Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 19, No. 1, pp. 97–118, 2023

PIECEWISE LINEAR RELAXATION METHOD FOR GLOBALLY SOLVING A CLASS OF MULTIPLICATIVE PROBLEMS*

Hongwei Jiao†, Wenjie Wang and Peiping Shen

Abstract: This paper presents an image space branch-and-bound algorithm for globally solving a class of multiplicative problems (MP). In the algorithm, by introducing new variables and the logarithmic transformation, we firstly transform the problem (MP) into an equivalent problem (EP1). Next, by using the properties of the logarithmic function, we propose a piecewise linear relaxation method for constructing the linear relaxation problem of the problem (EP1), which can be used to compute the lower bound of the optimal value of the problem (EP1). Furthermore, to improve the convergence speed of the algorithm, some image space region reduction techniques are added into the branch-and-bound algorithm. Finally, the global convergence of the algorithm is proved and the maximum number of iterations of the algorithm is estimated by analyzing its computational complexity, and numerical results show the feasibility and validity of the algorithm.

Key words: *multiplicative problem, global optimization, piecewise linear relaxation method, image space region reduction techniques, computational complexity*

Mathematics Subject Classification: *90C26, 65K05*

1 Introduction

This paper investigates the following multiplicative problem:

$$(\mathrm{MP}): \begin{cases} \min & f(x) = \prod_{j=1}^{p} (\sum_{i=1}^{n} c_{ji} x_i + d_j)^{\gamma_j}, \\ \text{s.t.} & x \in X = \{x \in \mathbb{R}^n | Ax \leq b\}, \end{cases}$$

where c_{ji}, d_j and γ_j are arbitrary real numbers, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and X is a nonempty bounded set, and for any $x \in X$, $\sum_{i=1}^{n} c_{ji} x_i + d_j > 0, j = 1, 2, \ldots, p$.

During the past years, the problem (MP) and its special cases have attracted wide attentions of many researchers and practitioners. On the one hand, the problem (MP) has a wide range of applications in data analysis and processing, system engineering, financial optimization, robust optimization, VLISI chip design, decision tree optimization, and so on

*This work was supported by the National Natural Science Foundation of China (11871196; 12071133), 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).

†Corresponding author.

[10, 22, 14, 9, 15, 4, 2]. On the other hand, the problem (MP) generally exists many local optimal solutions which are not global optimal.

In the last two decades, a large number of algorithms have been proposed for globally solving some special forms of the problem (MP). In general, these algorithms can be classified into the following categories: branch-and-bound algorithms [17, 3, 13, 30, 31, 5, 16, 27, 6, 24], outer-approximation algorithm [11], level set algorithm [12], polynomial time approximation method [19], cutting plane method [1], and so on. In addition to the algorithms mentioned in the above comments, especially recently, Shen et al. [20] proposed an outer space branch-and-bound algorithm for solving the linear multiplicative programming problem by utilizing the linear relaxation technique and the branch-and-bound scheme; Wang and Liu [25] proposed a D.C. decomposition linearization method for a generalized linear multiplicative programming problem; Zhang et al. [28] presented an output-space branch-and-bound reduction algorithm for the generalized linear multiplicative programming problem by utilizing a new relaxation bounding method; Jiao et al. [7] proposed an outer space branch-and-bound algorithm for the generalized linear multiplicative programming problem. Shen and Huang [18] proposed a rectangular branch-and-bound algorithm with standard bisection rule for solving the linear multiplicative problem by using a new linear programming relaxation. By dividing the outer space of the generalized linear multiplicative programming problem into finite polynomial rectangles and a new two-stage acceleration technique, Zhang et al. [29] proposed an efficient polynomial time algorithm for a class of generalized linear multiplicative programs with positive exponents. Shen et al. [21] proposed a branch and bound algorithm for globally solving the linear multiplicative problem based on the characteristics of the initial problem. However, the above reviewed methods can only deal with some particular cases of the problem (MP). Therefore, it is very necessary to propose a new algorithm for solving the problem (MP) in its general form.

