RAIRO-Oper. Res. 55 (2022) 1533–1552
https://doi.org/10.1051/ro/2022061

RAIRO Operations Research
www.rairo-ro.org

IMAGE SPACE BRANCH-REDUCTION-BOUND ALGORITHM FOR GLOBALLY MINIMIZING A CLASS OF MULTIPLICATIVE PROBLEMS

Hongwei Jiao[1,*], Wenjie Wang[1], Jingben Yin[1] and Youlin Shang[2]

Abstract. This paper presents an image space branch-reduction-bound algorithm for solving a class of multiplicative problems (MP). First of all, by introducing auxiliary variables and taking the logarithm of the objective function, an equivalent problem (EP) of the problem (MP) is obtained. Next, by using a new linear relaxation technique, the parametric linear relaxation programming (PLRP) of the equivalence problem (EP) can be established for acquiring the lower bound of the optimal value to the problem (EP). Based on the characteristics of the objective function of the equivalent problem and the structure of the branch-and-bound algorithm, some region reduction techniques are constructed for improving the convergence speed of the algorithm. Finally, the global convergence of the algorithm is proved and its computational complexity is estimated, and numerical experiments are reported to indicate the higher computational performance of the algorithm.

Mathematics Subject Classification. 90C26.

Received November 22, 2021. Accepted May 1, 2022.

1. Introduction

This paper considers the following a class of multiplicative problems:

$$(\mathrm{MP}):\begin{cases}\min & G(y)=\prod_{j=1}^{p}(\sum_{i=1}^{n}e_{ji}y_i+f_j)^{\alpha_j},\\ \text{s.t.} & y\in D=\{y\in\mathbb{R}^n|Ay\le b\},\end{cases}$$

where e_{ji}, f_j, and $\alpha_j, j=1,\ldots,p, i=1,\ldots,n$, are all arbitrary real numbers; $A\in\mathbb{R}^{m\times n}$, $b\in\mathbb{R}^m$; D is a nonempty bounded set, and for any $y\in D$, $\sum_{i=1}^{n}e_{ji}y_i+f_j>0, j=1,2,\ldots,p$.

The problem (MP) and its special form are worth studying, and which have attracted much attention from scholars. The first reason is that the problem (MP) and its special form have a wide of practical applications, such as network flows [3, 4], VLISI chip design [6], robust optimization [26], decision tree optimization [1], financial optimization [19,24], and so on [10,14,15]. The second reason is that the problem (MP) and its special form are all non-convex programming problem, which generally contain multiple local optimal solutions that are not global optimal, so that there are many computational difficulties.

Keywords. Multiplicative problem, global optimization, branch-and-bound algorithm, image space region reduction technique, computational complexity.

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

[2] School of Mathematics and Statistics, Henan University of Science and Technology, Luoyang 471023, P.R. China.

*Corresponding author: jiaohongwei@126.com

In the past several decades, some algorithms have been proposed for solving such problems, which can be classified into the following categories: branch-and-bound algorithms [7, 9, 12, 23, 25, 29, 35–38], level set algorithm [22], outer-approximation [21], cutting-plane method [2], finite algorithms [20,27], etc. In recent years, Shen *et al.* [30] proposed an outer space branch-and-bound algorithm for linear multiplicative programming problem by combining the linear relaxation method with the rectangular branching technique and the outer space region reduction technique; by using the decomposability of the problem, Shen and Huang [28] proposed a decomposition branch-and-bound algorithm for linear multiplicative problem; Jiao *et al.* [13] presented an efficient outer space branch-and-bound algorithm for generalized linear multiplicative programming problem based on the outer space search and the branch-and-bound framework; Zhang *et al.* [39] presented a new relaxation bounding method based on the search of the output space; based on the characteristics of the initial problem, Shen *et al.* [31] proposed a branch-and-bound algorithm for globally solving the linear multiplicative problem. Zhang *et al.* [40] proposed an efficient polynomial time algorithm for a class of generalized linear multiplicative programs with positive exponents by utilizing a new two-stage acceleration technique; Jiao and Shang [11] gave a two-Level linear relaxation method for generalized linear fractional programming problem, which includes linear multiplicative problem; by using new affine relaxed technique, Jiao *et al.* [16] formulated a novel branch-and-bound algorithm for solving generalized polynomial problem. However, although many scholars have proposed some algorithms, these algorithms either can only solve a certain special form of the problem (MP), or it is difficult to solve the problem (MP) with large-size variables. Therefore, it is still very necessary to propose a new effective algorithm for globally solving the general form of the problem (MP).

In this paper, an image space branch-reduction-bound algorithm is proposed for globally solving the problem (MP). In the algorithm, first of all, by introducing new auxiliary variables and taking the logarithm of the objective function, an equivalent problem (EP) of the problem (MP) is constructed. Next, by using the first-order differential median theorem, we present a new linear relaxation technique, which can be used to construct the parametric linear relaxation programming problem of the problem (EP). By subsequently refining the initial image space rectangle, and by solving a series of parametric linear relaxation programming problems, the optimal solution of the parametric linear relaxation programming problem (PLRP) can infinitely approximate the optimal solution of the problem (EP). Finally, the numerical results show that the proposed algorithm can effectively find the globally optimal solutions of all test instances with the given tolerance ϵ.

Compared with the known methods, the main advantages of the proposed algorithm are given as follows. Firstly, the branching search takes place in the image space $\mathbb{R}^p$ of the affine function $\sum_{i=1}^{n} e_{ji}y_i + f_j$, rather than in the variable dimension space $\mathbb{R}^n$, since n usually far exceeds p, which mitigates the required computational efforts. Secondly, some image region reduction techniques are proposed based on the characteristics of the objective function of the equivalent problem (EP) and the structure of the branch-and-bound algorithm, which provide a possibility to compress the investigated region, so that we can improve the computational efficiency of the algorithm. Thirdly, by analysing the computational complexity of the algorithm, we estimate the maximum iterations of the proposed algorithm for the first time. Finally, numerical comparison demonstrates the superiority of the algorithm.

The paper is organized as follows. In Section 2, the equivalence problem (EP) of the problem (MP) and its parametric linear relaxation programming problem (PLRP) are established. In Section 3, some image region reduction techniques are presented for improving the convergence speed of the algorithm. In Section 4, we propose an image space branch-reduction-bound algorithm, the global convergence of the algorithm is proved, and the complexity of the algorithm is analyzed. Numerical comparisons of some test examples in recent literatures are reported in Section 5. Finally, some conclusions are presented in Section 6.

2. Equivalent problem and its parametric linear relaxation programming

In order to globally solve the problem (MP), we first convert the problem (MP) into an equivalent problem (EP). Without losing generality, for each $j = 1, 2, \ldots, p$, let $l_j^0 = \min_{y \in D} \sum_{i=1}^{n} e_{ji}y_i + f_j$ and

$u_j^0 = \max_{y\in D} \sum_{i=1}^n e_{ji}y_i + f_j$. Since $\sum_{i=1}^n e_{ji}y_i + f_j$ is an affine function, we can get the values of l_j^0 and u_j^0 by solving the corresponding linear programming problems.

Let $\Lambda^0 = \{z \in \mathbb{R}^p | l_j^0 \le z_j \le u_j^0, j = 1, 2, \ldots, p\}$, by taking the logarithm of the objective function of the problem (MP) and introducing new variables $z_j = \sum_{i=1}^n e_{ji}y_i + f_j,\ j = 1, 2, \ldots, p$, we can convert the problem (MP) into the equivalent problem (EP) as follows:

$$(\mathrm{EP}): \begin{cases} \min & \phi(z) = \sum_{j=1}^p \alpha_j \ln z_j, \\ \text{s.t.} & z_j = \sum_{i=1}^n e_{ji}y_i + f_j, \quad j = 1, 2, \ldots, p, \\ & y \in D, z \in \Lambda^0. \end{cases}$$

It is obvious that, if $\left(y^*, z_1^*, z_2^*, \ldots, z_p^*\right)$ is a global optimal solution of the problem (EP), then y^* is a global optimal solution of the problem (MP), where $z_j^* = \sum_{i=1}^n e_{ji}y_i^* + f_j, j = 1, 2, \ldots, p$. On the contrary, if y^* is a global optimal solution of the problem (MP), let $z_j^* = \sum_{i=1}^n e_{ji}y_i^* + f_j, j = 1, 2, \ldots, p$, then $\left(y^*, z_1^*, z_2^*, \ldots, z_p^*\right)$ is a global optimal solution of the problem (EP).

In the following, for globally solving the problem (MP), we can solve its equivalent problem (EP) instead. Next, to design the branch-and-bound algorithm for solving the problem (EP), we need to construct the parametric linear relaxation programming of the problem (EP), which can provide a reliable lower bound for the global optimal value of the problem (EP).

