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

HONGWEI JIAO¹, WENJIE WANG¹, JINGBEN YIN¹, YOULIN SHANG²

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.

Keywords: Multiplicative problem; Global optimization; Branch-and-bound algorithm; Image space region reduction technique; Computational complexity

Mathematics Subject Classification. 90C26

---

* Corresponding E-mail: jiaohongwei@126.com (H. Jiao)
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, China

© EDP Sciences 2001
1. Introduction

This paper considers the following a class of multiplicative problems:

\[
\begin{align*}
\text{(MP)} & : \min \quad G(y) = \prod_{j=1}^{p} \left( \sum_{i=1}^{n} e_{ji} y_i + f_j \right)^{\alpha_j}, \\
& \text{s.t.} \quad y \in D = \{ y \in \mathbb{R}^n | Ay \leq b \},
\end{align*}
\]

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 [1,2], VLISI chip design [3], robust optimization [4], decision tree optimization [5], financial optimization [6,7], and so on [8-10]. 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.

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 [11-20], level set algorithm [21], outer-approximation [22], cutting-plane method [23], finite algorithms [24,25], etc. In recent years, Shen et al. [26] 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 [27] proposed a decomposition branch-and-bound algorithm for linear multiplicative problem; Jiao et al. [28] 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. [29] 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. [30] proposed a branch-and-bound algorithm for globally solving the linear multiplicative problem. Zhang et al. [31] 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 [32] 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. [33] 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 $\epsilon$.

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

Let $\Lambda^0 = \{ z \in \mathbb{R}^p | l^0_j \leq z_j \leq u^0_j, j = 1, 2, \ldots, p \}$, 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:

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

It is obvious that, if $(y^*, z^*_1, z^*_2, \ldots, z^*_p)$ 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 $(y^*, z^*_1, z^*_2, \ldots, z^*_p)$ 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 $\Lambda = \{ z \in \mathbb{R}^p | l_j \leq z_j \leq u_j, j = 1, 2, \ldots, p \} \subseteq \Lambda^0$, 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, $\exists \xi_j \in [l_j, u_j], \text{s.t. } \ln z_j = \ln l_j + \frac{1}{l_j} (z_j - l_j)$, so that we have

\[
\ln z_j \geq \ln l_j + \frac{1}{u_j} (z_j - l_j) \text{ and } \ln z_j \leq \ln l_j + \frac{1}{l_j} (z_j - l_j).
\]

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

\[
\ln z_j \geq \ln u_j + \frac{1}{l_j} (z_j - u_j) \text{ and } \ln z_j \leq \ln u_j + \frac{1}{u_j} (z_j - u_j).
\]

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

\[
\ln z_j \geq \max\{\ln l_j + \frac{1}{u_j} (z_j - l_j), \ln u_j + \frac{1}{l_j} (z_j - u_j)\} = \phi^l_j(z_j),
\]

\[
\ln z_j \leq \min\{\ln l_j + \frac{1}{l_j} (z_j - l_j), \ln u_j + \frac{1}{u_j} (z_j - u_j)\} = \phi^u_j(z_j).
\]
Therefore, for any \( z \in \Lambda \subseteq \Lambda^0 \), we have
\[
\varphi_j^l(z_j) \leq \varphi_j(z_j) \leq \varphi_j^n(z_j), j = 1, 2, \ldots, p.
\]

**Theorem 1.** For any \( z \in \Lambda = \{ z \in \mathbb{R}^p | z_j \leq u_j, j = 1, 2, \ldots, p \} \), let
\[
\Delta_1^l(z_j) = \varphi_j(z_j) - \varphi_j^l(z_j), \quad \Delta_2^l(z_j) = \varphi_j^n(z_j) - \varphi_j(z_j),
\]
consider the functions \( \varphi_j^l(z_j), \varphi_j(z_j), \) and \( \varphi_j^n(z_j), \). Then we have
\[
\lim_{\|u-l\| \to 0} \max_{z \in \Lambda} \Delta_1^l(z_j) = \lim_{\|u-l\| \to 0} \max_{z \in \Lambda} \Delta_2^l(z_j) = 0.
\]

**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
\[
\begin{align*}
\Delta_1^l(z_j) &= \varphi_j(z_j) - \varphi_j^l(z_j) \\
&= \ln z_j - \max \{ \ln l_j + \frac{1}{\eta_j}(z_j - l_j), \ln u_j + \frac{1}{\xi_j}(z_j - u_j) \} \\
&= \min \{ \ln z_j - \ln l_j - \frac{1}{\eta_j}(z_j - l_j), \ln z_j - \ln u_j + \frac{1}{\xi_j}(z_j - u_j) \} \\
&= \min \{ \frac{1}{\xi_j}(z_j - l_j) - \frac{1}{\eta_j}(z_j - l_j), \frac{1}{\eta_j}(z_j - u_j) - \frac{1}{\xi_j}(z_j - u_j) \} \\
&\leq \frac{1}{\eta_j u_j}(u_j - l_j) + \frac{1}{\xi_j}(u_j - l_j), \frac{1}{\xi_j}(u_j - l_j) + \frac{1}{\eta_j}(u_j - l_j)
\end{align*}
\]
since \( \Delta_1^l(z_j) \to 0 \) as \( u_j - l_j \to 0 \), we can obtain that
\[
\lim_{\|u-l\| \to 0} \max_{z \in \Lambda} \Delta_1^l(z_j) = 0.
\]

Similarly, we can get that
\[
\begin{align*}
\Delta_2^l(z_j) &= \varphi_j^n(z_j) - \varphi_j(z_j) \\
&= \min \{ \ln l_j + \frac{1}{\xi_j}(z_j - l_j), \ln u_j + \frac{1}{\eta_j}(z_j - u_j) \} - \ln z_j \\
&= \min \{ \ln l_j - \ln z_j + \frac{1}{\eta_j}(z_j - l_j), \ln u_j - \ln z_j + \frac{1}{\xi_j}(z_j - u_j) \} \\
&= \min \{ \frac{1}{\xi_j}(z_j - l_j) - \frac{1}{\eta_j}(z_j - l_j), \frac{1}{\xi_j}(z_j - u_j) + \frac{1}{\eta_j}(z_j - u_j) \} \\
&\leq \frac{1}{\eta_j u_j}(u_j - l_j) + \frac{1}{\xi_j}(u_j - l_j), \frac{1}{\xi_j}(u_j - l_j) + \frac{1}{\eta_j}(u_j - l_j)
\end{align*}
\]
since \( \Delta_2^l(z_j) \to 0 \) as \( u_j - l_j \to 0 \), we can get that
\[
\lim_{\|u-l\| \to 0} \max_{z \in \Lambda} \Delta_2^l(z_j) = 0.
\]