In this paper, we will present an image space branch-and-bound algorithm for globally solving the problem (MP). First of all, we convert the problem (MP) into the equivalent problem (EP1) by introducing new variables and the logarithmic transformation. Next, by utilizing the piecewise linear relaxation method, we derive the linear relaxation problem (LRP) of the problem (EP1). To improve the computational efficiency of the algorithm, we construct some image space region reduction techniques by using the variable separability of the objective function of the problem (EP1) and the framework of the branch-and-bound algorithm. Based on the branch-and-bound framework, an image space branch-and-bound algorithm is established by combining the linear relaxation problem and the image space region reduction techniques. Compared with the existing branch-and-bound methods, the proposed algorithm has the following potential practical and computational advantages: (i) The branch-and-bound search takes place in the p-dimensional image space of linear product functions, which greatly mitigates the required computational efforts; (ii) A new piecewise linear relaxation method is proposed for deriving the linear relaxation problems with higher compactness; (iii) Some new image space region reduction techniques are proposed for improving the computational efficiency of the algorithm; (iv) The computational complexity of the algorithm is analysed and the maximum iterations of the algorithm are estimated for the first time; (v) The numerical results show that the proposed algorithm can effectively find the globally optimal solutions of all test instances with better computational performance.

The structure of this paper is given as follows. In Section 2, the equivalent problem (EP1) of the problem (MP) is derived, and a new piecewise linear relaxation method is proposed for establishing the linear relaxation problem of the problem (EP1). In Section 3, some image space region reduction techniques are constructed for improving the computational efficiency of the algorithm, an image space branch-and-bound algorithm is presented, the

global convergence of the algorithm is proved, and its computational complexity is analyzed. Numerical results for some test examples in recent literatures are reported in Section 4. Finally, some conclusions are provided in Section 5.

2 Piecewise Linear Relaxation Method

To globally solve the problem (MP), we firstly convert the problem (MP) into an equivalent problem (EP). Next, by utilizing the logarithmic transformation, we transform the problem (EP) into the equivalent problem (EP1), which has the same optimal solution as the problem (EP).

Without losing generality, for each $j = 1, 2, \ldots, p$, introducing the variable $t_j = \sum_{i=1}^{n} c_{ji}x_i + d_j$, and letting

$$\begin{aligned} l_j^0 = \min \quad & \textstyle\sum_{i=1}^{n} c_{ji}x_i + d_j \\ \text{s.t.} \quad & Ax \le b, \\ & \textstyle\sum_{i=1}^{n} c_{ji}x_i + d_j > 0, \end{aligned}$$

$$\begin{aligned} u_j^0 = \max \quad & \textstyle\sum_{i=1}^{n} c_{ji}x_i + d_j \\ \text{s.t.} \quad & Ax \le b, \\ & \textstyle\sum_{i=1}^{n} c_{ji}x_i + d_j > 0, \end{aligned}$$

we can construct the initial rectangle $T^0 = \{t \in \mathbb{R}^p | l_j^0 \le t_j \le u_j^0, j = 1, 2, \ldots, p\}$, and the problem (MP) can be transformed into the following equivalent problem:

$$(\text{EP}) : \begin{cases} \min & g(t) = \prod_{j=1}^{p} (t_j)^{\gamma_j}, \\ \text{s.t.} & t_j = \sum_{i=1}^{n} c_{ji}x_i + d_j, \ j = 1, 2, \ldots, p, \\ & x \in X, t \in T^0. \end{cases}$$

Remark 2.1. If $(x^*, t_1^*, t_2^*, \ldots, t_p^*)$ is a global optimal solution to the problem (EP), then x^* is a global optimal solution to the problem (MP), where $t_j^* = \sum_{i=1}^{n} c_{ji}x_i^* + d_j, j = 1, 2, \ldots, p$. In turn, if x^* is a global optimal solution to the problem (MP), let $t_j^* = \sum_{i=1}^{n} c_{ji}x_i^* + d_j, j = 1, 2, \ldots, p$, then $(x^*, t_1^*, t_2^*, \ldots, t_p^*)$ is a global optimal solution to the problem (EP).