Without losing generality, for any $\Lambda = \{z \in \mathbb{R}^p | l_j \le z_j \le u_j, j = 1, 2, \ldots, p\} \subseteq \Lambda^0$, for any $z_j \in [l_j, u_j], j = 1, 2, \ldots, p$, let $\varphi_j(z_j) = \ln z_j$, by the first-order differential median theorem, $\exists \xi_j \in [l_j, u_j]$, s.t. $\ln z_j = \ln l_j + \frac{1}{\xi_j}(z_j - l_j)$, so that we have

$$\ln z_j \ge \ln l_j + \frac{1}{u_j}(z_j - l_j) \text{ and } \ln z_j \le \ln l_j + \frac{1}{l_j}(z_j - l_j).$$

Similarly, for any $\Lambda = \{z \in \mathbb{R}^p | l_j \le z_j \le u_j, j = 1, 2, \ldots, p\} \subseteq \Lambda^0$, for any $z_j \in [l_j, u_j], j = 1, 2, \ldots, p$, by the first-order differential median theorem, $\exists \eta_j \in [l_j, u_j]$, s.t. $\ln z_j = \ln u_j + \frac{1}{\eta_j}(z_j - u_j)$, so that we have

$$\ln z_j \ge \ln u_j + \frac{1}{l_j}(z_j - u_j) \text{ and } \ln z_j \le \ln u_j + \frac{1}{u_j}(z_j - u_j).$$

By the above discussions, for any $z_j \in [l_j, u_j], j = 1, 2, \ldots, p$, we can obtain that

$$\ln z_j \ge \max\left\{\ln l_j + \frac{1}{u_j}(z_j - l_j), \ln u_j + \frac{1}{l_j}(z_j - u_j)\right\} = \varphi_j^l(z_j),$$
$$\ln z_j \le \min\left\{\ln l_j + \frac{1}{l_j}(z_j - l_j), \ln u_j + \frac{1}{u_j}(z_j - u_j)\right\} = \varphi_j^u(z_j).$$

Therefore, for any $z \in \Lambda \subseteq \Lambda^0$, we have

$$\varphi_j^l(z_j) \le \varphi_j(z_j) \le \varphi_j^u(z_j), j = 1, 2, \ldots, p.$$

Theorem 2.1. *For any $z \in \Lambda = \{z \in \mathbb{R}^p | l_j \le z_j \le u_j, j = 1, 2, \ldots, p\}$, let $\Delta_j^1(z_j) = \varphi_j(z_j) - \varphi_j^l(z_j)$, $\Delta_j^2(z_j) = \varphi_j^u(z_j) - \varphi_j(z_j)$, consider the functions $\varphi_j^l(z_j)$, $\varphi_j(z_j)$, and $\varphi_j^u(z_j)$, $j = 1, 2, \ldots, p$, then we have*

$$\lim_{\|u-l\|\to 0} \max_{z\in\Lambda} \Delta_j^1(z_j) = \lim_{\|u-l\|\to 0} \max_{z\in\Lambda} \Delta_j^2(z_j) = 0.$$

Proof. For any $z_j \in [l_j, u_j], j = 1, 2, \ldots, p$, by the first-order differential median theorem, we know that $\exists\ \xi_j \in [l_j, u_j]$, s.t. $\ln z_j - \ln l_j = \frac{1}{\xi_j}(z_j - l_j)$ and $\exists\ \eta_j \in [l_j, u_j]$, s.t. $\ln z_j - \ln u_j = \frac{1}{\eta_j}(z_j - u_j)$. Thus, we can obtain that

$$\Delta_j^1(z_j) = \varphi_j(z_j) - \varphi_j^l(z_j)$$

$$
\begin{aligned}
&= \ln z_j - \max\left\{\ln l_j + \frac{1}{u_j}(z_j - l_j), \ln u_j + \frac{1}{l_j}(z_j - u_j)\right\} \\
&= \min\left\{\ln z_j - \ln l_j - \frac{1}{u_j}(z_j - l_j), \ln z_j - \ln u_j - \frac{1}{l_j}(z_j - u_j)\right\} \\
&= \min\left\{\frac{1}{\xi_j}(z_j - l_j) - \frac{1}{u_j}(z_j - l_j), \frac{1}{\eta_j}(z_j - u_j) - \frac{1}{l_j}(z_j - u_j)\right\} \\
&= \min\left\{\left(\frac{1}{\xi_j} - \frac{1}{u_j}\right)(z_j - l_j), \left(\frac{1}{\eta_j} - \frac{1}{l_j}\right)(z_j - u_j)\right\} \\
&= \min\left\{\frac{1}{\xi_j u_j}(u_j - \xi_j)(z_j - l_j), \frac{1}{\eta_j l_j}(\eta_j - l_j)(u_j - z_j)\right\} \\
&\le \min\left\{\frac{1}{l_j u_j}(u_j - l_j)(u_j - l_j), \frac{1}{(l_j)^2}(u_j - l_j)(u_j - l_j)\right\} \\
&= \frac{1}{l_j u_j}(u_j - l_j)^2,
\end{aligned}
$$

since $\Delta_j^1(z_j) \to 0$ as $u_j - l_j \to 0$, we can obtain that $\lim_{\|u-l\|\to 0} \max_{z\in\Lambda} \Delta_j^1(z_j) = 0$.

Similarly, we can get that

$$
\begin{aligned}
\Delta_j^2(z_j) &= \varphi_j^u(z_j) - \varphi_j(z_j) \\
&= \min\left\{\ln l_j + \frac{1}{l_j}(z_j - l_j), \ln u_j + \frac{1}{u_j}(z_j - u_j)\right\} - \ln z_j \\
&= \min\left\{\ln l_j - \ln z_j + \frac{1}{l_j}(z_j - l_j), \ln u_j - \ln z_j + \frac{1}{u_j}(z_j - u_j)\right\} \\
&= \min\left\{-\frac{1}{\xi_j}(z_j - l_j) + \frac{1}{l_j}(z_j - l_j), -\frac{1}{\eta_j}(z_j - u_j) + \frac{1}{u_j}(z_j - u_j)\right\} \\
&= \min\left\{\left(\frac{1}{l_j} - \frac{1}{\xi_j}\right)(z_j - l_j), \left(\frac{1}{u_j} - \frac{1}{\eta_j}\right)(z_j - u_j)\right\} \\
&= \min\left\{\frac{1}{\xi_j l_j}(\xi_j - l_j)(z_j - l_j), \frac{1}{u_j \eta_j}(u_j - \eta_j)(u_j - z_j)\right\} \\
&\le \min\left\{\frac{1}{(l_j)^2}(u_j - l_j)(u_j - l_j), \frac{1}{l_j u_j}(u_j - l_j)(u_j - l_j)\right\} \\
&= \frac{1}{l_j u_j}(u_j - l_j)^2,
\end{aligned}
$$

since $\Delta_j^2(z_j) \to 0$ as $u_j - l_j \to 0$, we can get that $\lim_{\|u-l\|\to 0} \max_{z\in\Lambda} \Delta_j^2(z_j) = 0$.

Taken together above, we can obtain $\lim_{\|u-l\|\to 0} \max_{z\in\Lambda} \Delta_j^1(z_j) = \lim_{\|u-l\|\to 0} \max_{z\in\Lambda} \Delta_j^2(z_j) = 0$, and the proof is complete. □

Next, let $\psi(z)$ denote the underestimating function of the function $\phi(z)$, by Theorem 2.1, we have

$$
\phi(z) = \sum_{j=1}^{p} \alpha_j \ln z_j \ge \sum_{j=1,\alpha_j>0}^{p} \alpha_j \varphi_j^l(z_j) + \sum_{j=1,\alpha_j<0}^{p} \alpha_j \varphi_j^u(z_j) = \psi(z).
$$

Therefore, by the above discussions, and introduce parameters $s_j, j = 1, 2, \ldots, p$, we can construct the parametric linear relaxation programming problem (PLRP) of the problem (EP) as follows:

$$(\mathrm{PLRP}): \begin{cases} \min & \psi(z) = \sum_{j=1,\alpha_j>0}^{p} \alpha_j s_j + \sum_{j=1,\alpha_j<0}^{p} \alpha_j s_j, & \\ \text{s.t.} & \ln l_j + \frac{1}{u_j}(z_j - l_j) \le s_j, & \text{if } \alpha_j > 0, j = 1, 2, \ldots, p, \\ & \ln u_j + \frac{1}{l_j}(z_j - u_j) \le s_j, & \text{if } \alpha_j > 0, j = 1, 2, \ldots, p, \\ & \ln l_j + \frac{1}{l_j}(z_j - l_j) \ge s_j, & \text{if } \alpha_j < 0, j = 1, 2, \ldots, p, \\ & \ln u_j + \frac{1}{u_j}(z_j - u_j) \ge s_j, & \text{if } \alpha_j < 0, j = 1, 2, \ldots, p, \\ & z_j = \sum_{i=1}^{n} e_{ji} y_i + f_j, & j = 1, 2, \ldots, p, \\ & y \in D, z \in \Lambda. & \end{cases}$$

It is obvious that the optimal value of the parametric linear relaxation programming problem (PLRP) is less than or equal to the optimal value of the problem (EP).

3. Image space region reduction techniques

In order to improve the convergence speed of the algorithm, we propose some image space region reduction techniques, which provide a possibility to reduce the current rectangle to a smaller rectangle or delete the rectangle without losing the global optimal solution of the problem (EP). Thus, without losing generality, we assume that $\overline{\mathrm{UB}}$ is a best currently known upper bound of the global optimal value of the problem (EP), for any $z \in \Lambda = \{z \in \mathbb{R}^p : l_j \le z_j \le u_j, j = 1, 2, \ldots, p\} \subseteq \Lambda^0$, we let

$$\widehat{\theta} = \sum_{j=1,\alpha_j>0}^{p} \alpha_j \ln l_j + \sum_{j=1,\alpha_j<0}^{p} \alpha_j \ln u_j.$$

The specific region reduction process is given by the following theorem.

Theorem 3.1. *For any $z \in \Lambda = \{z \in \mathbb{R}^p : l_j \le z_j \le u_j, j = 1, 2, \ldots, p\} \subseteq \Lambda^0$, the following conclusions hold: if $\widehat{\theta} > \overline{\mathrm{UB}}$, then there exists no globally optimal solution of the problem (EP) over Λ; otherwise, we have the following two cases: if $\alpha_\tau > 0$ for some index $\tau \in \{1, 2, \ldots, p\}$ and $l_\tau \le \exp\left(\frac{\overline{\mathrm{UB}} - \widehat{\theta} + \alpha_\tau \ln l_\tau}{\alpha_\tau}\right) \le u_\tau$, then there exists no globally optimal solution of the problem (EP) over $\widehat{\Lambda}^1$; if $\alpha_\tau < 0$ for some index $\tau \in \{1, 2, \ldots, p\}$ and $l_\tau \le \exp\left(\frac{\overline{\mathrm{UB}} - \widehat{\theta} + \alpha_\tau \ln u_\tau}{\alpha_\tau}\right) \le u_\tau$, then there exists no globally optimal solution of the problem (EP) over $\widehat{\Lambda}^2$, where*

$$\widehat{\Lambda}^1 = \begin{cases} l_j \le z_j \le u_j, & j = 1, 2, \ldots, p, j \ne \tau, \\ \exp\left(\frac{\overline{\mathrm{UB}} - \widehat{\theta} + \alpha_\tau \ln l_\tau}{\alpha_\tau}\right) < z_\tau \le u_\tau, & j = \tau, \end{cases}$$

and

$$\widehat{\Lambda}^2 = \begin{cases} l_j \le z_j \le u_j, & j = 1, 2, \ldots, p, j \ne \tau, \\ l_\tau \le z_\tau < \exp\left(\frac{\overline{\mathrm{UB}} - \widehat{\theta} + \alpha_\tau \ln u_\tau}{\alpha_\tau}\right), & j = \tau. \end{cases}$$

Proof. If $\widehat{\theta} > \overline{\mathrm{UB}}$, then we have that

$$\min_{z \in \Lambda} \phi(z) = \min_{z \in \Lambda} \sum_{j=1}^{p} \alpha_j \ln z_j = \sum_{j=1,\alpha_j>0}^{p} \alpha_j \ln l_j + \sum_{j=1,\alpha_j<0}^{p} \alpha_j \ln u_j = \widehat{\theta} > \overline{\mathrm{UB}}.$$

Therefore, there exists no globally optimal solution of the problem (EP) over Λ.