Taken together above, we can obtain
\[
\lim_{\|u-l\| \to 0} \max_{z \in \Lambda} \Delta_1^l(z_j) = \lim_{\|u-l\| \to 0} \max_{z \in \Lambda} \Delta_2^l(z_j) = 0,
\]
and the proof is complete. \( \diamond \)
Next, let \( \psi(z) \) denote the underestimating function of the function \( \phi(z) \), by Theorem 1, we have

\[
\phi(z) = \sum_{j=1}^{p} \alpha_j \ln z_j \geq \sum_{j=1, \alpha_j > 0}^{p} \alpha_j \phi_j^l(z_j) + \sum_{j=1, \alpha_j < 0}^{p} \alpha_j \phi_j^u(z_j) = \psi(z).
\]

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:

\[
(\text{PLRP}): \begin{cases}
\min \psi(z) = \sum_{j=1, \alpha_j > 0}^{p} \alpha_j s_j + \sum_{j=1, \alpha_j < 0}^{p} \alpha_j s_j, \\
\text{s.t.} \quad \ln l_j + \frac{1}{\alpha_j}(z_j - l_j) \leq s_j, \alpha_j > 0, j = 1, 2, \ldots, p, \\
\ln l_j + \frac{1}{\alpha_j}(z_j - u_j) \leq s_j, \alpha_j > 0, j = 1, 2, \ldots, p, \\
\ln l_j + \frac{1}{\alpha_j}(z_j - l_j) \geq s_j, \alpha_j < 0, j = 1, 2, \ldots, p, \\
\ln u_j + \frac{1}{\alpha_j}(z_j - u_j) \geq s_j, \alpha_j < 0, j = 1, 2, \ldots, p, \\
z_j = \sum_{i=1}^{n} e_j y_i + f_j, j = 1, 2, \ldots, p, \\
y \in D, z \in \Lambda.
\end{cases}
\]

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 \( \overline{UB} \) is a best currently known upper bound of the global optimal value of the problem (EP), for any \( z \in \Lambda = \{ z \in \mathbb{R}^p : l_j \leq z_j \leq u_j, j = 1, 2, \ldots, p \} \subseteq \Lambda^0 \), we let

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

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

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

\[
\hat{\Lambda}^2 = \left\{ \begin{array}{ll}
l_j \leq z_j \leq u_j, & j = 1, 2, \ldots, p, j \neq \tau, \\
\exp \left( UB - \hat{\theta} + \alpha_\tau \ln l_\tau \right) < z_\tau \leq u_\tau, & j = \tau,
\end{array} \right.
\]

and

\[
\hat{\Lambda}^2 = \left\{ \begin{array}{ll}
l_j \leq z_j \leq u_j, & j = 1, 2, \ldots, p, j \neq \tau, \\
l_\tau \leq z_\tau < \exp \left( UB - \hat{\theta} + \alpha_\tau \ln l_\tau \right), & j = \tau.
\end{array} \right.
\]

**Proof.** If \( \hat{\theta} > UB \), then we have that

\[
\min_{z \in \hat{\Lambda}} \phi(z) = \min_{z \in \hat{\Lambda}} \left( \sum_{j=1}^{p} \alpha_j \ln z_j \right) = \sum_{j=1, \alpha_j > 0}^{p} \alpha_j \ln l_j + \sum_{j=1, \alpha_j < 0}^{p} \alpha_j \ln u_j = \hat{\theta} > UB.
\]

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

If \( \hat{\theta} \leq UB \), for any \( z \in \hat{\Lambda}^1 \), consider the \( \tau \)th component \( z_\tau \) of \( z \), we have \( \exp \left( UB - \hat{\theta} + \alpha_\tau \ln l_\tau \right) < z_\tau \leq u_\tau \), thus, we can obtain

\[
z_\tau \geq \exp \left( UB - \hat{\theta} + \alpha_\tau \ln l_\tau \right),
\]

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

\[
\alpha_\tau \ln z_\tau > UB - \hat{\theta} + \alpha_\tau \ln l_\tau.
\]

Also because

\[
\hat{\theta} = \sum_{j=1, \alpha_j > 0}^{p} \alpha_j \ln l_j + \sum_{j=1, \alpha_j < 0}^{p} \alpha_j \ln u_j,
\]

we can follow that

\[
\min_{z \in \hat{\Lambda}^1} \phi(z) = \min_{z \in \hat{\Lambda}^1} \left( \sum_{j=1}^{p} \alpha_j \ln z_j \right) > \sum_{j=1, j \neq \tau, \alpha_j > 0}^{p} \alpha_j \ln l_j + \sum_{j=1, j \neq \tau, \alpha_j < 0}^{p} \alpha_j \ln u_j + \alpha_\tau \ln \left( \exp \left( UB - \hat{\theta} + \alpha_\tau \ln l_\tau \right) \right)
\]

\[= \hat{\theta} - \alpha_\tau \ln l_\tau + UB - \hat{\theta} + \alpha_\tau \ln l_\tau
\]

\[= UB.
\]

Thus, there exists no globally optimal solution of the problem (EP) over \( \hat{\Lambda}^1 \).
For any \( z \in \hat{\Lambda}^2 \), consider the \( \tau \)th component \( z_\tau \) of \( z \), we have \( l_\tau \leq z_\tau < \exp \left( \frac{UB - \hat{\theta} + \alpha_\tau \ln u_\tau}{\alpha_\tau} \right) \), Thus, we can obtain \( z_\tau < \exp \left( \frac{UB - \hat{\theta} + \alpha_\tau \ln u_\tau}{\alpha_\tau} \right) \), since \( \alpha_\tau < 0 \), and taking the logarithm of both sides of the above formula, we can follow that \( \alpha_\tau \ln z_\tau > UB - \hat{\theta} + \alpha_\tau \ln u_\tau \).