By applying the logarithmic transformation, we can convert the problem (EP) into the following equivalent problem (EP1):

$$(\text{EP1}) : \begin{cases} \min & s(t) = \sum_{j=1}^{p} \gamma_j \ln t_j, \\ \text{s.t.} & t_j = \sum_{i=1}^{n} c_{ji}x_i + d_j, j = 1, 2, \ldots, p, \\ & x \in X, \ t \in T^0. \end{cases}$$

Obviously, the problems (EP1) and (EP) have the same optimal solutions and optimal values. In the following, the main work is to solve the problem (EP1), so that we need to construct the linear relaxation problem (LRP) of the problem (EP1), which can provide a reliable lower bound for the optimal value of the problem (EP1) in the branch-and-bound searching process.

Without losing generality, for convenience, for any $t \in T = \{t \in \mathbb{R}^p | l_j \le t_j \le u_j, j = 1, 2, \ldots, p\} \subseteq T^0$, the following symbols are introduced.

$$\begin{aligned} & \delta_j(t_j) = \ln t_j, \quad k_{1j} = \frac{\ln \frac{l_j+u_j}{2} - \ln l_j}{\frac{l_j+u_j}{2} - l_j}, \quad k_{2j} = \frac{\ln u_j - \ln \frac{l_j+u_j}{2}}{u_j - \frac{l_j+u_j}{2}}, \\ & \delta_j^l(t_j) = \min\{\ln l_j + k_{1j}(t_j - l_j), \ln \tfrac{l_j+u_j}{2} + k_{2j}(t_j - \tfrac{l_j+u_j}{2})\}, \\ & \delta_j^u(t_j) = \min\{k_{1j}t_j - 1 - \ln k_{1j}, k_{2j}t_j - 1 - \ln k_{2j}\}. \end{aligned}$$

By utilizing the geometric properties of the logarithmic function, we need to construct the linear relaxation problem of the problem (EP1), and the following Theorem 2.2 presents an effective method for constructing the linear relaxation problem.

Theorem 2.2. *For any $t_j \in [l_j, u_j], j = 1, 2, \ldots, p$, let $\omega_j = u_j - l_j$, consider the functions $\delta_j^l(t_j)$, $\delta_j(t_j)$, and $\delta_j^u(t_j)$, $j = 1, 2, \ldots, p$, we have the following conclusions:*

(i) $\delta_j^l(t_j) \le \delta_j(t_j) \le \delta_j^u(t_j), j = 1, 2, \ldots, p$;

(ii) $\lim_{\omega_j \to 0}[\delta_j(t_j) - \delta_j^l(t_j)] = \lim_{\omega_j \to 0}[\delta_j^u(t_j) - \delta_j(t_j)]$.

Proof. (i) For any $t_j \in [l_j, \frac{l_j+u_j}{2}], j = 1, 2, \ldots, p$, Since the function $\ln t_j$ is a concave and monotone increase function over $[l_j, \frac{l_j+u_j}{2}]$, its convex envelope is $\ln l_j + k_{1j}(t_j - l_j)$. Since the tangential supporting function for $\ln t_j$ is parallel with the $\ln l_j + k_{1j}(t_j - l_j)$, thus the point of tangential support will occur at $\frac{1}{k_{1j}}$, and the corresponding tangential supporting function is $k_{1j}t_j - 1 - \ln k_{1j}$. Therefore, by the geometric property of the logarithmic function $\delta_j(t_j)$, we have

$$\ln l_j + k_{1j}(t_j - l_j) \le \ln t_j \le k_{1j}t_j - 1 - \ln k_{1j}.$$

Similarly, for any $t_j \in [\frac{l_j+u_j}{2}, u_j], j = 1, 2, \ldots, p$, by the geometric property of the logarithmic function $\delta_j(t_j)$, we have

$$\ln \frac{l_j + u_j}{2} + k_{2j}(t_j - \frac{l_j + u_j}{2}) \le \ln t_j \le k_{2j}t_j - 1 - \ln k_{2j}.$$