If $\widehat{\theta} \leq \overline{\mathrm{UB}}$, for any $z \in \widehat{\Lambda}^1$, consider the τ_{th} component z_τ of z, we have $\exp\left(\frac{\overline{\mathrm{UB}}-\widehat{\theta}+\alpha_\tau \ln l_\tau}{\alpha_\tau}\right) < z_\tau \leq u_\tau$, Thus, we can obtain

$$z_\tau > \exp\left(\frac{\overline{\mathrm{UB}} - \widehat{\theta} + \alpha_\tau \ln l_\tau}{\alpha_\tau}\right),$$

since $\alpha_\tau > 0$, and taking the logarithm of both sides of the above formula, we can follow that

$$\alpha_\tau \ln z_\tau > \overline{\mathrm{UB}} - \widehat{\theta} + \alpha_\tau \ln l_\tau.$$

Also because

$$\widehat{\theta} = \sum_{j=1,\alpha_j>0}^{p} \alpha_j \ln l_j + \sum_{j=1,\alpha_j<0}^{p} \alpha_j \ln u_j,$$

we can follow that

$$\begin{aligned}\min_{z\in\widehat{\Lambda}^1} \phi(z) &= \min_{z\in\widehat{\Lambda}^1} \left(\sum_{j=1}^{p} \alpha_j \ln z_j\right) > \sum_{j=1,j\neq\tau,\alpha_j>0}^{p} \alpha_j \ln l_j + \sum_{j=1,j\neq\tau,\alpha_j<0}^{p} \alpha_j \ln u_j + \alpha_\tau \ln\left(\exp\left(\frac{\overline{\mathrm{UB}} - \widehat{\theta} + \alpha_\tau \ln l_\tau}{\alpha_\tau}\right)\right) \\ &= \widehat{\theta} - \alpha_\tau \ln l_\tau + \overline{\mathrm{UB}} - \widehat{\theta} + \alpha_\tau \ln l_\tau \\ &= \overline{\mathrm{UB}}.\end{aligned}$$

Thus, there exists no globally optimal solution of the problem (EP) over $\widehat{\Lambda}^1$.

For any $z \in \widehat{\Lambda}^2$, consider the τ_{th} component z_τ of z, we have $l_\tau \leq z_\tau < \exp\left(\frac{\overline{\mathrm{UB}}-\widehat{\theta}^\tau+\alpha_\tau \ln u_\tau}{\alpha_\tau}\right)$, Thus, we can obtain

$$z_\tau < \exp\left(\frac{\overline{\mathrm{UB}} - \widehat{\theta} + \alpha_\tau \ln u_\tau}{\alpha_\tau}\right),$$

since $\alpha_\tau < 0$, and taking the logarithm of both sides of the above formula, we can follow that

$$\alpha_\tau \ln z_\tau > \overline{\mathrm{UB}} - \widehat{\theta} + \alpha_\tau \ln u_\tau.$$

Also because

$$\widehat{\theta} = \sum_{j=1,\alpha_j>0}^{p} \alpha_j \ln l_j + \sum_{j=1,\alpha_j<0}^{p} \alpha_j \ln u_j,$$

Thus, we can follow that

$$\begin{aligned}\min_{z\in\widehat{\Lambda}^2} \phi(z) &= \min_{z\in\widehat{\Lambda}^2} \left(\sum_{j=1}^{p} \alpha_j \ln z_j\right) > \sum_{j=1,j\neq\tau,\alpha_j>0}^{p} \alpha_j \ln l_j + \sum_{j=1,j\neq\tau,\alpha_j<0}^{p} \alpha_j \ln u_j + \alpha_\tau \ln\left(\exp\left(\frac{\overline{\mathrm{UB}} - \widehat{\theta} + \alpha_\tau \ln u_\tau}{\alpha_\tau}\right)\right) \\ &= \widehat{\theta} - \alpha_\tau \ln u_\tau + \overline{\mathrm{UB}} - \widehat{\theta} + \alpha_\tau \ln u_\tau \\ &= \overline{\mathrm{UB}}.\end{aligned}$$

Thus, there exists no globally optimal solution of the problem (EP) over $\widehat{\Lambda}^2$, and the proof is complete. □

According to Theorem 3.1, we can construct some image space region reduction techniques, which can provide the possibility for deleting the whole investigated rectangle Λ or part of it, in which there is no globally optimal solution of the problem (EP).

4. Algorithm and its computational complexity

In this section, we firstly present a fundamental branching rule. Next, we propose an image space branch-reduction-bound algorithm for solving the problem (EP) based on the former parametric linear relaxation programming problem. Furthermore, the global convergence of the algorithm is proved, and the computational complexity of the algorithm is analysed.

4.1. Branching rule

Consider the rectangle $\Lambda = \{z \in \mathbb{R}^p | l_j \le z_j \le u_j, j = 1, 2, \ldots, p\} \subseteq \Lambda^0$, the proposed branching rule is given as follows:

(i) let $q = \arg\max\{u_j - l_j | j = 1, 2, \ldots, p\}$;
(ii) let

$$\Lambda^1 = \left\{ z \in \mathbb{R}^p | l_j \le z_j \le u_j, j \ne q, j = 1, 2, \ldots, p;\ l_j \le z_j \le \frac{l_j + u_j}{2}, j = q \right\}$$

and

$$\Lambda^2 = \left\{ z \in \mathbb{R}^p | l_j \le z_j \le u_j, j \ne q, j = 1, 2, \ldots, p;\ \frac{l_j + u_j}{2} \le z_j \le u_j, j = q \right\}.$$

By the branching rule, the rectangle Λ can be subdivided into two new sub-rectangles Λ^1 and Λ^2.

4.2. Algorithm statement

Let $\mathrm{LB}(\Lambda)$ and $(y(\Lambda), z(\Lambda))$ be the optimal value and the optimal solution of the (PLRP) over the sub-rectangle Λ, respectively. Next, the basic steps of the proposed algorithm can be summarized as follows.

Step 1. Initialize the set all active nodes $\Omega_0 = \{\Lambda^0\}$, the upper bound $\mathrm{UB} = +\infty$, the set of feasible points $F = \emptyset$, the termination tolerance $\epsilon > 0$, and the iteration counter $k = 0$.
Solve the problem (PLRP) for $\Lambda = \Lambda^0$. Let $\mathrm{LB}_0 = \mathrm{LB}(\Lambda^0)$, $(y^0, z^0) = \big(y(\Lambda^0), z\big(\Lambda^0\big)\big)$, $\mathrm{UB} = \phi\big(z^0\big)$, and $F = F \bigcup \big\{\big(y^0, z^0\big)\big\}$.
If $\mathrm{UB} - \mathrm{LB}_0 \le \epsilon$, then the algorithm stops: (y^0, z^0) is a global ϵ-optimal solution of the problem (EP). Otherwise, proceed with Step 2.

Step 2. Use the proposed branching rule to subdivide Λ^k into two new sub-rectangles, and denote the set of new partitioning sub-rectangles as $\overline{\Lambda}^k$. For each $\Lambda \in \overline{\Lambda}^k$, use the proposed image space region reduction techniques to reduce the sub-rectangle Λ, and still denote the remaining sub-rectangles and their set as Λ and $\overline{\Lambda}^k$, respectively.

Step 3. If $\overline{\Lambda}^k \ne \emptyset$, for each $\Lambda \in \overline{\Lambda}^k$, solve the following improved parametric linear relaxation programming problem:

$$(\mathrm{IPLRP}): \begin{cases} \min & \psi(z) = \sum_{j=1,\alpha_j>0}^{p} \alpha_j s_j + \sum_{j=1,\alpha_j<0}^{p} \alpha_j s_j, & \\ \text{s.t.} & \ln l_j + \frac{1}{u_j}(z_j - l_j) \le s_j, & \text{if } \alpha_j > 0, j = 1, 2, \ldots, p, \\ & \ln u_j + \frac{1}{l_j}(z_j - u_j) \le s_j, & \text{if } \alpha_j > 0, j = 1, 2, \ldots, p, \\ & \ln l_j + \frac{1}{l_j}(z_j - l_j) \ge s_j, & \text{if } \alpha_j < 0, j = 1, 2, \ldots, p, \\ & \ln u_j + \frac{1}{u_j}(z_j - u_j) \ge s_j, & \text{if } \alpha_j < 0, j = 1, 2, \ldots, p, \\ & z_j = \sum_{i=1}^{n} e_{ji} y_i + f_j, & j = 1, 2, \ldots, p, \\ & \sum_{j=1,\alpha_j>0}^{p} \alpha_j s_j + \sum_{j=1,\alpha_j<0}^{p} \alpha_j s_j \le \mathrm{UB}, & \\ & y \in D, z \in \Lambda, & \end{cases}$$

we can obtain $\mathrm{LB}(\Lambda)$ and $(y(\Lambda), z(\Lambda))$.