Also because \( \hat{\theta} = \sum_{j=1, \alpha_j > 0}^{p} \alpha_j \ln l_j + \sum_{j=1, \alpha_j < 0}^{p} \alpha_j \ln u_j \),

Thus, we can follow that

\[
\min_{z \in \hat{\Lambda}^2} \phi(z) = \min_{z \in \hat{\Lambda}^2} \left( \sum_{j=1}^{p} \alpha_j \ln z_j \right) \geq \sum_{j=1, \alpha_j > 0}^{p} \alpha_j \ln l_j + \sum_{j=1, \alpha_j < 0}^{p} \alpha_j \ln u_j + \alpha_\tau \ln \left( \exp \left( \frac{UB - \hat{\theta} + \alpha_\tau \ln u_\tau}{\alpha_\tau} \right) \right) = \hat{\theta} - \alpha_\tau \ln u_\tau + UB - \hat{\theta} + \alpha_\tau \ln u_\tau = UB.
\]

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

According to Theorem 2, we can construct some image space region reduction techniques, which can provide the possibility for deleting the whole investigated rectangle \( \Lambda \) 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 \( \Lambda = \{ z \in \mathbb{R}^p | l_j \leq z_j \leq u_j, j = 1, 2, \ldots, p \} \subseteq \Lambda^0 \), the proposed branching rule is given as follows:

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

\[ \Lambda^1 = \{ z \in \mathbb{R}^p | l_j \leq z_j \leq u_j, j \neq q, j = 1, 2, \ldots, p; \ l_j \leq z_j \leq \frac{l_j + u_j}{2}, j = q \} \]

and

\[ \Lambda^2 = \{ z \in \mathbb{R}^p | l_j \leq z_j \leq u_j, j \neq q, j = 1, 2, \ldots, p; \ \frac{l_j + u_j}{2} \leq z_j \leq u_j, j = q \} . \]

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

### 4.2. Algorithm statement

Let \( LB(\Lambda) \) and \( (y(\Lambda), z(\Lambda)) \) be the optimal value and the optimal solution of the (PLRP) over the sub-rectangle \( \Lambda \), 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 \( UB = +\infty \), 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 \( LB_0 = LB(\Lambda^0) \), \( (y^0, z^0) = (y(\Lambda^0), z(\Lambda^0)) \), \( UB = \phi(z^0) \), and \( F = F \cup \{(y^0, z^0)\} \).

If \( UB - LB_0 \leq \epsilon \), then the algorithm stops: \( (y^0, z^0) \) is a global \( \epsilon \)-optimal solution of the problem (EP). Otherwise, proceed with Step 2.

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

**Step 3.** If \( \Lambda^k \neq \emptyset \), for each \( \Lambda \in \Lambda^k \), solve the following improved parametric linear relaxation programming problem:

\[
\begin{aligned}
\text{(IPLRP):} \quad & \min \psi(z) = \sum_{j=1, \alpha_j > 0}^p \alpha_j s_j + \sum_{j=1, \alpha_j < 0}^p \alpha_j s_j, \\
\text{s.t.} \quad & \ln l_j + \frac{1}{u_j} (z_j - l_j) \leq s_j, \alpha_j > 0, j = 1, 2, \ldots, p, \\
& \ln u_j + \frac{1}{l_j} (z_j - u_j) \leq s_j, \alpha_j > 0, j = 1, 2, \ldots, p, \\
& \ln l_j + \frac{1}{l_j} (z_j - l_j) \geq s_j, \alpha_j < 0, j = 1, 2, \ldots, p, \\
& \ln u_j + \frac{1}{u_j} (z_j - u_j) \geq s_j, \alpha_j < 0, j = 1, 2, \ldots, p, \\
& z_j = \sum_{i=1}^n e_{ji} y_i + f_j, j = 1, 2, \ldots, p, \\
& \sum_{j=1, \alpha_j > 0}^p \alpha_j s_j + \sum_{j=1, \alpha_j < 0}^p \alpha_j s_j \leq UB, \\
& y \in \mathcal{D}, z \in \Lambda,
\end{aligned}
\]

we can obtain \( LB(\Lambda) \) and \( (y(\Lambda), z(\Lambda)) \).
If $LB(\Lambda) > UB$, set $\overline{X}^k = \overline{X} \backslash \Lambda$. 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 $\Omega_k = (\Omega_k \backslash \Lambda) \cup \overline{X}^k$, and denote the new lower bound as $LB_k = \inf_{\Lambda \in \Omega_k} LB(\Lambda)$.

**Step 4.** Let $\Omega_{k+1} = \Omega_k \backslash \{\Lambda | UB - LB(\Lambda) \leq \epsilon, \Lambda \in \Omega_k\}$. If $\Omega_{k+1} = \emptyset$, then the algorithm stops: $UB$ and $(y^k, z^k)$ are the global $\epsilon$-optimal value and the global $\epsilon$-optimal solution of the problem (EP), respectively. Otherwise, select an active node $\Lambda^{k+1}$ such that $\Lambda^{k+1} = \arg\min_{\Lambda \in \Omega_{k+1}} LB(\Lambda)$, 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 3.** 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

$$UB - LB_k \leq \epsilon.$$

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

$$\phi(z^k) - LB_k \leq \epsilon.$$

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

$$LB_k \leq v.$$

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

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

Taken together above, it implies that

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

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

$$v \leq \phi(z^k) \leq v + \epsilon.$$
Hence, when the presented algorithm finitely terminates after $k$ iterations, it follows that $y^k$ is a global $\epsilon$-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 $\Lambda^k$, then it is obvious that $(y^k, z^k)$ is a sequence of feasible solution for the problem (EP) over $\Lambda^0$, where

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

Let $(y^*, z^*)$ be a gathering point of the sequence $\{(y^k, z^k)\}$, and we have

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

So we have

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

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 $\Lambda^0$. Since $\{LB(\Lambda^k)\}$ is a real increasing sequence, we have that

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

From the renewal process of lower bound and Theorem 1, we have that

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

Therefore, we can get that

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

Therefore, $(y^*, z^*)$ is a global optimum solution for the problem (EP) over $\Lambda^0$. 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

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

as

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

and for convenience, we also define

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

**Theorem 4.** For any given termination tolerance $\epsilon > 0$, for any $\Lambda \subseteq \Lambda^0$, if $\delta^2(\Lambda) \leq \epsilon$, then, for any feasible solution $(y, z)$ to the problem (PLRP) over $\Lambda$, we have that