Therefore, for any $t_j \in [l_j, u_j], j = 1, 2, \ldots, p$, we have that

$$\begin{aligned} \delta_j^l(t_j) &= \min\{\ln l_j + k_{1j}(t_j - l_j), \ln \tfrac{l_j+u_j}{2} + k_{2j}(t_j - \tfrac{l_j+u_j}{2})\} \\ &\le \ln t_j = \delta_j(t_j) \\ &\le \min\{k_{1j}t_j - 1 - \ln k_{1j}, k_{2j}t_j - 1 - \ln k_{2j}\} \\ &= \delta_j^u(t_j), \end{aligned}$$

i.e.,

$$\delta_j^l(t_j) \le \delta_j(t_j) \le \delta_j^u(t_j).$$

(ii) When $t_j \in [l_j, \frac{l_j+u_j}{2}]$, we have $\delta_j^l(t_j) = \ln l_j + k_{1j}(t_j - l_j)$, which is a linear function about t_j, also since the function $\delta_j(t_j)$ is a concave function, we have that $\delta_j(t_j) - \delta_j^l(t_j)$ is a concave function about t_j over $[l_j, \frac{l_j+u_j}{2}]$. Thus, the maximum value of $\delta_j(t_j) - \delta_j^l(t_j)$ over $\left[l_j, \frac{l_j+u_j}{2}\right]$ can be obtained at the point of tangential support hyperplane $\frac{1}{k_{1j}}$. Similarly, we can prove that, when $t_j \in \left[\frac{l_j+u_j}{2}, u_j\right]$, $\delta_j(t_j) - \delta_j^l(t_j)$ is also a concave function about t_j over $\left[\frac{l_j+u_j}{2}, u_j\right]$, the maximum value of $\delta_j(t_j) - \delta_j^l(t_j)$ over $\left[\frac{l_j+u_j}{2}, u_j\right]$ can be also obtained at the point of tangential support hyperplane $\frac{1}{k_{2j}}$. Therefore, we have

$$\begin{aligned} &\max[\delta_j(t_j) - \delta_j^l(t_j)] \\ &= \max\left\{\ln t_j - \min\{\ln l_j + k_{1j}(t_j - l_j), \ln \tfrac{l_j+u_j}{2} + k_{2j}(t_j - \tfrac{l_j+u_j}{2})\}\right\} \\ &= \max\left\{-\ln k_{1j} - \ln l_j - 1 + k_{1j}l_j, -\ln k_{2j} - \ln \tfrac{l_j+u_j}{2} - 1 + k_{2j}\tfrac{l_j+u_j}{2}\right\} \\ &= \max\left\{\ln \frac{\left(\frac{l_j+u_j}{2l_j}\right)-1}{\ln\left(\frac{l_j+u_j}{2l_j}\right)} - 1 + \frac{\ln\left(\frac{l_j+u_j}{2l_j}\right)}{\left(\frac{l_j+u_j}{2l_j}\right)-1}, \ln \frac{\left(\frac{u_j}{l_j+u_j}{2}\right)-1}{\ln\left(\frac{u_j}{l_j+u_j}{2}\right)} - 1 + \frac{\ln\left(\frac{u_j}{l_j+u_j}{2}\right)}{\left(\frac{u_j}{l_j+u_j}{2}\right)-1}\right\}. \end{aligned}$$

Since $\frac{\frac{l_j+u_j}{2}}{l_j} \to 1$ or $\frac{u_j}{\frac{l_j+u_j}{2}} \to 1$ as $\omega_j \to 0$, so that we can obtain that $\max[\delta_j(t_j)-\delta_j^l(t_j)] \to 0$ as $\omega_j \to 0$.