If $\mathrm{LB}(\Lambda) > \mathrm{UB}$, set $\overline{\Lambda}^k = \overline{\Lambda}^k \backslash \Lambda$. Otherwise, update the upper bound UB, the feasible point set F, and the best feasible solution (y^k, z^k).

Denote the remaining partitioning set as $\Omega_k = (\Omega_k \backslash \Lambda) \bigcup \overline{\Lambda}^k$, and denote the new lower bound as $\mathrm{LB}_k = \inf_{\Lambda \in \Omega_k} \mathrm{LB}(\Lambda)$.

Step 4. Let $\Omega_{k+1} = \Omega_k \backslash \{\Lambda | \mathrm{UB} - \mathrm{LB}(\Lambda) \leq \epsilon, \Lambda \in \Omega_k\}$. If $\Omega_{k+1} = \emptyset$, then the algorithm stops: UB and (y^k, z^k) are the global ϵ-optimal value and the global ϵ-optimal solution of the problem (EP), respectively. Otherwise, select an active node Λ^{k+1} such that $\Lambda^{k+1} = \arg\min_{\Lambda \in \Omega_{k+1}} \mathrm{LB}(\Lambda)$, let $k = k + 1$, and return to Step 2 of the algorithm.

Remark. In fact, the objective function of the problem (PLRP) is always less than or equal to the currently known upper bound, when we add this condition to the problem (PLRP) as a linear constraint, we can get an improved parameter linear relaxation programming problem (IPLRP).

4.3. Global convergence analysis

In the subsection, we will discuss the global convergence of the above algorithm.

Theorem 4.1. *The presented algorithm either terminates finitely at the global optimum solution of the problem (MP) or generates an infinite sequence $\{y^k\}$ of which any gathering point of the sequence $\{y^k\}$ is a global optimal solution for the problem (MP).*

Proof. If the presented algorithm is finite, by steps of the presented algorithm, then it terminates at some step $k \geq 0$. According to the termination condition of the algorithm, we have that

$$\mathrm{UB} - \mathrm{LB}_k \leq \epsilon.$$

By Steps 1 and 4 of the presented algorithm, a better feasible solution $\left(y^k, z^k\right)$ for the problem (EP) can be found, and we have the following inequality:

$$\phi\left(z^k\right) - \mathrm{LB}_k \leq \epsilon.$$

Let v be the global optimal value of the problem (EP), we have that

$$\mathrm{LB}_k \leq v.$$

Since $\left(y^k, z^k\right)$ is a feasible solution for the problem (EP), we can obtain that

$$\phi\left(z^k\right) \geq v.$$

Taken together above, it implies that

$$v \leq \phi\left(z^k\right) \leq \mathrm{LB}_k + \epsilon \leq v + \epsilon.$$

So $\left(y^k, z^k\right)$ is a global ϵ-optimal solution for the problem (EP) in the sense that

$$v \leq \phi\left(z^k\right) \leq v + \epsilon.$$

Hence, when the presented algorithm finitely terminates after k iterations, it follows that y^k is a global ϵ-optimal solution for the problem (MP).

If the presented algorithm creates an infinite sequence $\{(y^k, z^k)\}$ of solution which is obtained by solving the problem (PLRP) over Λ^k, then it is obvious that $\{(y^k, z^k)\}$ is a sequence of feasible solution for the problem (EP) over Λ^0, where $z_j^k = \sum_{i=1}^n e_{ji} y_i^k + f_j, j = 1, 2, \ldots, p$. Let (y^*, z^*) be a gathering point of the sequence $\{\left(y^k, z^k\right)\}$, and we have

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

So we have

$$\lim_{k\to\infty} z_j^k = \lim_{k\to\infty} \sum_{i=1}^{n} e_{ji} y_i^k + f_j = \sum_{i=1}^{n} e_{ji} y_i^* + f_j = z_j^*,\ j = 1, 2, \ldots, p.$$

By the continuity of $\sum_{i=1}^{n} e_{ji} y_i + f_j$, $\sum_{i=1}^{n} e_{ji} y_i^k + f_j = z_j^k \in [l_j^k, u_j^k], j = 1, 2, \ldots, p$, and the exhaustiveness of the partitioning method, we can obtain that

$$\sum_{i=1}^{n} e_{ji} y_i^* + f_j = \lim_{k\to\infty} \sum_{i=1}^{n} e_{ji} y_i^k + f_j = \lim_{k\to\infty} z_j^k = \lim_{k\to\infty} [l_j^k, u_j^k] = \lim_{k\to\infty} \bigcap_k [l_j^k, u_j^k] = z_j^*.$$

Thus, (y^*, z^*) is a feasible solution for the problem (EP) over Λ^0. Since $\{\mathrm{LB}(\Lambda^k)\}$ is a real increasing sequence, we have that

$$\lim_{k\to\infty} \mathrm{LB}(\Lambda^k) \le \phi(z^*).$$

From the renewal process of lower bound and Theorem 2.1, we have that

$$\lim_{k\to\infty} \mathrm{LB}(\Lambda^k) = \lim_{k\to\infty} \psi(z^k) \text{ and } \lim_{k\to\infty} \psi(z^k) = \lim_{k\to\infty} \phi(z^k) = \phi(z^*).$$

Therefore, we can get that

$$\lim_{k\to\infty} \mathrm{LB}(\Lambda^k) = \phi(z^*).$$

Therefore, (y^*, z^*) is a global optimum solution for the problem (EP) over Λ^0. At the same time, by the equivalence conclusions of the problems (EP) and (MP), it follows that y^* is a global optimum solution for the problem (MP), and the proof of the theorem is completed. □

4.4. Computational complexity of the algorithm

In this subsection, we will analyze the computational complexity of the presented algorithm for estimating the maximum iterations of the algorithm. To this end, we define the size $\delta(\Lambda)$ of a rectangle

$$\Lambda = \{z \in \mathbb{R}^p : l_j \le z_j \le u_j,\ j = 1, 2, \ldots, p\}$$

as

$$\delta(\Lambda) := \max\{u_j - l_j |\ j = 1, 2, \ldots, p\},$$

and for convenience, we also define

$$\mu = \max_{j=1,\ldots,p} |\alpha_i| \text{ and } \sigma = \max_{j=1,\ldots,p} \frac{1}{(l_j^0)^2}.$$

Theorem 4.2. *For any given termination tolerance $\epsilon > 0$, for any $\Lambda \subseteq \Lambda^0$, if $\delta^2(\Lambda) \le \epsilon$, then, for any feasible solution (y, z) to the problem (PLRP) over Λ, we have that*

$$|\phi(z) - \psi(z)| \le p\mu\sigma\epsilon.$$

Proof. For any feasible solution (y, z) to the problem (PLRP) over Λ, it is obvious that (y, z) is a feasible solution to the problem (EP) over Λ. From Theorem 2.1, for any $z \in \Lambda$, if $\delta^2(\Lambda) \le \epsilon$ for any sufficiently small positive number ϵ, we have that

$$|\phi(z) - \psi(z)| = \sum_{j=1}^{p} \alpha_j \ln z_j - \left[\sum_{j=1,\alpha_j>0}^{p} \alpha_j s_j + \sum_{j=1,\alpha_j<0}^{p} \alpha_j s_j \right]$$

$$
\begin{aligned}
&= \sum_{j=1,\alpha_j>0}^{p} \alpha_j \left[\ln z_j - \max\left\{ \ln l_j + \frac{1}{u_j}(z_j - l_j), \ln u_j + \frac{1}{l_j}(z_j - u_j) \right\} \right] \\
&\quad + \sum_{j=1,\alpha_j<0}^{p} \alpha_j \left[\ln z_j - \min\left\{ \ln l_j + \frac{1}{l_j}(z_j - l_j), \ln u_j + \frac{1}{u_j}(z_j - u_j) \right\} \right] \\
&= \sum_{j=1,\alpha_j>0}^{p} \alpha_j \left[\varphi_j(z_j) - \varphi_j^l(z_j) \right] - \sum_{j=1,\alpha_j<0}^{p} \alpha_j \left[\varphi_j^u(z_j) - \varphi_j(z_j) \right] \\
&\le \sum_{j=1}^{p} |\alpha_j| \times \frac{1}{l_j u_j}(u_j - l_j)^2 \\
&\le \sum_{j=1}^{p} |\alpha_j| \times \frac{1}{\left(l_j^0\right)^2}(u_j - l_j)^2 \\
&\le p\mu\sigma\delta^2(\Lambda) \\
&\le p\mu\sigma\epsilon,
\end{aligned}
$$

and the proof of the theorem is completed. □

Theorem 4.3. *For any given termination tolerance $\epsilon \in (0,1)$, the presented algorithm can obtain a global ϵ-optimal solution in at most*

$$p \cdot \left\lceil \frac{1}{2} \log_2 \frac{p\mu\sigma\delta^2\left(\Lambda^0\right)}{\epsilon} \right\rceil$$

iterations.

