 3x_3 \leq -1, \\
& \quad 11x_1 + 7x_2 + 12x_3 \leq 47, \\
& \quad 13x_1 + 13x_2 + 6x_3 \leq 56, \\
& \quad -6x_1 + 2x_2 + 3x_3 \leq -1, \\
& \quad 1.0 \leq x_1 \leq 2, 0.35 \leq x_2 \leq 0.9, 1.0 \leq x_3 \leq 1.55.
\end{cases}
\end{align*}
\]

Example 6 ([13, 15, 18]).
\[
\begin{align*}
\begin{cases}
\min & \max \left\{ \frac{37x_1+73x_2+13}{13x_1+13x_2+13}, \frac{63x_1-18x_2+39}{13x_1+26x_2+13} \right\} \\
\text{s.t.} & \quad 5x_1 - 3x_2 = 3, \\
& \quad 1.5 \leq x_1 \leq 3.
\end{cases}
\end{align*}
\]

Example 7.
\[
\begin{align*}
\begin{cases}
\min & \max \left\{ \frac{2x_1^2+x_2^2+2x_1x_2}{3x_1^2-2x_2^2+4x_1x_2+14}, \frac{x_1^2+x_1x_2+x_2^2+1}{-x_1^2+5} \right\} \\
\text{s.t.} & \quad x_1^2 + x_2^2 \leq 4, \\
& \quad x_1 + x_2 \geq 1.
\end{cases}
\end{align*}
\]

Fig. 1. 3-D image of the objective function over [0.2;0.2] in Example 7.

Example 8.
\[
\begin{align*}
\begin{cases}
\min & \max \left\{ \frac{2x_1^2+3x_2^2}{6-x_2^2}, \frac{3x_1^2+x_2^2+4x_1x_2+0.5}{-9x_1^2+43} \right\} \\
\text{s.t.} & \quad 2x_1 + 3x_2 \leq 5, \\
& \quad -2x_1 + x_2 \leq -1, \\
& \quad 1 \leq x_1 \leq 2, 0.35 \leq x_2 \leq 0.9.
\end{cases}
\end{align*}
\]
Example 9 ([13]).

\[
\begin{align*}
\min & \max \left\{ \sum_{i=1}^{N} n_{i1} x_{i1} + \tilde{h}_{1}, \ldots , \sum_{i=1}^{N} n_{2i} x_{i1} + \tilde{h}_{2}, \ldots , \sum_{i=1}^{N} n_{pi} x_{i1} + \tilde{h}_{p} \right\} \\
\text{s.t. } & AX \leq b, \\
& 0 \leq x_{i} \leq 3, i = 1, 2, \ldots , N.
\end{align*}
\]

Table 1. Results of the numerical contrast experiments 1-7.

| Example | Methods | Optimal value | Optimal solution | Iter | Nods | Time |
|---------|---------|---------------|-----------------|------|------|------|
| 1       | [14]    | 2.2801        | (1.008333, 0.5, 1.45) | 7    | 8    | 0.029 |
|         | our     | 2.2801        | (1.00833, 0.5000, 1.4500) | 3    | 5    | 0.0109 |
| 2       | [15]    | 1.1616        | (1.0, 0.55, 1.45)  | 113  | 106  | 0.2352 |
|         | our     | 1.1616        | (1.0000, 0.5500, 1.4500) | 18   | 31   | 0.1332 |
| 3       | [13]    | 0.9899        | (1.3453, 0.5000, 1.946) | 21   | 20   | 0.5577 |
|         | [15]    | 0.9899        | (1.3398, 0.5000, 1.943) | 590  | 420  | 1.1912 |
|         | our     | 0.9897        | (1.3452, 0.5000, 1.9465) | 16   | 31   | 0.04966 |
| 4       | [13]    | 1.1181        | (1.5049, 0.5000, 1.550) | 20   | 20   | 0.0820 |
|         | [15]    | 1.1181        | (1.5049, 0.5000, 1.550) | 747  | 638  | 1.6861 |
|         | our     | 1.1179        | (1.5053, 0.5000, 1.550) | 13   | 22   | 0.0641 |
| 5       | [13]    | 1.1186        | (1.7539, 0.350, 1.550) | 2901 | 2534 | 6.5217 |
|         | [15]    | 1.1184        | (1.7529, 0.350, 1.550) | 26   | 22   | 0.1488 |
|         | our     | 1.1183        | (1.7538, 0.3500, 1.550) | 10   | 12   | 0.0835 |
| 6       | [13]    | 1.1186        | (1.5, 1.5)        | 6    | 7    | 0.0048 |
|         | [15]    | 1.1184        | (1.5, 1.5)        | 10   | 3    | 0.0068 |
|         | [18]    | 1.1184        | (1.5, 1.5)        | 1    | 4    | 0.0000 |
|         | our     | 1.48951       | (1.5, 1.5)        | 1    | 1    | 0.0000 |
| 7       | our     | 0.3633        | (0.3668, 0.6332)  | 9    | 17   | 0.0454 |
| 8       | our     | 0.0742        | (1.000, 0.35)     | 12   | 27   | 0.0638 |