When $t_j \in [l_j, \frac{l_j+u_j}{2}]$, we have $\delta_j^u(t_j) = k_{1j}t_j - 1 - \ln k_{1j}$, which is a linear function about t_j, also since the function $-\delta_j(t_j)$ is a convex function about t_j, we have that $\delta_j^u(t_j) - \delta_j(t_j)$ is a convex function about t_j over $[l_j, \frac{l_j+u_j}{2}]$. Thus, the maximum value of the function $\delta_j^u(t_j) - \delta_j(t_j)$ over $\left[l_j, \frac{l_j+u_j}{2}\right]$ can be achieved at the point l_j or $\frac{l_j+u_j}{2}$. Similarly, we can prove that, when $t_j \in \left[\frac{l_j+u_j}{2}, u_j\right]$, $\delta_j^u(t_j) - \delta_j(t_j)$ is also a convex function about t_j, the maximum value of the function $\delta_j^u(t_j) - \delta_j(t_j)$ over $\left[\frac{l_j+u_j}{2}, u_j\right]$ can be achieved at the point $\frac{l_j+u_j}{2}$ or u_j. Therefore, we have

$$\begin{aligned}
&\delta_j^u(l_j) - \delta_j(l_j) = k_{1j}l_j - 1 - \ln k_{1j} - \ln l_j,\\
&\delta_j^u(\tfrac{l_j+u_j}{2}) - \delta_j(\tfrac{l_j+u_j}{2}) = k_{1j}\tfrac{l_j+u_j}{2} - 1 - \ln k_{1j} - \ln \tfrac{l_j+u_j}{2},\\
&\delta_j^u(\tfrac{l_j+u_j}{2}) - \delta_j(\tfrac{l_j+u_j}{2}) = k_{2j}\tfrac{l_j+u_j}{2} - 1 - \ln k_{2j} - \ln \tfrac{l_j+u_j}{2},\\
&\delta_j^u(u_j) - \delta_j(u_j) = k_{2j}u_j - 1 - \ln k_{2j} - \ln u_j.
\end{aligned}$$

By $k_{1j}l_j - \ln l_j = k_{1j}\frac{l_j+u_j}{2} - \ln \frac{l_j+u_j}{2}$ and $k_{2j}\frac{l_j+u_j}{2} - \ln \frac{l_j+u_j}{2} = k_{2j}u_j - \ln u_j$, we have that

$$\delta_j^u(l_j) - \delta_j(l_j) = \delta_j^u(\frac{l_j+u_j}{2}) - \delta_j(\frac{l_j+u_j}{2}) = \delta_j^u(u_j) - \delta_j(u_j).$$

So that

$$\begin{aligned}
&\max[\delta_j^u(t_j) - \delta_j(t_j)]\\
&= \max[k_{1j}l_j - 1 - \ln k_{1j} - \ln l_j, k_{2j}u_j - 1 - \ln k_{2j} - \ln u_j]\\
&= \max[-\ln k_{1j} - \ln l_j - 1 + k_{1j}l_j, -\ln k_{2j} - \ln \tfrac{l_j+u_j}{2} - 1 + k_{2j}\tfrac{l_j+u_j}{2}]\\
&= \max[\delta_j(t_j) - \delta_j^l(t_j)].
\end{aligned}$$

Therefore, we can obtain that $\max[\delta_j^u(t_j)-\delta_j(t_j)] \to 0$ as $\omega_j \to 0$, and the proof is completed. □

To better illustrate the piecewise linear lower bound $\delta_j^l(t_j)$ and the piecewise linear upper bound $\delta_j^u(t_j)$ of the function $\delta_j(t_j)$ in the revised manuscript, we give the following Figure 1.

Next, denoting $\phi(t)$ as the underestimating function of the function $s(t)$, by Theorem 2.2, for any $t \in T$, we have

$$s(t) = \sum_{j=1}^{p} \gamma_j \ln t_j \geq \sum_{j=1,\gamma_j>0}^{p} \gamma_j \delta_j^l(t_j) + \sum_{j=1,\gamma_j<0}^{p} \gamma_j \delta_j^u(t_j) = \phi(t).$$

Therefore, by the above discussions, for any $T \subseteq T^0$, we construct the relaxation problem (RP) of the problem (EP1) as follows:

$$(\mathrm{RP}): \begin{cases} \min & \phi(t) = \sum_{j=1,\gamma_j>0}^{p} \gamma_j \delta_j^l(t_j) + \sum_{j=1,\gamma_j<0}^{p} \gamma_j \delta_j^u(t_j),\\ \text{s.t.} & t_j = \sum_{i=1}^{n} c_{ji}x_i + d_j, j = 1, 2, \ldots, p,\\ & x \in X,\ t \in T, \end{cases}$$