Proof. Without losing generality, we assume that, at each iteration, the rectangle partition is performed in the selected sub-rectangle Λ at step 2 of the algorithm. After $k \cdot p$ iterations, we have that

$$\delta(\Lambda) \le \frac{1}{2^k}\delta\left(\Lambda^0\right),$$

thus, we can obtain that

$$\delta^2(\Lambda) \le \frac{1}{2^{2k}}\delta^2\left(\Lambda^0\right),$$

by the proof of Theorem 4.2, if

$$\frac{1}{2^{2k}}\delta^2\left(\Lambda^0\right) \le \frac{\epsilon}{p\mu\sigma},$$

i.e.,

$$k \ge \frac{1}{2} \log_2 \frac{p\mu\sigma\delta^2\left(\Lambda^0\right)}{\epsilon},$$

we can get that $|\phi(z) - \psi(z)| \le \epsilon$. Therefore, after at most

$$p \cdot \left\lceil \frac{1}{2} \log_2 \frac{p\mu\sigma\delta^2\left(\Lambda^0\right)}{\epsilon} \right\rceil$$

iterations, we have that

$$0 \le \phi(z) - \phi(z^*) \le \phi(z) - \mathrm{LB}(\Lambda) = |\phi(z) - \psi(z)| \le \epsilon,$$

where (y^*, z^*) is the global optimal solution for the problem (EP). From Step 3 of the proposed algorithm, (y^k, z^k) is the current best feasible point for the problem (EP), and by Theorem 4.2, we know that $\{\phi(z^k)\}$ is a decreasing sequence and it satisfies $\phi(z^k) \leq \phi(z)$. Therefore, we have

$$\phi(z^k) - \phi(z^*) \leq \phi(z) - \phi(z^*) \leq \epsilon,$$

which shows that, when the presented algorithm terminates, (y^k, z^k) is a global ϵ-optimal solution for the problem (EP). Therefore, y^k is a global ϵ-optimal solution for the problem (MP), and the proof of the theorem is completed. □

5. Numerical experiments

In this section, we numerically compare our algorithm with the software BARON [18] and the known branch-and-bound algorithms [5,8,22,32,34,39]. All numerical tests are implemented in MATLAB R2014a and run on a microcomputer with Intel(R) Core(TM) i5-7200U CPU @2.50 GHz processor and 16 GB RAM. In all numerical tests, the maximum CPU time limit is set to 3600 s.

First of all, with the given convergence tolerance 10^{-6}, we tested some small-size examples, see Appendices A.1–A.8 for details, compared with the existing branch-and-bound algorithms [5, 8, 22, 32, 34, 39] and the software BARON, and the corresponding numerical comparisons are reported in Table 1.

Next, with the given convergence tolerance 10^{-6}, we tested some large-size examples, see Problems 5.1–5.3 for details, which are generated randomly to further verify the robustness and effectiveness of our algorithm, and the corresponding numerical comparisons are reported in Tables 2 and 4–7. Moreover, with the given convergence tolerance 10^{-2}, we further tested Problem 5.1 with the larger p, and the numerical comparison results are shown in Table 3.

Some notations have been used for column headers in Tables 1–7: Iter: the number of iteration for the algorithm; CPU Time: the execution CPU time of algorithm in seconds; Avg.N: the average number of iterations; Std.N: the standard deviations for number of iteration; Avg.T: the average execution CPU time of the algorithm in seconds; Std.T: the standard deviation of the execution CPU time of the algorithm; and "−" denotes the situation that some of ten random instances failed to be solved in 3600 s.

Problem 5.1 (Ref. [33]).

$$\begin{cases} \min & \prod_{j=1}^{p}(\sum_{i=1}^{n} e_{ji}y_i + 1), \\ \text{s.t.} & Ay \leq b, y \geq 0, \end{cases}$$

where $e_{ji}, j = 1, 2, \ldots, p,\ i = 1, \ldots, n$, are all randomly generated in $[0, 1]$, all components of the matrix A are randomly generated in $[-1, 1]$, and $b_i, i = 1, \ldots, n$, are all randomly generated by setting $b_i = \sum_{j=1}^{n} a_{ij} + 2\pi$, where π are randomly generated in $[0, 1]$.

Problem 5.2 (Ref. [39]).

$$\begin{cases} \min & \prod_{j=1}^{p} e_j^{\top} y, & \\ \text{s.t.} & \sum_{i=1}^{n} A_{si}y_i \leq b_s, & s = 1, 2, \ldots, m, \\ & 0 \leq y_i \leq 1, & i = 1, 2, \ldots, n, \end{cases}$$

where all components of the vector $e_j, j = 1, 2, \ldots, p$, are randomly generated in $[0, 1]$, $A_{si}, s = 1, 2, \ldots, m, i = 1, 2, \ldots, n$, are all randomly generated in $[-1, 1]$, and $b_s, s = 1, 2, \ldots, m$, are all generated by setting $b_s = \sum_{i=1}^{n} A_{si} + 2\pi$, where π is randomly generated in $[0, 1]$.

Problem 5.3.

$$\begin{cases} \min & \prod_{j=1}^{p}(\sum_{i=1}^{n} e_{ji}y_i + f_j)^{\alpha_j}, \\ \text{s.t.} & Ay \leq b, y \geq 0, \end{cases}$$

TABLE 1. Numerical comparisons among the software BARON, the algorithms of references [5, 8, 22, 32, 34, 39] and our algorithm on Appendices A.1–A.8.

Example	Refs.	Optimal value	Optimal solution	Iter	CPU time
1	Our	0.89019	(1.3148, 0.1396, 0.0, 0.4233)	1	0.04
	[34]	0.8902	(1.3148, 0.1396, 0.0, 0.4233)	1	0.19
	BARON	0.8902	(1.3148, 0.1396, 0.0, 0.4233)	3	0.05
2	Our	0.53333	(0.0, 0.0)	23	0.95
	[32]	0.53333	(0.0, 0.0)	16	0.05
	BARON	0.5333	(0.0, 0.0)	1	0.03
3	Our	10.0	(2.0, 8.0)	3	0.12
	[5]	10.0	(2.0, 8.0)	41	0.02
	BARON	10.0000	(2, 8)	1	0.05
4	Our	997.66127	(1, 1)	0	0.01
	[22]	997.66127	(1, 1)	3	0.09
	BARON	997.6613	(1, 1)	1	0.03
5	Our	263.78893	(1.25, 1)	1	0.01
	[22]	263.7889	(1.25, 1)	11	0.30
	BARON	263.7889	(1.25, 1)	3	0.03
6	Our	5.00931	(3, 2)	0	0.01
	[22]	5.00931	(3, 2)	2	0.05
	BARON	5.0093	(3, 2)	1	0.03
7	Our	0.90123	(0.0, 8.0, 1.0)	9	0.09
	[8]	0.901235	(0.0, 8.0, 1.0)	3	0.05
	BARON	0.9012	(0.0, 8.0, 1.0)	7	0.06
8	Our	9504.0	(1.0, 2.0, 1.0, 1.0, 1.0)	4	0.14
	[39]	9504.0	(1.0, 2.0, 1.0, 1.0, 1.0)	2	0.07
	BARON	9504.0	(1.0, 2.0, 1.0, 1.0, 1.0)	1	0.03

where $e_{ji}, j = 1, 2, \ldots, p, i = 1, 2, \ldots, n$, and $f_j, j = 1, 2, \ldots, p$, are all randomly generated in $[0, 1]$, all components of the matrix A and $\alpha_j, j = 1, 2, \ldots, p$, are all generated in $[-1, 1]$, and all the components of the vector b are generated by setting $b_i = \sum_{j=1}^{n} a_{ij} + 2\pi$, where π is randomly generated in $[0, 1]$.

According to the numerical results in Table 1, for small-size examples A.1–A.8 in appendix, it can be observed that, with almost the same computational efficiency, our algorithm can obtain the same optimal solutions and optimal values as the existing branch-and-bound algorithms [5, 8, 22, 32, 34, 39] and the software BARON [18]. At the same time, we can also observe that, when the variable $n \leq 5$, the software BARON is more effective for solving the problem (MP).

For large-size Problems 5.1–5.3, we solved ten independently generated instances and recorded their average number of iteration and average CPU time. For Problems 5.1–5.3, when $n > 20$, the software BARON failed to solve some of ten independently generated instances in 3600 s, so that we only numerically compared our algorithm with the algorithm of Jiao *et al.* [17] in Table 3, and we only numerically compared our algorithm with the algorithm of Zhang *et al.* [39] in Table 5, and we only reported the numerical results of our algorithm in Tables 4 and 7. Meanwhile, we use graphs to show the numerical comparison results of Tables 3 and 5 respectively, and see Figures 1–3 for details.

According to the numerical results in Table 2, it can be understood that, compared with the algorithm of Wang and Liu [33], our algorithm takes less average time and average number of iteration to solve Problem 5.1 with large-size variables, and the software BARON is only effective for Problem 5.1 with small-size variables, such as $p = 2$, $m \leq 10$ and $n \leq 20$. When $p = 2$, $m \geq 20$, and $n \geq 20$, the software BARON failed to

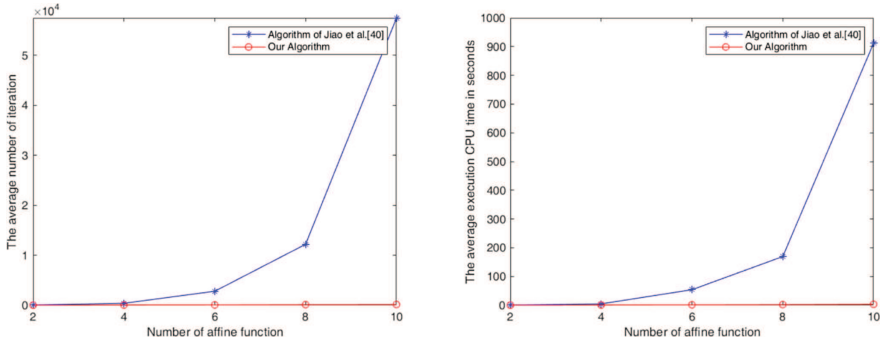

FIGURE 1. Numerical comparisons between our algorithm and that of Jiao *et al.* [17] on the average number of iteration (*left*) and the average CPU time (*right*) for Problem 5.1 with the fixed $m = 10$ and $n = 10$.

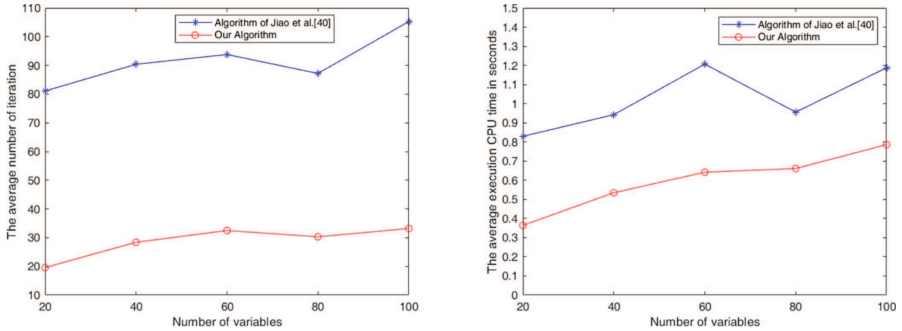

FIGURE 2. Numerical comparisons between our algorithm and that of Jiao *et al.* [17] on the average number of iteration (*left*) and the average CPU time (*right*) for Problem 5.1 with the fixed $p = 2$ and $m = 10$.