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

**Proof.** For any feasible solution $(y, z)$ to the problem (PLRP) over $\Lambda$, it is obvious that $(y, z)$ is a feasible solution to the problem (EP) over $\Lambda$. From Theorem 1, for any $z \in \Lambda$, if $\delta^2(\Lambda) \leq \epsilon$ for any sufficiently small positive number $\epsilon$, we have that

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

$$= \sum_{j=1, \alpha_j > 0}^{p} \alpha_j \left[\ln z_j - \max\{\ln l_j + \frac{1}{u_j} (z_j - l_j), \ln u_j + \frac{1}{l_j} (z_j - u_j)\}\right]$$

$$+ \sum_{j=1, \alpha_j < 0}^{p} \alpha_j \left[\ln z_j - \min\{\ln l_j + \frac{1}{u_j} (z_j - l_j), \ln u_j + \frac{1}{l_j} (z_j - u_j)\}\right]$$

$$= \sum_{j=1, \alpha_j > 0}^{p} \alpha_j |\varphi_j(z_j) - \varphi_j^l(z_j)| - \sum_{j=1, \alpha_j < 0}^{p} \alpha_j |\varphi_j(z_j) - \varphi_j^u(z_j)|$$

$$\leq \sum_{j=1}^{p} |\alpha_j| \times \frac{1}{(l_j)^2} (u_j - l_j)^2$$

$$\leq \sum_{j=1}^{p} (\alpha_j) \times \frac{1}{(l_j)^2} (u_j - l_j)^2$$

$$\leq p\mu\sigma\delta^2(\Lambda)$$

$$\leq p\mu\sigma\epsilon,$$

and the proof of the theorem is completed.

**Theorem 5.** For any given termination tolerance $\epsilon \in (0, 1)$, the presented algorithm can obtain a global $\epsilon$-optimal solution in at most

$$p \cdot \left[\frac{1}{2 \log_2 \frac{p\mu\sigma\delta^2(\Lambda^0)}{\epsilon}}\right]$$
Proof. Without losing generality, we assume that, at each iteration, the rectangle partition is performed in the selected sub-rectangle $\Lambda$ at step 2 of the algorithm. After $k \cdot p$ iterations, we have that

$$\delta(\Lambda) \leq \frac{1}{2^k} \delta(\Lambda^0),$$

thus, we can obtain that

$$\delta^2(\Lambda) \leq \frac{1}{2^{2k}} \delta^2(\Lambda^0),$$

by the proof of Theorem 4, if

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

e.i.,

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

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

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

iterations, we have that

$$0 \leq \phi(z) - \phi(z^*) \leq \phi(z) - LB(\Lambda) = |\phi(z) - \psi(z)| \leq \epsilon,$$

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, 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 $\epsilon$-optimal solution for the problem (EP). Therefore, $y^k$ is a global $\epsilon$-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 [34] and the known branch-and-bound algorithms [21, 29, 35-38]. All numerical tests are implemented in MATLAB R2014a and run on a microcomputer with Intel(R) Core(TM) i5-7200U CPU @2.50GHz processor and 16 GB RAM. In all numerical tests, the maximum CPU time limit is set to 3600s.
First of all, with the given convergence tolerance $10^{-6}$, we tested some small-size examples, see Appendix A.1-A.8 for details, compared with the existing branch-and-bound algorithms [21, 29, 35-38] 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 1-3 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 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 3600s.

**Problem 1 (Ref.[39]).**

\[
\begin{align*}
\min & \prod_{j=1}^{p} \left( \sum_{i=1}^{n} e_{ji} y_{i} + 1 \right), \\
\text{s.t.} & \quad A y \leq b, y \geq 0,
\end{align*}
\]

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 $b, i = 1, \ldots, n$, are all randomly generated by setting $b_{i} = \sum_{j=1}^{p} a_{ij} + 2 \pi$, where $\pi$ is randomly generated in $[0, 1]$.

**Problem 2 (Ref.[29]).**

\[
\begin{align*}
\min & \prod_{j=1}^{p} e_{j}^{\top} y, \\
\text{s.t.} & \quad \sum_{i=1}^{n} A_{si} y_{i} \leq b_{s}, s = 1, 2, \ldots, m, \\
& \quad 0 \leq y_{i} \leq 1, i = 1, 2, \ldots, n,
\end{align*}
\]

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 $\pi$ is randomly generated in $[0, 1]$.

**Problem 3.**

\[
\begin{align*}
\min & \prod_{j=1}^{p} \left( \sum_{i=1}^{n} e_{ji} y_{i} + f_{j} \right)^{\alpha_{j}}, \\
\text{s.t.} & \quad A y \leq b, y \geq 0,
\end{align*}
\]
where \( e_{ji}, j = 1, 2, \ldots, p, i = 1, 2, \ldots, n, \) and \( f_{ij}, j = 1, 2, \ldots, p, \) are all randomly generated in \([0,1]\), all components of the matrix \( A \) and \( \alpha_{ji}, j = 1, 2, \ldots, p, \) 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 \( \pi \) is randomly generated in \([0,1]\).

**Table 1.** Numerical comparisons among the software BARON, the algorithms of Refs.\[21, 29, 35-38\] and our algorithm on Appendix A.1-A.8.