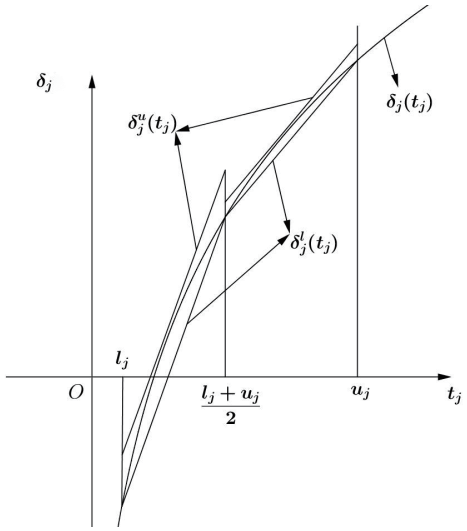

Figure 2.1: The piecewise linear lower bound $\delta_j^l(t_j)$ and the piecewise linear upper bound $\delta_j^u(t_j)$ of the function $\delta_j(t_j)$

where

$$\delta_j^l(t_j) = \min\{\ln l_j + k_{1j}(t_j - l_j), \ln\frac{l_j+u_j}{2} + k_{2j}(t_j - \frac{l_j+u_j}{2})\},$$
$$\delta_j^u(t_j) = \min\{k_{1j}t_j - 1 - \ln k_{1j}, k_{2j}t_j - 1 - \ln k_{2j}\}.$$

Remark 2.3. Obviously, the feasible set of the problem (RP) is the same as that of the problem (EP1) over T, and $\phi(t) \le s(t)$. Therefore, the global optimal value of the problem (RP) provides a lower bound for the global optimal value of the problem (EP1) over T.

By introducing new variables $\lambda_j, j = 1, 2, \ldots, p$, the problem (RP) can be rewritten as the following equivalent linear relaxation problem (LRP):

$$(\text{LRP}): \begin{cases} \min & \phi(t) = \sum_{j=1}^{p} \gamma_j \lambda_j, \\ \text{s.t.} & \lambda_j \le \ln l_j + k_{1j}(t_j - l_j), \text{ if } \gamma_j > 0, j = 1, 2, \ldots, p, \\ & \lambda_j \le \ln\frac{l_j+u_j}{2} + k_{2j}(t_j - \frac{l_j+u_j}{2}), \text{ if } \gamma_j > 0, j = 1, 2, \ldots, p, \\ & \lambda_j \le k_{1j}t_j - 1 - \ln k_{1j}, \text{ if } \gamma_j < 0, j = 1, 2, \ldots, p, \\ & \lambda_j \le k_{2j}t_j - 1 - \ln k_{2j}, \text{ if } \gamma_j < 0, j = 1, 2, \ldots, p, \\ & t_j = \sum_{i=1}^{n} c_{ji}x_i + d_j, j = 1, 2, \ldots, p. \\ & x \in X, \ t \in T. \end{cases}$$

Thus, in the process of iterative solution, we can solve the problem (LRP) instead of solving the problem (RP).

3 Algorithm and its Complexity Analysis

In this section, we first present a rectangle branching rule and some image space region reduction techniques. Next, an image space branch-reduction-bound algorithm is proposed for solving the problem (EP1) based on the former linear relaxation problem. Finally, we give the detail proof of the convergence and the complexity analysis of the algorithm.