In Table 1, we provide the comparative results of our algorithm with some other methods identified in the second column of the table. It can be seen that our algorithm performs much better than other algorithms with respect to efficiency.

Table 2. Random numerical experiment 8 comparison with Ref. [13].

| (p,m,n) | Method | Nods | time(s) |
|---------|--------|------|---------|
| (5,4,3) | [13]   | 14   | 0.211   |
|         | our    | 14   | 0.211   |
| (6,5,5) | [13]   | 59   | 0.694   |
|         | our    | 11   | 0.694   |
| (7,5,6) | [13]   | 96   | 1.518   |
|         | our    | 26   | 1.518   |
| (7,5,7) | [13]   | 14   | 0.929   |
|         | our    | 10   | 0.929   |
| (9,6,7) | [13]   | 57   | 14.99   |
|         | our    | 50   | 14.99   |
| (9,7,10) | [13]  | 217  | 30.92   |
|         | our    | 24   | 30.92   |
| (20,7,10) | [13] | 12   | 1.617   |
|         | our    | 12   | 1.617   |
| (45,70,10) | [13] | 12   | 6.385   |
|         | our    | 12   | 6.385   |
| (50,7,10) | [13] | 33   | 1.522   |

In Table 2, the results of randomized test compared with reference [13] are reported. Also we demonstrated the comparative results with a double y-axes broken line graph in Figure 2. From table 2 and Figure 2, it easy to see that our method is much more stable and effective than Jiao’s method.
Fig. 2. Result of random experiment 8 comparison with Ref. [13].

![Graph showing comparison](image)

| (p,m,n) | Method | Nods | time(s) |
|---------|--------|------|---------|
| (2,1,5) | [15]   | 1809 | 2.93991 |
|         | bur    | 19   | 0.341   |
| (2,3,5) | [15]   | 1818 | 3.63135 |
|         | bur    | 18   | 0.163   |
| (3,3,5) | [15]   | 1597 | 64.1806 |
|         | bur    | 13   | 0.368   |
| (4,3,3) | [15]   | 330  | 0.53731 |
|         | bur    | 20   | 0.323   |
| (10,2,3)| [15]   | 110  | 0.373   |
|         | bur    | 10   | 0.293   |
| (11,3,3)| [15]   | 921  | 42.2537 |
|         | bur    | 11   | 0.384   |
| (12,3,5)| [15]   | 1280 | 143.786 |
|         | bur    | 20   | 0.702   |
| (18,3,5)| [15]   | 1656 | 17.5244 |
|         | bur    | 10   | 0.523   |
| (25,10,4)| [15] | 672  | 17.5517 |
|         | bur    | 15   | 0.422   |

Table 3. Random numerical experiment 8 comparison with Ref. [15].

In Table 3, the results of randomized test compared with reference [15] are reported. Also we demonstrated the comparative results with a double y-axes broken line graph in Figure 3. From table 3 and Figure 3, it easy to see that our method performed significantly better than Feng’s method in [15].

6 Concluding remarks

In this paper, we described an efficient implementation of unidimensional outcome branch and bound algorithm for global optimization of the MFP problem. Instead of using linearization or concave envelope technique, we constructed the concise convex relaxation programming by simple arithmetic operations. This relaxation method is more practical and available for convex-concave ratios problem which generalized the linear case appeared in various works. Moreover, unidimensional adapted interval partition and reduction techniques are incorporated into the algorithm scheme, which can significantly improve the performance of the algorithm. Global convergence is proved and the feasibility and robustness of the algorithm is illustrated by numerical experiments.
Fig. 3. Result of random experiment 8 comparison with Ref. [15].