<table>
<thead>
<tr>
<th>Example</th>
<th>Refs.</th>
<th>Optimal value</th>
<th>Optimal solution</th>
<th>Iter</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Our</td>
<td>0.89019</td>
<td>(1.3148, 0.1396, 0.0, 0.4233)</td>
<td>1</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td>[35]</td>
<td>0.8902</td>
<td>(1.3148, 0.1396, 0.0, 0.4233)</td>
<td>1</td>
<td>0.19</td>
</tr>
<tr>
<td></td>
<td>BARON</td>
<td>0.8902</td>
<td>(1.3148, 0.1396, 0.0, 0.4233)</td>
<td>3</td>
<td>0.05</td>
</tr>
<tr>
<td>2</td>
<td>Our</td>
<td>0.53333</td>
<td>(0.0, 0.0)</td>
<td>23</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>[36]</td>
<td>0.53333</td>
<td>(0.0, 0.0)</td>
<td>16</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>BARON</td>
<td>0.53333</td>
<td>(0.0, 0.0)</td>
<td>1</td>
<td>0.03</td>
</tr>
<tr>
<td>3</td>
<td>Our</td>
<td>10.0</td>
<td>(2.0, 8.0)</td>
<td>3</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td>[37]</td>
<td>10.0</td>
<td>(2.0, 8.0)</td>
<td>41</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>BARON</td>
<td>10.0000</td>
<td>(2, 8)</td>
<td>1</td>
<td>0.05</td>
</tr>
<tr>
<td>4</td>
<td>Our</td>
<td>997.66127</td>
<td>(1, 1)</td>
<td>0</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>[21]</td>
<td>997.66127</td>
<td>(1, 1)</td>
<td>3</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>BARON</td>
<td>997.6613</td>
<td>(1, 1)</td>
<td>1</td>
<td>0.03</td>
</tr>
<tr>
<td>5</td>
<td>Our</td>
<td>263.78893</td>
<td>(1.25, 1)</td>
<td>1</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>[21]</td>
<td>263.7889</td>
<td>(1.25, 1)</td>
<td>11</td>
<td>0.30</td>
</tr>
<tr>
<td></td>
<td>BARON</td>
<td>263.7889</td>
<td>(1.25, 1)</td>
<td>3</td>
<td>0.03</td>
</tr>
<tr>
<td>6</td>
<td>Our</td>
<td>5.00931</td>
<td>(3, 2)</td>
<td>0</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>[21]</td>
<td>5.00931</td>
<td>(3, 2)</td>
<td>2</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>BARON</td>
<td>5.0093</td>
<td>(3, 2)</td>
<td>1</td>
<td>0.03</td>
</tr>
<tr>
<td>7</td>
<td>Our</td>
<td>0.99123</td>
<td>(0.0, 8.0, 1.0)</td>
<td>9</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>[38]</td>
<td>0.9901235</td>
<td>(0.0, 8.0, 1.0)</td>
<td>3</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>BARON</td>
<td>0.9912</td>
<td>(0.0, 8.0, 1.0)</td>
<td>7</td>
<td>0.06</td>
</tr>
<tr>
<td>8</td>
<td>Our</td>
<td>9504.0</td>
<td>(1.0, 2.0, 1.0, 1.0, 1.0)</td>
<td>4</td>
<td>0.14</td>
</tr>
<tr>
<td></td>
<td>[29]</td>
<td>9504.0</td>
<td>(1.0, 2.0, 1.0, 1.0, 1.0)</td>
<td>2</td>
<td>0.07</td>
</tr>
<tr>
<td></td>
<td>BARON</td>
<td>9504.0</td>
<td>(1.0, 2.0, 1.0, 1.0, 1.0)</td>
<td>1</td>
<td>0.03</td>
</tr>
</tbody>
</table>

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 \[21, 29, 35-38\] and the software BARON \[34\]. 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 1-3, we solved ten independently generated instances and recorded their average number of iteration and average CPU time. For Problems 1-3, when \( n > 20 \), the software BARON failed to solve some of ten independently generated instances in 3600s, so that we only numerically compared our algorithm with the algorithm of Jiao et al. \[40\] in Table 3, and we only numerically compared our algorithm with the algorithm of Zhang et al. \[29\] 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 Table 3 and Table 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 [39], our algorithm takes less average time and average number of iteration to solve Problem 1 with large-size variables, and the software BARON is only effective for Problem 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 solve some of them in 3600s. But, our algorithm can solve Problem 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. [40] for solving Problem 1 with the larger $p$, so our algorithm is superior than the algorithm of Jiao et al. [40].

According to the numerical results in Tables 4 and 7, we can see that our algorithm can solve Problems 2 and 3 with the large-size variables, but the software BARON failed to solve some of them in 3600s, so this indicates the effectiveness and robustness of our algorithm.

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. [29], it can be known that our algorithm spend less average time and average number of iterations for solving Problem 2 with large-size variables, so our algorithm outperforms that of Zhang et al. [29].

According to the numerical results in Table 6, for Problem 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 3600s, 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 1-3. In addition, our algorithm outperforms the existing branch-and-bound algorithms [21, 29, 35-38] and the software BARON [34] for globally solving large-size Problems 1-3.

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 $\epsilon$-approximate optimal solution in at most $p \left[ \frac{1}{2} \log_2 \frac{p\mu \delta^2(\Lambda^0)}{\epsilon} \right]$
Table 2. Numerical comparisons among the software BARON, algorithms of Wang and Liu [39], and our algorithm on Problem 1.

<table>
<thead>
<tr>
<th>(p,m,n)</th>
<th>Algorithm of Wang and Liu [39]</th>
<th>Our algorithm</th>
<th>BARON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg(Std).N</td>
<td>Avg(Std).T</td>
<td>Avg(Std).N</td>
</tr>
<tr>
<td>(2,10,20)</td>
<td>14.2(1.5492) 0.6062(0.0695)</td>
<td>27.0(14.0712) 1.0567(0.5579)</td>
<td>8.8(9.21) 0.1(0.0267)</td>
</tr>
<tr>
<td>(2,20,20)</td>
<td>17.4(1.7127) 0.8368(0.0756)</td>
<td>21.3(11.0860) 1.4858(0.9255)</td>
<td>-</td>
</tr>
<tr>
<td>(2,22,20)</td>
<td>18.5(1.9003) 0.9400(0.1235)</td>
<td>16.6(9.1797) 1.3410(0.7558)</td>
<td>-</td>
</tr>
<tr>
<td>(2,20,30)</td>
<td>19.9(0.5676) 1.0781(0.0674)</td>
<td>29.1(9.4569) 1.9719(1.0081)</td>
<td>-</td>
</tr>
<tr>
<td>(2,35,50)</td>
<td>21.2(0.4316) 1.8415(0.1338)</td>
<td>31.1(2.9981) 2.0209(0.3103)</td>
<td>-</td>
</tr>
<tr>
<td>(2,45,60)</td>
<td>23.0(0.6667) 2.4338(0.1016)</td>
<td>30.9(2.9981) 3.2201(0.5860)</td>
<td>-</td>
</tr>
<tr>
<td>(2,45,100)</td>
<td>35.7(1.1595) 5.1287(0.0935)</td>
<td>36.1(2.6437) 3.8516(0.5096)</td>
<td>-</td>
</tr>
<tr>
<td>(2,60,100)</td>
<td>36.1(0.7379) 6.8143(0.1713)</td>
<td>32.1(2.9981) 4.6820(0.6611)</td>
<td>-</td>
</tr>
<tr>
<td>(2,70,100)</td>
<td>36.6(1.2649) 8.1967(0.2121)</td>
<td>30.5(3.8370) 5.3423(0.8261)</td>
<td>-</td>
</tr>
<tr>
<td>(2,70,120)</td>
<td>39.1(1.6633) 9.5642(0.2975)</td>
<td>33.3(3.0569) 6.7911(0.9495)</td>
<td>-</td>
</tr>
<tr>
<td>(2,100,100)</td>
<td>37.5(2.1731) 13.0578(0.3543)</td>
<td>22.1(6.5735) 7.2817(2.1901)</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3. Numerical comparisons between our algorithm and that of Jiao et al. [40] on Problem 1.