3.1 Branching rule

Consider any node subproblem identified by the rectangle $T = \{t \in \mathbb{R}^p | l_j \le t_j \le u_j, j = 1, 2, \ldots, p\} \subseteq T^0$. The proposed branching rule is given as follows:

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

(ii) We subdivide the T into two sub-rectangles

$$\overline{T} = \{t \in \mathbb{R}^p | l_j \le t_j \le \frac{l_j + u_j}{2}, j = q; l_j \le t_j \le u_j, j = 1, 2, \ldots, p, j \ne q\}$$

and

$$\overline{\overline{T}} = \{t \in \mathbb{R}^p | \frac{l_j + u_j}{2} \le t_j \le u_j, j = q; l_j \le t_j \le u_j, j = 1, 2, \ldots, p, j \ne q\}.$$

By the branching rule, the rectangle T can be partitioned into two sub-rectangles $\overline{T}$ and $\overline{\overline{T}}$.

3.2 Image space region reduction techniques

In this subsection, some image space region reduction techniques are constructed for improving the convergence speed of the algorithm.

Without loss of generality, for any $t \in T = \{t \in \mathbb{R}^p | l_j \le t_j \le u_j, j = 1, 2, \ldots, p\} \subseteq T^0$, denote $\overline{UB}$ as the current upper bound of the global optimal value of the problem (EP1), let

$$\underline{LB} = \sum_{j=1,\gamma_j>0}^{p} \gamma_j \ln l_j + \sum_{j=1,\gamma_j<0}^{p} \gamma_j \ln u_j,\ E_\varrho = \sum_{j=1,j\ne\varrho,\gamma_j>0}^{p} \gamma_j \ln l_j + \sum_{j=1,j\ne\varrho,\gamma_j<0}^{p} \gamma_j \ln u_j,$$

$$\overline{T}^1 = \left\{ t \in \mathbb{R}^p | l_j \le t_j \le u_j, j = 1, 2, \ldots, p, j \ne \varrho; \exp\left(\frac{\overline{UB} - E_\varrho}{\gamma_\varrho}\right) < t_\varrho \le u_\varrho, j = \varrho \right\}$$

and

$$\overline{T}^2 = \left\{ t \in \mathbb{R}^p | l_j \le t_j \le u_j, j = 1, 2, \ldots, p, j \ne \varrho; l_\varrho \le t_\varrho < \exp\left(\frac{\overline{UB} - E_\varrho}{\gamma_\varrho}\right), j = \varrho \right\},$$

the detailed image space region reduction techniques are discussed in two cases as follows.

Theorem 3.1. *For any $T = \{t \in \mathbb{R}^p | l_j \le t_j \le u_j, j = 1, 2, \ldots, p\} \subseteq T^0$, we have the following conclusions:*

(i) *If $\underline{LB} > \overline{UB}$, then there exists no globally optimal solution to the problem (EP1) over T.*

(ii) *If $\underline{LB} \le \overline{UB}$, then we have the following two cases:*

if there exists some $\varrho \in \{1, 2, \ldots, p\}$ satisfying that $\gamma_\varrho > 0$ and $l_\varrho \le \exp\left(\frac{\overline{UB} - E_\varrho}{\gamma_\varrho}\right) \le u_\varrho$, then $\overline{T}^1$ contains no globally optimal solution to the problem (EP1);

if there exists some $\varrho \in \{1, 2, \ldots, p\}$ satisfying that $\gamma_\varrho < 0$ and $l_\varrho \le \exp\left(\frac{\overline{UB} - E_\varrho}{\gamma_\varrho}\right) \le u_\varrho$, then $\overline{T}^2$ contains no globally optimal solution to the problem (EP1).

Proof. (i) If $\underline{LB} > \overline{UB}$, then for any $t \in T$, we have

$$\min_{t\in T} s(t) = \min_{t\in T}\left(\sum_{j=1}^{p}\gamma_j \ln t_j\right) = \sum_{j=1,\gamma_j>0}^{p}\gamma_j \ln l_j + \sum_{j=1,\gamma_j<0}^{p}\gamma_j \ln u_j = \underline{LB} > \overline{UB}.$$

Thus, T contains no globally optimal solution to the problem (EP1).