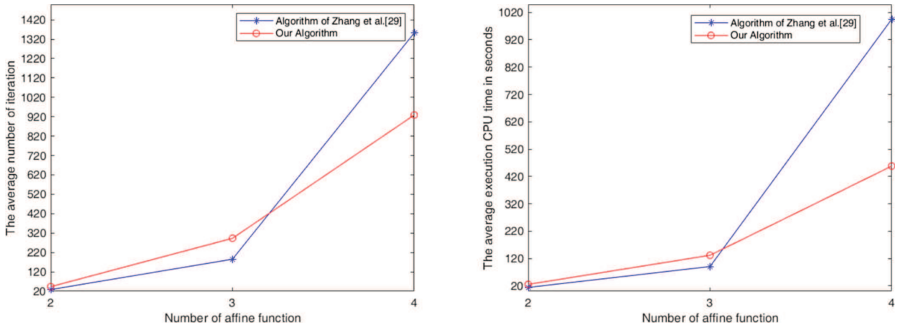

FIGURE 3. Numerical comparisons between our algorithm and that of Zhang *et al.* [39] on the average number of iteration (*left*) and the average CPU time (*right*) for Problem 5.2 with the fixed $m = 10$ and $n = 2000$.

TABLE 2. Numerical comparisons among the software BARON, algorithms of Wang and Liu [33], and our algorithm on Problem 5.1.

	Algorithm of Wang and Liu [33]		Our algorithm		BARON	
(p, m, n)	Avg(Std).N	Avg(Std).T	Avg(Std).N	Avg(Std).T	Avg(Std).N	Avg(Std).T
$(2, 10, 20)$	14.2(1.5492)	0.6062(0.0695)	27.0(14.0712)	1.0567(0.5579)	8.8(9.21)	0.1(0.0267)
$(2, 20, 20)$	17.4(1.7127)	0.8368(0.0756)	21.3(11.0860)	1.4858(0.9255)	–	–
$(2, 22, 20)$	18.5(1.9003)	0.9460(0.1235)	16.6(9.1797)	1.3410(0.7558)	–	–
$(2, 20, 30)$	19.9(0.5676)	1.0781(0.0674)	29.1(9.4569)	1.9719(1.0081)	–	–
$(2, 35, 50)$	21.2(0.4316)	1.8415(0.1338)	31.1(2.9981)	2.0209(0.3103)	–	–
$(2, 45, 60)$	23.0(0.6667)	2.4338(0.1016)	30.9(2.9981)	3.2201(0.5866)	–	–
$(2, 45, 100)$	35.7(1.1595)	5.1287(0.0935)	36.1(2.6437)	3.8516(0.5096)	–	–
$(2, 60, 100)$	36.1(0.7379)	6.8143(0.1713)	32.1(2.9981)	4.6820(0.6611)	–	–
$(2, 70, 100)$	36.6(1.2649)	8.1967(0.2121)	30.5(3.8370)	5.3423(0.8261)	–	–
$(2, 70, 120)$	39.1(1.6633)	9.5642(0.2975)	33.3(3.0569)	6.7911(0.9495)	–	–
$(2, 100, 100)$	37.5(2.1731)	13.0578(0.3543)	22.1(6.5735)	7.2817(2.1901)	–	–

TABLE 3. Numerical comparisons between our algorithm and that of Jiao *et al.* [17] on Problem 5.1.

	Algorithm of Jiao *et al.* [17]		Our algorithm	
(p, m, n)	Avg(Std).N	Avg(Std).T	Avg(Std).N	Avg(Std).T
$(2, 10, 10)$	51.6(27.1260)	0.5366(0.2774)	7.6(9.1797)	0.1440(0.1568)
$(2, 10, 20)$	81.1(29.0304)	0.8296(0.2990)	19.6(12.9974)	0.3642(0.2395)
$(2, 10, 40)$	90.4(24.3867)	0.9427(0.2610)	28.4(9.5475)	0.5349(0.1841)
$(2, 10, 60)$	93.8(27.3447)	1.2067(0.3675)	32.5(2.5927)	0.6421(0.0721)
$(2, 10, 80)$	87.2(23.3419)	0.9561(0.3115)	30.3(10.7708)	0.6612(0.2127)
$(2, 10, 100)$	105.3(43.7113)	1.1878(0.4902)	33.2(1.6193)	0.7862(0.0637)
$(4, 10, 10)$	385.6(304.8264)	4.6592(3.5951)	35.8(22.9918)	0.6465(0.3884)
$(4, 10, 20)$	1544.9(1034.5785)	15.3208(10.4886)	80.4(18.0012)	1.6899(0.4456)
$(4, 10, 40)$	3766.5(3714.9863)	41.3699(43.3325)	125.5(49.4711)	2.7247(1.4014)
$(4, 10, 60)$	3500.4(4733.4820)	42.8343(67.3455)	123.1(25.6924)	2.9937(0.7626)
$(4, 10, 80)$	6588.5(4183.8860)	108.5261(76.9548)	144.2(26.1950)	3.5831(0.6546)
$(4, 10, 100)$	4475.8(6161.0653)	82.6249(143.2454)	196.4(87.1183)	5.5485(2.6318)
$(6, 10, 10)$	2795.0(4542.3064)	53.9199(94.6386)	61.2(29.0777)	1.3197(0.7418)
$(6, 10, 20)$	27 365.3(31 428.9002)	426.2640(663.2382)	133.9(69.5453)	2.9194(1.6542)
$(6, 10, 40)$	66 717.5(73 801.7557)	1895.0217(3169.6473)	306.1(204.2659)	6.7800(4.7208)
$(8, 10, 10)$	12 192.2(17 724.0386)	169.3895(306.5079)	112.6(58.7995)	2.1026(1.2695)
$(8, 10, 20)$	79 305.9(68 019.9694)	1454.1710(1541.1460)	357.4(249.8036)	7.6240(5.5012)
$(10, 10, 10)$	57 421.5(54 655.0563)	911.9321(1098.6795)	135.6(72.0990)	3.2027(1.5347)

solve some of them in 3600 s. But, our algorithm can solve Problem 5.1 with large-size variables and has better computational performance.

According to the numerical results in Table 3, and from Figures 1 and 2, it can be seen that our algorithm takes less average time and average number of iteration than that of Jiao *et al.* [17] for solving Problem 5.1 with the larger p, so our algorithm is superior than the algorithm of Jiao *et al.* [17].

According to the numerical results in Tables 4 and 7, we can see that our algorithm can solve Problems 5.2 and 5.3 with the large-size variables, but the software BARON failed to solve some of them in 3600 s, so this indicates the effectiveness and robustness of our algorithm.

TABLE 4. Numerical computational results of our algorithm for Problem 5.2.

(m,n)		$p=4$	$p=5$	$p=6$	$p=7$
(10, 20)	Avg(Std).N	37.2(22.2051)	96.9(84.0918)	159.7(94.3752)	302.6(240.4973)
	Avg(Std).T	3.0331(2.3414)	6.8133(4.6833)	12.4855(9.3914)	18.3431(13.7249)
(20, 40)	Avg(Std).N	57.9(62.9964)	68.7(30.2032)	116.7(64.0365)	289.2(338.0614)
	Avg(Std).T	3.7998(3.9706)	5.1644(2.0187)	9.9100(4.9113)	23.0972(24.1184)
(30, 60)	Avg(Std).N	52.8(33.0178)	94.1(45.9213)	178.4(93.5524)	273.3(197.6355)
	Avg(Std).T	5.8097(4.5024)	10.4212(6.3453)	16.9114(8.2818)	25.4896(18.3698)
(40, 80)	Avg(Std).N	41.9(17.3554)	78.8(40.0078)	144.4(78.5765)	228.8(137.0230)
	Avg(Std).T	5.0930(2.1380)	10.3765(5.0349)	19.9002(10.7279)	33.8311(19.6794)
(50, 100)	Avg(Std).N	45.8(19.2111)	77.8(54.5034)	103.3(42.0504)	189.1(91.1780)
	Avg(Std).T	7.3425(3.6532)	13.1858(9.1603)	18.4209(7.6899)	35.1482(15.7116)
(60, 120)	Avg(Std).N	32.0(10.5409)	65.9(35.9829)	98.2(58.9742)	208.5(152.8240)
	Avg(Std).T	7.0007(2.5679)	14.8812(8.4005)	23.4874(15.0108)	50.3251(34.2680)
(70, 140)	Avg(Std).N	32.9(12.4226)	86.4(32.7353)	84.5(22.4215)	113.5(57.7163)
	Avg(Std).T	9.0893(3.2417)	28.4109(12.6269)	27.4017(7.6908)	37.3657(18.4767)
(80, 160)	Avg(Std).N	26.8(8.5088)	51.1(15.6521)	98.7(37.6152)	141.9(52.4032)
	Avg(Std).T	9.8870(2.9608)	20.2917(6.0380)	42.5319(16.5759)	66.4867(22.7845)
(90, 180)	Avg(Std).N	29.0(3.6515)	49.4(11.4134)	92.5(30.9237)	139.4(44.7492)
	Avg(Std).T	13.9006(2.1030)	26.1733(6.9935)	56.6390(20.6265)	95.1055(33.7132)
(100, 200)	Avg(Std).N	27.2(5.3914)	52.8(24.2661)	63.7(24.7837)	119.7(65.0317)
	Avg(Std).T	16.6237(3.3583)	38.6381(20.6037)	43.9363(17.9797)	96.7868(48.2903)

TABLE 5. Numerical comparisons between the algorithm of Zhang *et al.* [39] and our algorithm on Problem 5.2.