<table>
<thead>
<tr>
<th>(p,m,n)</th>
<th>Algorithm of Jiao et al. [40]</th>
<th>Our algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg(Std).N</td>
<td>Avg(Std).T</td>
</tr>
<tr>
<td>(2,10,10)</td>
<td>51.6(27.1260) 0.5366(0.2774)</td>
<td>7.6(9.1797) 0.1440(0.1568)</td>
</tr>
<tr>
<td>(2,10,20)</td>
<td>81.1(29.0304) 0.8296(0.2990)</td>
<td>19.6(12.9974) 0.3642(0.2395)</td>
</tr>
<tr>
<td>(2,10,40)</td>
<td>90.4(24.3867) 0.9427(0.2610)</td>
<td>28.4(9.5475) 0.5349(0.1841)</td>
</tr>
<tr>
<td>(2,10,60)</td>
<td>93.8(27.3447) 1.2067(0.3675)</td>
<td>32.5(5.5927) 0.6421(0.0721)</td>
</tr>
<tr>
<td>(2,10,80)</td>
<td>87.2(23.3419) 0.9561(0.3115)</td>
<td>30.3(10.7078) 0.6612(0.2127)</td>
</tr>
<tr>
<td>(2,10,100)</td>
<td>105.3(43.7113) 1.1878(0.4902)</td>
<td>33.2(1.6193) 0.7802(0.0637)</td>
</tr>
<tr>
<td>(4,10,10)</td>
<td>385.6(304.8264) 4.6592(3.5951)</td>
<td>35.8(2.9916) 0.6465(0.3884)</td>
</tr>
<tr>
<td>(4,10,20)</td>
<td>1544.9(1034.5785) 15.3208(10.4886)</td>
<td>80.4(18.0012) 1.6899(0.456)</td>
</tr>
<tr>
<td>(4,10,40)</td>
<td>3766.5(3714.9863) 41.3699(43.3225)</td>
<td>125.5(49.4711) 2.7247(1.4014)</td>
</tr>
<tr>
<td>(4,10,60)</td>
<td>3500.4(4733.4820) 42.8343(67.4555)</td>
<td>123.1(25.6922) 2.9937(0.7626)</td>
</tr>
<tr>
<td>(4,10,80)</td>
<td>6588.5(4183.8860) 108.5281(76.9548)</td>
<td>144.2(26.1950) 3.5831(0.6546)</td>
</tr>
<tr>
<td>(4,10,100)</td>
<td>4475.8(6161.0653) 82.6249(143.2454)</td>
<td>196.4(87.1183) 5.5485(2.6318)</td>
</tr>
<tr>
<td>(6,10,10)</td>
<td>2785.0(4542.3064) 53.9199(94.6386)</td>
<td>61.2(29.0777) 1.3197(0.7418)</td>
</tr>
<tr>
<td>(6,10,20)</td>
<td>27365.3(3428.9002) 426.2640(663.2382)</td>
<td>133.9(69.5453) 2.9194(1.6542)</td>
</tr>
<tr>
<td>(6,10,40)</td>
<td>66717.5(7380.1577) 1895.0217(3169.6473)</td>
<td>306.1(204.2659) 7.6800(4.7208)</td>
</tr>
<tr>
<td>(8,10,10)</td>
<td>12192.2(1774.0386) 169.3895(306.5079)</td>
<td>112.6(58.7995) 2.1026(1.2695)</td>
</tr>
<tr>
<td>(8,10,20)</td>
<td>79305.9(6809.9694) 1454.1710(1541.1460)</td>
<td>357.4(249.8036) 7.6240(5.5012)</td>
</tr>
<tr>
<td>(10,10,10)</td>
<td>57421.5(54655.0563) 911.9321(1098.6795)</td>
<td>135.6(72.0990) 3.2027(1.5347)</td>
</tr>
</tbody>
</table>

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.
Figure 1. Numerical comparisons between our algorithm and that of Jiao et al. [40] on the average number of iteration (left) and the average CPU time (right) for Problem 1 with the fixed $m = 10$ and $n = 10$.

Figure 2. Numerical comparisons between our algorithm and that of Jiao et al. [40] on the average number of iteration (left) and the average CPU time (right) for Problem 1 with the fixed $p = 2$ and $m = 10$.

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).
Table 4. Numerical computational results of our algorithm for Problem 2.

<table>
<thead>
<tr>
<th>(n,n)</th>
<th>p = 4</th>
<th>p = 5</th>
<th>p = 6</th>
<th>p = 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>(20,10)</td>
<td>Avg(Std).N: 37.2(22.051)</td>
<td>96.9(84.0918)</td>
<td>159.7(94.3752)</td>
<td>302.6(240.4973)</td>
</tr>
<tr>
<td></td>
<td>Avg(Std).T: 3.0331(2.3414)</td>
<td>6.8133(4.6833)</td>
<td>12.4855(9.3914)</td>
<td>18.3431(13.7249)</td>
</tr>
<tr>
<td>(40,20)</td>
<td>Avg(Std).N: 57.9(62.9964)</td>
<td>68.7(30.2032)</td>
<td>116.7(64.0365)</td>
<td>289.2(338.0614)</td>
</tr>
<tr>
<td></td>
<td>Avg(Std).T: 3.7998(3.9706)</td>
<td>5.1644(2.0187)</td>
<td>9.9100(4.1113)</td>
<td>23.0972(22.1184)</td>
</tr>
</tbody>
</table>

Table 5. Numerical comparisons between the algorithm of Zhang et al. [29] and our algorithm on Problem 2.