Acknowledgement: The authors would like to express their sincere thanks to the responsible editor and the anonymous referees for their valuable comments and suggestions, which have greatly improved the earlier version of our paper.

References

[1] Ahmad I., Husain Z., Duality in nondifferentiable minimax fractional programming with generalized convexity, Appl. Math. Comput., 2006, 176, 545-551
[2] Damaneh M., On fractional programming problems with absolute-value functions, Int. J. Comput. Math., 2011, 88(4), 661-664
[3] Zhu J., Liu H., Hao B., A new semismooth newton method for NCPs based on the penalized KK function, Int. J. Comput. Math., 2012, 89(4), 543-560
[4] Jiao H., Liu S., A practicable branch and bound algorithm for sum of linear ratios problem, Euro. J. Oper. Res., 2015, 243, 723-730
[5] Barrodale I., Best rational approximation and strict quasiconvexity, SIAM J. Numer. Anal., 1973, 10, 8-12
[6] Leber M., Kaderali L., Schnhuth, R., Schrader, A, Fractional programming approach to efficient DNA melting temperature calculation, Bioinformatics, 2005, 21(10), 2375-2382
[7] Fasakhodi A., Nouri S., Amini M., Water resources sustainability and optimal cropping pattern in farming systems: a multi-objective fractional goal programming approach, Water Res. Manag., 2010, 24, 4639-4657
[8] Balasubramaniam P., Lakshmanan S., Delay-interval-dependent robust-stability criteria for neutral stochastic neural networks with polytopic and linear fractional uncertainties, Int. J. Comput. Math., 2011, 88(10), 2001-2015
[9] Goedhart M., Spronk J., Financial planning with fractional, goals. Eur. J. Oper. Res., 1995, 82(1), 111-124
[10] Ding F., Coupled-least-squares identification for multivariable systems, IET Control Theory Appl., 2013, 7(1), 68-79
[11] Ding F., Hierarchical multi-innovation stochastic gradient algorithm for hammerstein nonlinear system modeling, Appl. Math. Model., 2013, 37, 1694-1704
[12] Liu Y., Ding R., Consistency of the extended gradient identification algorithm for multi-input multi-output systems with moving average noises, Int. J. Comput. Math., 2013, 90(9), 1840-1852
[13] Jiao H., Liu S., A new linearization technique for minimax linear fractional programming, Int. J. Comput. Math., 2014, 91(8), 1730-1743
[14] Feng Q., Jiao H., Mao H., Chen Y., A Deterministic Algorithm for Min-max and Max-min Linear Fractional Programming Problems, Int. J. Comput. Intell. Syst., 2011, 4, 134-141
[15] Feng Q., Mao H., Jiao H., A feasible method for a class of mathematical problems in manufacturing system, Key Eng. Mater., 2011, 460-461, 806-809
[16] Crouzeix J., Ferland J., Schaible S., An algorithm for generalized fractional programs, J. Optim. Theory Appl., 1985, 47, 135-149
[17] Freund R., Jarre F., An interior-point method for fractional programs with convex constraints, Math. Program., 1994, 67, 407-440
[18] Phuong N., Tuy H., A unified monotonic approach to generalized linear fractional programming, J. Global Optim., 2003, 26, 229-259
[19] Borde J., Crouzeix J., Convergence of a dinkelbach-type algorithm in generalized fractional programming, Z. Oper. Res., 1987, 31(1), A31-A54.

[20] Lin J., Sheu R., Modified dinkelbach-type algorithm for generalized fractional programs with infinitely many ratios, J. Optim. Theory Appl., 2005, 126(2), 323-343.

[21] Jeyakumar V., Li G., Srisatkunarajah S., Strong duality for robust minimax fractional programming problems, Eur. J. Oper. Res., 2013, 228, 331-336.

[22] Lai H., Huang T., Optimality conditions for nondifferentiable minimax fractional programming with complex variables, J. Math. Anal. Appl., 2009, 359, 229-239.