	Algorithm of Zhang *et al.* [39]		Our new algorithm	
(p,m,n)	Avg.N	Avg.T	Avg.N	Avg.T
(2, 10, 1000)	15.5	2.6293	38.5	11.1218
(2, 10, 2000)	28.5	14.0012	43.5	25.0182
(3, 10, 1000)	101.8	19.3235	177.2	35.6843
(3, 10, 2000)	185.4	90.3898	293.1	131.6528
(4, 10, 1000)	757.6	156.5649	939.3	219.1879
(4, 10, 2000)	1352.1	995.4707	928.9	457.9584

According to the numerical results in Table 5, and from Figure 3, when $p \geq 4$ and $n \geq 2000$, compared with the algorithm of Zhang *et al.* [39], it can be known that our algorithm spend less average time and average number of iterations for solving Problem 5.2 with large-size variables, so our algorithm outperforms that of Zhang *et al.* [39].

According to the numerical results in Table 6, for Problem 5.3, it can be obtained that, when $p = 2$, $m \geq 20$ and $n \geq 20$, the software BARON failed to solve some of them in 3600 s, but our algorithm can solve such a problem with large-size variables, so that our algorithm has strong robustness.

According to the numerical results in Tables 1–7, we can obtain that the software BARON is most effective for small-size problems, but the software BARON is difficult to solve large-size Problems 5.1–5.3. In addition, our algorithm outperforms the existing branch-and-bound algorithms [5, 8, 22, 32, 34, 39] and the software BARON [18] for globally solving large-size Problems 5.1–5.3.

TABLE 6. Numerical comparisons between the software BARON and our algorithm on Problem 5.3.

	Our algorithm		BARON	
(p, m, n)	Avg(Std).N	Avg(Std).T	Avg(Std).N	Avg(Std).T
$(2, 10, 20)$	38.1(16.1483)	2.3581(1.2124)	3.6(3.8930)	0.0760(0.0455)
$(2, 20, 20)$	16.1(12.9825)	1.3093(1.0285)	–	–
$(2, 22, 20)$	14.0(13.0979)	0.6381(0.6014)	–	–
$(2, 20, 30)$	21.2(17.5233)	1.7240(1.5391)	–	–
$(2, 35, 50)$	27.2(23.8225)	2.6217(2.6083)	–	–
$(2, 45, 60)$	21.4(19.0741)	2.3570(1.9797)	–	–
$(2, 45, 100)$	34.4(24.5909)	4.1453(3.1211)	–	–
$(2, 60, 100)$	22.6(17.3282)	4.0992(3.8821)	–	–
$(2, 70, 100)$	27.1(22.0477)	5.1873(4.0660)	–	–
$(2, 70, 120)$	16.7(19.1256)	3.4148(3.8112)	–	–
$(2, 100, 100)$	15.8(13.2816)	5.3048(4.5256)	–	–

TABLE 7. Numerical computational results of our algorithm for Problem 5.3.

(m, n)		$p = 4$	$p = 5$	$p = 6$	$p = 7$
$(10, 20)$	Avg(Std).N	247.2(263.87)	354.4(356.52)	2729.2(2592.85)	3717.3(3263.63)
	Avg(Std).T	9.86(10.64)	12.97(11.60)	95.65(89.26)	137.62(119.17)
$(20, 40)$	Avg(Std).N	214.2(173.77)	1014.6(818.77)	2023.1(1842.23)	5409.6(5291.86)
	Avg(Std).T	10.85(8.75)	48.42(37.25)	93.76(85.29)	270.44(268.75)
$(30, 60)$	Avg(Std).N	401.5(452.72)	1181.5(894.69)	3625.8(3930.85)	9624.9(12243.0)
	Avg(Std).T	26.12(30.83)	73.15(56.78)	229.65(238.37)	662.27(853.38)
$(40, 80)$	Avg(Std).N	370.7(354.64)	989.4(1074.01)	4971.7(6468.19)	13029.5(14689.56)
	Avg(Std).T	32.53(30.95)	82.32(82.72)	549.76(729.55)	1397.89(1880.44)
$(50, 100)$	Avg(Std).N	92.1(175.08)	1499.9(1597.43)	4056.3(3662.31)	12680.7(20150.97)
	Avg(Std).T	12.17(23.46)	221.58(233.72)	641.87(587.81)	2179.26(3445.89)
$(60, 120)$	Avg(Std).N	599.2(461.27)	1343.6(1540.05)	4937.8(5198.42)	14711.4(20419.31)
	Avg(Std).T	115.69(87.77)	267.6(297.65)	1030.96(1142.62)	3222.76(4555.49)
$(70, 140)$	Avg(Std).N	541.8(530.41)	801.0(539.52)	3218.9(5362.54)	9445.5(5927.61)
	Avg(Std).T	147.4(140.14)	191.96(126.11)	884.13(1573.26)	2612.33(1648.91)
$(80, 160)$	Avg(Std).N	176.7(127.17)	1670.9(1999.67)	4821.6(5773.35)	4935.3(6383.97)
	Avg(Std).T	56.04(42.38)	600.18(685.37)	1887.85(2183.56)	1871.4(2504.29)
$(90, 180)$	Avg(Std).N	221.5(201.11)	1792.6(2208.91)	7978.9(14937.33)	12709.0(14685.25)
	Avg(Std).T	96.01(87.51)	865.49(1062.07)	4084.24(7775.46)	6650.20(7774.70)
$(100, 200)$	Avg(Std).N	509.2(438.99)	1654.3(2247.93)	4344.6(5054.17)	5312.3(5370.92)
	Avg(Std).T	302.82(264.99)	969.45(1329.17)	2733.13(3079.83)	3607.57(3481.61)

6. CONCLUSION

This paper studies a class of multiplicative problems (MP) and presents an image space branch-reduction-bound algorithm. In this algorithm, we proposed a novel linear relaxation technique for constructing the parametric linear relaxation programming problem of the equivalent problem. In contrast to the existing branch-and-bound algorithms, the branching search takes place in the image space $\mathbb{R}^p$ of the affine function $\sum_{i=1}^{n} e_{ji}y_i + f_j, j = 1, \ldots, p$, which mitigates the required computational efforts and the computational complexity of the algorithm. Our algorithm can find an ϵ-approximate optimal solution in at most $p \cdot \left[\frac{1}{2}\log_2 \frac{p\mu\sigma\delta^2(\Lambda^0)}{\epsilon}\right]$

iterations. Numerical comparisons show the superiority and efficiency of our algorithm. The future work is to extend our algorithm to solve generalized convex (concave) multiplicative programming problems.

Appendix A.

A.1. Reference [34]

$$\begin{cases} \min & (0.813396y_1 + 0.67440y_2 + 0.305038y_3 + 0.129742y_4 + 0.217796) \\ & \times(0.224508y_1 + 0.063458y_2 + 0.932230y_3 + 0.528736y_4 + 0.091947) \\ \text{s.t.} & 0.488509y_1 + 0.063458y_2 + 0.945686y_3 + 0.210704y_4 \le 3.562809, \\ & -0.324014y_1 - 0.501754y_2 - 0.719204y_3 + 0.099562y_4 \le -0.052215, \\ & 0.445225y_1 - 0.346896y_2 + 0.637939y_3 - 0.257623y_4 \le 0.427920, \\ & -0.202821y_1 + 0.647361y_2 + 0.920135y_3 - 0.983091y_4 \le 0.840950, \\ & -0.886420y_1 - 0.802444y_2 - 0.305441y_3 - 0.180123y_4 \le -1.353686, \\ & -0.515399y_1 - 0.424820y_2 + 0.897498y_3 + 0.187268y_4 \le 2.137251, \\ & -0.591515y_1 + 0.060581y_2 - 0.427365y_3 + 0.579388y_4 \le -0.290987, \\ & 0.423524y_1 + 0.940496y_2 - 0.437944y_3 - 0.742941y_4 \le 0.373620, \\ & y_1 \ge 0, y_2 \ge 0, y_3 \ge 0, y_4 \ge 0. \end{cases}$$

A.2. Reference [32]

$$\begin{cases} \min & (-y_1 + 2y_2 + 2)(4y_1 - 3y_2 + 4)(3y_1 - 4y_2 + 5)^{-1}(-2y_1 + y_2 + 3)^{-1} \\ \text{s.t.} & y_1 + y_2 \le 1.5,\ 0 \le y_1 \le 1,\ 0 \le y_2 \le 1. \end{cases}$$

A.3. Reference [5]

$$\begin{cases} \min & (y_1 + y_2)(y_1 - y_2 + 7) \\ \text{s.t.} & 2y_1 + y_2 \le 14,\quad y_1 + y_2 \le 10, \\ & -4y_1 + y_2 \le 0,\quad 2y_1 + y_2 \ge 6,\ y_1 + 2y_2 \ge 6, \\ & y_1 - y_2 \le 3,\ y_1 + y_2 \ge 0, \\ & y_1 - y_2 + 7 \ge 0,\ y_1, y_2 \ge 0. \end{cases}$$

A.4. Reference [22]

$$\begin{cases} \min & (y_1 + y_2 + 1)^{2.5}(2y_1 + y_2 + 1)^{1.1}(y_1 + 2y_2 + 1)^{1.9} \\ \text{s.t.} & y_1 + 2y_2 \le 6,\quad 2y_1 + 2y_2 \le 8, \\ & 1 \le y_1 \le 3,\ 1 \le y_2 \le 3. \end{cases}$$

A.5. Reference [22]

$$\begin{cases} \min & (3y_1 - 4y_2 + 5)(y_1 + 2y_2 - 1)^{0.5}(2y_1 - y_2 + 4)(y_1 - 2y_2 + 8)^{0.5}(2y_1 + y_2 - 1) \\ \text{s.t.} & 5y_1 - 8y_2 \ge -24,\quad 5y_1 + 8y_2 \le 44, \\ & 6y_1 - 3y_2 \le 15,\quad 4y_1 + 5y_2 \ge 10, \\ & 1 \le y_1 \le 3,\quad 0 \le y_2 \le 1. \end{cases}$$