<table>
<thead>
<tr>
<th>(p,m,n)</th>
<th>Algorithm of Zhang et al. [29]</th>
<th>Our new algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2,10,1000)</td>
<td>15.5</td>
<td>2.6293</td>
</tr>
<tr>
<td>(2,10,2000)</td>
<td>28.5</td>
<td>14.0012</td>
</tr>
<tr>
<td>(3,10,1000)</td>
<td>101.8</td>
<td>19.3235</td>
</tr>
<tr>
<td>(3,10,2000)</td>
<td>185.4</td>
<td>90.3898</td>
</tr>
<tr>
<td>(4,10,1000)</td>
<td>757.6</td>
<td>156.5649</td>
</tr>
<tr>
<td>(4,10,2000)</td>
<td>353.1</td>
<td>995.4707</td>
</tr>
</tbody>
</table>

References

Figure 3. Numerical comparisons between our algorithm and that of Zhang et al. [29] on the average number of iteration (left) and the average CPU time (right) for Problem 2 with the fixed $m = 10$ and $n = 2000$.

Table 6. Numerical comparisons between the software BARON and our algorithm on Problem 3.

<table>
<thead>
<tr>
<th>(p,m,n)</th>
<th>Our algorithm</th>
<th>BARON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg[Std].N</td>
<td>Avg[Std].T</td>
</tr>
<tr>
<td>(2,10,20)</td>
<td>38.1(16.1483)</td>
<td>2.3581(1.2124)</td>
</tr>
<tr>
<td>(2,20,20)</td>
<td>16.1(12.9825)</td>
<td>1.3093(1.0285)</td>
</tr>
<tr>
<td>(2,22,20)</td>
<td>14.0(13.0979)</td>
<td>0.6381(0.6014)</td>
</tr>
<tr>
<td>(2,20,30)</td>
<td>21.2(17.5233)</td>
<td>1.7240(1.5391)</td>
</tr>
<tr>
<td>(2,35,50)</td>
<td>27.2(23.8225)</td>
<td>2.6217(2.6083)</td>
</tr>
<tr>
<td>(2,45,60)</td>
<td>21.4(19.0741)</td>
<td>2.3570(1.9797)</td>
</tr>
<tr>
<td>(2,45,100)</td>
<td>34.4(24.5909)</td>
<td>4.1453(3.1211)</td>
</tr>
<tr>
<td>(2,60,100)</td>
<td>22.6(17.3282)</td>
<td>4.0992(3.8821)</td>
</tr>
<tr>
<td>(2,70,100)</td>
<td>27.1(22.0477)</td>
<td>5.1873(4.8660)</td>
</tr>
<tr>
<td>(2,70,120)</td>
<td>16.7(19.1256)</td>
<td>3.4148(3.8112)</td>
</tr>
<tr>
<td>(2,100,100)</td>
<td>15.8(13.2816)</td>
<td>5.3048(4.5256)</td>
</tr>
</tbody>
</table>


### Table 7. Numerical computational results of our algorithm for Problem 3.

<table>
<thead>
<tr>
<th>(m, n)</th>
<th>( p = 4 )</th>
<th>( p = 5 )</th>
<th>( p = 6 )</th>
<th>( p = 7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(10,20)</td>
<td>Avg(Std).N 247.2(263.87)</td>
<td>354.4(356.52)</td>
<td>2729.2(2592.85)</td>
<td>3717.3(3263.63)</td>
</tr>
<tr>
<td></td>
<td>Avg(Std).T 9.86(10.64)</td>
<td>12.97(11.60)</td>
<td>95.65(89.26)</td>
<td>137.62(119.17)</td>
</tr>
<tr>
<td>(20,40)</td>
<td>Avg(Std).N 214.2(173.77)</td>
<td>1014.6(818.77)</td>
<td>2023.1(1842.23)</td>
<td>5409.6(5291.86)</td>
</tr>
<tr>
<td></td>
<td>Avg(Std).T 10.85(8.75)</td>
<td>48.42(37.25)</td>
<td>93.76(85.29)</td>
<td>270.44(268.75)</td>
</tr>
<tr>
<td>(30,60)</td>
<td>Avg(Std).N 401.5(452.72)</td>
<td>1181.5(894.69)</td>
<td>3625.8(3930.85)</td>
<td>9624.9(12243.0)</td>
</tr>
<tr>
<td></td>
<td>Avg(Std).T 26.12(30.83)</td>
<td>73.15(56.78)</td>
<td>229.65(238.37)</td>
<td>662.27(853.38)</td>
</tr>
<tr>
<td>(40,80)</td>
<td>Avg(Std).N 370.7(354.64)</td>
<td>989.4(1074.01)</td>
<td>4971.7(6468.19)</td>
<td>13029.5(14689.56)</td>
</tr>
<tr>
<td></td>
<td>Avg(Std).T 32.53(30.95)</td>
<td>82.32(82.72)</td>
<td>549.76(729.55)</td>
<td>1397.89(2680.44)</td>
</tr>
<tr>
<td>(50,100)</td>
<td>Avg(Std).N 92.1(157.08)</td>
<td>1499.9(1597.43)</td>
<td>4056.3(3662.31)</td>
<td>12680.7(20150.97)</td>
</tr>
<tr>
<td></td>
<td>Avg(Std).T 12.17(23.46)</td>
<td>221.58(233.72)</td>
<td>641.87(587.81)</td>
<td>2179.26(3445.89)</td>
</tr>
<tr>
<td>(60,120)</td>
<td>Avg(Std).N 599.2(461.27)</td>
<td>1343.6(1540.05)</td>
<td>4937.8(5198.42)</td>
<td>14711.4(40149.31)</td>
</tr>
<tr>
<td></td>
<td>Avg(Std).T 115.9(87.77)</td>
<td>267.6(297.65)</td>
<td>1039.9(1142.62)</td>
<td>3222.76(4559.49)</td>
</tr>
<tr>
<td>(70,140)</td>
<td>Avg(Std).N 541.8(530.41)</td>
<td>801.0(539.52)</td>
<td>3218.9(5362.54)</td>
<td>945.5(5927.61)</td>
</tr>
<tr>
<td></td>
<td>Avg(Std).T 147.4(140.14)</td>
<td>191.96(126.11)</td>
<td>884.13(1573.26)</td>
<td>2612.33(1648.91)</td>
</tr>
<tr>
<td>(80,160)</td>
<td>Avg(Std).N 176.7(127.17)</td>
<td>1670.9(1999.67)</td>
<td>4821.6(5773.35)</td>
<td>4935.3(6383.97)</td>
</tr>
<tr>
<td></td>
<td>Avg(Std).T 56.04(42.38)</td>
<td>600.18(685.37)</td>
<td>1887.85(2183.56)</td>
<td>1871.4(2504.29)</td>
</tr>
<tr>
<td>(90,180)</td>
<td>Avg(Std).N 221.5(201.11)</td>
<td>1792.6(2208.91)</td>
<td>7978.9(14937.33)</td>
<td>12709.0(14685.25)</td>
</tr>
<tr>
<td></td>
<td>Avg(Std).T 96.01(87.51)</td>
<td>865.49(1062.07)</td>
<td>4084.24(7775.46)</td>
<td>6650.20(7774.70)</td>
</tr>
<tr>
<td>(100,200)</td>
<td>Avg(Std).N 509.2(438.99)</td>
<td>1654.3(2247.93)</td>
<td>4344.6(5054.17)</td>
<td>5312.3(5370.92)</td>
</tr>
<tr>
<td></td>
<td>Avg(Std).T 302.82(264.99)</td>
<td>969.45(1329.17)</td>
<td>2733.13(3079.83)</td>
<td>3607.57(3481.61)</td>
</tr>
</tbody>
</table>