A.6. Reference [22]

$$\begin{cases} \min & (3y_1 - 2y_2 - 2)^{\frac{2}{3}}(y_1 + 2y_2 + 2)^{\frac{2}{5}} \\ \text{s.t.} & 2y_1 - y_2 \geq 2, \quad y_1 - 2y_2 \leq 2, \\ & y_1 + y_2 \leq 5, \ 3 \leq y_1 \leq 5, \ 1 \leq y_2 \leq 3. \end{cases}$$

A.7. Reference [8]

$$\begin{cases} \min & (y_1 + \frac{1}{9}y_3)(y_2 + \frac{1}{9}y_3) \\ \text{s.t.} & 9y_1 + 9y_2 + 2y_3 \leq 81, \\ & 8y_1 + y_2 + 8y_3 \leq 72, \\ & y_1 + 8y_2 + 8y_3 \leq 72, \\ & 7y_1 + y_2 + y_3 \geq 9, \\ & y_1 + 7y_2 + y_3 \geq 9, \\ & y_1 + y_2 + 7y_3 \geq 9, \\ & 0 \leq y_1 \leq 8, \ 0 \leq y_2 \leq 8, \ 0 \leq y_3 \leq 9. \end{cases}$$

A.8. Reference [39]

$$\begin{cases} \min & (-4y_1 - 2y_4 + 3y_5 + 21)(4y_1 + 2y_2 + 3y_3 - 4y_4 + 4y_5 - 3) \\ & \times(3y_1 + 4y_2 + 2y_3 - 2y_4 + 2y_5 - 7)(-2y_1 + y_2 - 2y_3 + 2y_5 + 11) \\ \text{s.t.} & 4y_1 + 4y_2 + 5y_3 + 3y_4 + y_5 \leq 25, \\ & -y_1 - 5y_2 + 2y_3 + 3y_4 + y_5 \leq 2, \\ & y_1 + 2y_2 + y_3 - 2y_4 + 2y_5 \geq 6, \\ & 4y_2 + 3y_3 - 8y_4 + 11y_5 \geq 8, \\ & y_1 + y_2 + y_3 + y_4 + y_5 \leq 6, \\ & y_1 \geq 1, y_2 \geq 1, y_3 \geq 1, y_4 \geq 1, y_5 \geq 1. \end{cases}$$

Acknowledgements. The authors are very grateful to the responsible editors and the anonymous referees for their valuable comments and suggestions, which have greatly improved the earlier version of this paper. This work is supported by the National Natural Science Foundation of China (11871196; 12071133; 12071112), the China Postdoctoral Science Foundation (2017M622340), the Key Scientific and Technological Research Projects in Henan Province (202102210147; 192102210114), the Science and Technology Climbing Program of Henan Institute of Science and Technology (2018JY01).

References

[1] K.P. Bennett, Global tree optimization: a non-greedy decision tree algorithm. *Comput. Sci. Stat.* **26** (1994) 156–160.
[2] H.P. Benson and G.M. Boger, Outcome-space cutting-plane algorithm for linear multiplicative programming. *J. Optim. Theory App.* **104** (2000) 301–322.
[3] A. Cambini and L. Martein, Generalized Convexity and Optimization: Theory and Applications. *Lecture Notes in Economics and Mathematical Systems*. Springer (2009).
[4] R. Cambini and C. Sodini, On the minimization of a class of generalized linear functions on a flow polytope. *Optimization* **63** (2014) 1449–1464.
[5] Y. Chen and H. Jiao, A nonisolated optimal solution of general linear multiplicative programming problems. *Comput. Oper. Res.* **36** (2009) 2573–2579.
[6] M.C. Dorneich and N.V. Sahinidis, Global optimization algorithms for chip design and compaction. *Eng. Optim.* **25** (1995) 131–154.

[7] Y.L. Gao, C.X. Xu and Y.T. Yang, Outcome-space branch and bound algorithm for solving linear multiplicative programming. In: International Conference on Computational and Information Science. Springer, Berlin, Heidelberg (2005) 675–681.
[8] Y. Gao, C. Xu and Y. Yang, An outcome-space finite algorithm for solving linear multiplicative programming. *Appl. Math. Comput.* **179** (2006) 494–505.
[9] H. Jiao, A branch and bound algorithm for globally solving a class of nonconvex programming problems. *Nonlinear Anal. Theory Methods App.* **70** (2009) 1113–1123.
[10] H. Jiao and Y. Shang, Image space branch-reduction-bound algorithm for globally solving the sum of affine ratios problem. *J. Comput. Math.* (2022) in press.
[11] H.-W. Jiao and Y.-L. Shang, Two-level linear relaxation method for generalized linear fractional programming. *J. Oper. Res. Soc. China* (2022). DOI: 10.1007/s40305-021-00375-4.
[12] H. Jiao, S. Liu and Y. Chen, Global optimization algorithm of a generalized linear multiplicative programming. *J. Appl. Math. Comput.* **40** (2012) 551–568.
[13] H. Jiao, W. Wang, R. Chen, Y. Shang and J. Yin, An efficient outer space algorithm for generalized linear multiplicative programming problem. *IEEE Access* **8** (2020) 80629–80637.
[14] H. Jiao, Y. Shang and R. Chen, A potential practical algorithm for minimizing the sum of affine fractional functions. *Optimization* (2022). DOI: 10.1080/02331934.2022.2032051.
[15] H. Jiao, J. Ma and Y. Shang, Image space branch-and-bound algorithm for globally solving minimax linear fractional programming problem. *Pac. J. Optim.* **18** (2022) 195–212.
[16] H. Jiao, Y. Shang and W. Wang, Solving generalized polynomial problem by using new affine relaxed technique. *Int. J. Comput. Math.* **99** (2022) 309–331.
[17] H.W. Jiao, W.J. Wang and P.P. Shen, Piecewise linear relaxation method for globally solving a class of multiplicative problems. *Pac. J. Optim.* (2022) in press.
[18] A. Khajavirad and N.V. Sahinidis, A hybrid LP/NLP paradigm for global optimization relaxations. *Math. Prog. Comput.* **10** (2018) 383–421.
[19] H. Konno, H. Shirakawa and H. Yamazaki, A mean-absolute deviation-skewness portfolio optimization model. *Ann. Oper. Res.* **45** (1993) 205–220.
[20] T. Kuno, A finite branch-and-bound algorithm for linear multiplicative programming. *Comput. Optim. App.* **20** (2001) 119–135.
[21] T. Kuno, Y. Yajima and H. Konno, An outer approximation method for minimizing the product of several convex functions on a convex set. *J. Global Optim.* **3** (1993) 325–335.
[22] S. Liu and Y. Zhao, An efficient algorithm for globally solving generalized linear multiplicative programming. *J. Comput. Appl. Math.* **296** (2016) 840–847.
[23] X. Liu, T. Umegaki and Y. Yamamoto, Heuristic methods for linear multiplicative programming. *J. Global Optim.* **15** (1999) 433–447.
[24] C.D. Maranas, I.P. Androulakis, C.A. Floudas, A.J. Berger and J.M. Mulvey, Solving long-term financial planning problems via global optimization. *J. Econ. Dyn. Control* **21** (1997) 1405–1425.
[25] T. Matsui, NP-hardness of linear multiplicative programming and related problem. *J. Global Optim.* **9** (1996) 113–119.
[26] J.M. Mulvey, R.J. Vanderbei and S.A. Zenios, Robust optimization of large-scale systems. *Oper. Res.* **43** (1995) 264–281.
[27] H.S. Ryoo and N.V. Sahinidis, Global optimization of multiplicative programs. *J. Global Optim.* **26** (2003) 387–418.
[28] P. Shen and B. Huang, Global algorithm for solving linear multiplicative programming problems. *Optim. Lett.* **14** (2020) 693–710.
[29] P. Shen, X. Bai and W. Li, A new accelerating method for globally solving a class of nonconvex programming problems. *Nonlinear Anal. Theory Methods App.* **71** (2009) 2866–2876.
[30] P. Shen, K. Wang and T. Lu, Outer space branch and bound algorithm for solving linear multiplicative programming problems. *J. Optim.* **78** (2020) 453–482.
[31] P. Shen, K. Wang and T. Lu, Global optimization algorithm for solving linear multiplicative programming problems. *Optimization* (2020). DOI: 10.1080/02331934.2020.1812603.
[32] N.V. Thoai, A global optimization approach for solving the convex multiplicative programming problems. *J. Global Optim.* **1** (1991) 341–357.
[33] C.F. Wang and S.Y. Liu, A new linearization method for generalized linear multiplicative programming. *Comput. Oper. Res.* **38** (2011) 1008–1013.
[34] C.F. Wang, S.Y. Liu and P.P. Shen, Global minimization of a generalized linear multiplicative programming. *Appl. Math. Modell.* **36** (2012) 2446–2451.
[35] C.F. Wang, Y.Q. Bai and P.P. Shen, A practicable branch-and-bound algorithm for globally solving multiplicative programming. *Optimization* **66** (2017) 397–405.
[36] L.P. Yang, P.P. Shen and Y.G. Pei, A global optimization approach for solving generalized nonlinear multiplicative programming problem. In: Abstract Applied Analysis. Hindawi (2014). DOI: 10.1155/2014/641909.
[37] Y.F. Zhao and S.Y. Liu, An efficient method for generalized linear multiplicative programming problem with multiplicative constraints. *SpringerPlus* **5** (2016) 1–14.
[38] Y.F. Zhao and T. Zhao, Global optimization for generalized linear multiplicative programming using convex relaxation. *Math. Prob. Eng.* (2018). DOI: 10.1155/2018/9146309.
[39] B. Zhang, Y. Gao, X. Liu and X. Huang, Output-space branch-and-bound reduction algorithm for a class of linear multiplicative programs. *Mathematics* **8** (2020) 315.

[40] B. Zhang, Y. Gao and X. Liu, An efficient polynomial time algorithm for a class of generalized linear multiplicative programs with positive exponents. *Math. Prob. Eng.* **2021** (2021). DOI: 10.1155/2021/8877037.