TITLE WILL BE SET BY THE PUBLISHER 23

APPENDIX

A.1 (Ref.[35]).
\[
\begin{align*}
\min & \quad (0.813396y_1 + 0.67440y_2 + 0.305038y_3 + 0.129742y_4 + 0.217796) \\
& \times (0.224508y_1 + 0.063458y_2 + 0.932230y_3 + 0.528736y_4 + 0.091947) \\
\text{s.t.} & \quad 0.488509y_1 + 0.063458y_2 + 0.945686y_3 + 0.210704y_4 \leq 3.562809, \\
& \quad -0.324014y_1 - 0.501754y_2 - 0.719204y_3 + 0.099562y_4 \leq -0.052215, \\
& \quad 0.445225y_1 - 0.346896y_2 + 0.637939y_3 - 0.257623y_4 \leq 0.427920, \\
& \quad -0.202821y_1 + 0.647361y_2 + 0.920135y_3 - 0.983091y_4 \leq 0.840950, \\
& \quad -0.886420y_1 - 0.802444y_2 - 0.305441y_3 - 0.180123y_4 \leq -1.353686, \\
& \quad -0.515399y_1 - 0.424820y_2 + 0.897498y_3 + 0.187268y_4 \leq 2.137251, \\
& \quad -0.591515y_1 + 0.060581y_2 - 0.427365y_3 + 0.579388y_4 \leq -0.290987, \\
& \quad 0.423524y_1 + 0.940496y_2 - 0.437944y_3 - 0.742941y_4 \leq 0.373620, \\
y_1 \geq 0, y_2 \geq 0, y_3 \geq 0, y_4 \geq 0.
\end{align*}
\]

A.2 (Ref.[36]).
\[
\begin{align*}
\min & \quad (-y_1 + 2y_2 + 2)(4y_1 - 3y_2 + 4)(3y_1 - 4y_2 + 5)^{-1}(-2y_1 + y_2 + 3)^{-1} \\
\text{s.t.} & \quad y_1 + y_2 \leq 1.5, \quad 0 \leq y_1 \leq 1, \quad 0 \leq y_2 \leq 1.
\end{align*}
\]

A.3 (Ref.[37]).
\[
\begin{align*}
\min & \quad (y_1 + y_2)(y_1 - y_2 + 7) \\
\text{s.t.} & \quad 2y_1 + y_2 \leq 14, \quad y_1 + y_2 \leq 10, \\
& \quad -4y_1 + y_2 \leq 0, \quad 2y_1 + y_2 \geq 6, \quad y_1 + 2y_2 \geq 6, \\
& \quad y_1 - y_2 \leq 3, \quad y_1 + y_2 \geq 0, \\
& \quad y_1 - y_2 + 7 \geq 0, \quad y_1, y_2 \geq 0.
\end{align*}
\]

A.4 (Ref.[21]).
\[
\begin{align*}
\min & \quad (y_1 + y_2 + 1)^{2.5}(2y_1 + y_2 + 1)^{1.1}(y_1 + 2y_2 + 1)^{1.9} \\
\text{s.t.} & \quad y_1 + 2y_2 \leq 6, \quad 2y_1 + 2y_2 \leq 8, \\
& \quad 1 \leq y_1 \leq 3, \quad 1 \leq y_2 \leq 3.
\end{align*}
\]

A.5 (Ref.[21]).
\[
\begin{align*}
\min & \quad (3y_1 - 4y_2 + 5)(y_1 + 2y_2 - 1)^{0.5}(2y_1 - y_2 + 4)(y_1 - 2y_2 + 8)^{0.5}(2y_1 + y_2 - 1) \\
\text{s.t.} & \quad 5y_1 - 8y_2 \geq -24, \quad 5y_1 + 8y_2 \leq 44, \\
& \quad 6y_1 - 3y_2 \leq 15, \quad 4y_1 + 5y_2 \geq 10, \\
& \quad 1 \leq y_1 \leq 3, \quad 0 \leq y_2 \leq 1.
\end{align*}
\]

A.6 (Ref.[21]).
\[
\begin{align*}
\min & \quad (3y_1 - 2y_2 - 2)^{\frac{3}{2}}(y_1 + 2y_2 + 2)^{\frac{3}{2}} \\
\text{s.t.} & \quad 2y_1 - y_2 \geq 2, \quad y_1 - 2y_2 \leq 2, \\
& \quad y_1 + y_2 \leq 5, \quad 3 \leq y_1 \leq 5, \quad 1 \leq y_2 \leq 3.
\end{align*}
\]
A.7 (Ref.[38]).

\[
\begin{align*}
&\min \ (y_1 + \frac{1}{9} y_3)(y_2 + \frac{1}{7} y_3) \\
&s.t. \quad 9y_1 + 9y_2 + 2y_3 \leq 81, \\
&\quad 8y_1 + y_2 + 8y_3 \leq 72, \\
&\quad y_1 + 8y_2 + 8y_3 \leq 72, \\
&\quad 7y_1 + y_2 + y_3 \geq 9, \\
&\quad y_1 + 7y_2 + y_3 \geq 9, \\
&\quad y_1 + y_2 + 7y_3 \geq 9, \\
&\quad 0 \leq y_1 \leq 8, \ 0 \leq y_2 \leq 8, \ 0 \leq y_3 \leq 9.
\end{align*}
\]

A.8 (Ref.[29]).

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