(ii) If $\underline{LB} \le \overline{UB}$, then for any $t \in \overline{T}^1$, for some $\varrho \in \{1, 2, \ldots, p\}$, we have

$$l_j \le t_j \le u_j, j = 1, 2, \ldots, p, j \ne \varrho,\ \exp\left(\frac{\overline{UB} - E_\varrho}{\gamma_\varrho}\right) < t_\varrho \le u_\varrho, j = \varrho.$$

Thus, by $\gamma_\varrho > 0$, it follows that

$$\begin{array}{rcl} \min_{t\in\overline{T}^1} s(t) & = & \min_{t\in\overline{T}^1}\left(\sum_{j=1}^{p}\gamma_j \ln t_j\right) > \sum_{j=1,j\ne\varrho,\gamma_j>0}^{p}\gamma_j \ln l_j \\ & & + \sum_{j=1,\gamma_j<0}^{p}\gamma_j \ln u_j + \gamma_\varrho \ln\left(\exp\left(\frac{\overline{UB}-E_\varrho}{\gamma_\varrho}\right)\right) \\ & = & E_\varrho + \overline{UB} - E_\varrho \\ & = & \overline{UB}. \end{array}$$

Therefore, $\overline{T}^1$ contains no globally optimal solution to the problem (EP1).

Similarly, if $\underline{LB} \le \overline{UB}$, then for any $t \in \overline{T}^2$, for some $\varrho \in \{1, 2, \ldots, p\}$, we have

$$l_j \le t_j \le u_j, j = 1, 2, \ldots, p, j \ne \varrho,\ l_\varrho \le t_\varrho < \exp\left(\frac{\overline{UB} - E_\varrho}{\gamma_k}\right), j = \varrho.$$

Hence, by $\gamma_\varrho < 0$, it follows that

$$\begin{array}{rcl} \min_{t\in\overline{T}^2} s(t) & = & \min_{t\in\overline{T}^2}\left(\sum_{j=1}^{p}\gamma_j \ln t_j\right) \\ & > & \sum_{j=1,\gamma_j>0}^{p}\gamma_j \ln l_j + \sum_{j=1,j\ne\varrho,\gamma_j<0}^{p}\gamma_j \ln u_j + \gamma_\varrho \ln\left(\exp\left(\frac{\overline{UB}-E_\varrho}{\gamma_\varrho}\right)\right) \\ & = & E_\varrho + \overline{UB} - E_\varrho \\ & = & \overline{UB}. \end{array}$$

Therefore, $\overline{T}^2$ contains no globally optimal solution to the problem (EP1). □

Remark 3.2. According to the conclusions of Theorem 3.1, for any $T \subseteq T^0$, when $\underline{LB} > \overline{UB}$, we can let $T = \emptyset$; when $\underline{LB} \le \overline{UB}$, for each $\varrho \in \{1, 2, \ldots, p\}$, we can let $T = T\backslash\overline{T}^1$ or $T = T\backslash\overline{T}^2$. So Theorem 3.1 provides a possibility for deleting the whole or a large part of the currently investigated image space rectangles in which the global optimal solution of the problem (EP1) does not exist.

3.3 Image space branch-reduction-bound algorithm

Definition 3.3. Let (x^k, t^k) be a feasible solution of the problem (EP1), and let v be the global minimum of the problem (EP1), if $s(t^k) - v \le \epsilon$, then (x^k, t^k) is a global ϵ-optimal solution of the problem (EP1).

Based on the previous discussions, the basic steps of the algorithm are summarized as follows.

Step 0. (Initialization) Given the convergence tolerance $\epsilon > 0$ and the initial rectangle T^0.

Solve the (LRP(T^0)) to obtain its optimal solution $(x(T^0), t(T^0))$ and optimal value $LB(T^0)$. Let $LB_0 = LB(T^0)$ and $(x^0, t^0) = (x(T^0), t(T^0))$. Since (x^0, t^0) is a feasible solution of the problem (EP1), let $UB_0 = s(t^0)$.

If $UB_0 - LB_0 < \epsilon$, then the algorithm terminates with that (x^0, t^0) is a global ϵ-optimal solution of the problem (EP1) over T^0. Otherwise, we denote $F = \{(x^0, t^0)\}$ as the set of feasible points, denote $Q_0 = \{T^0\}$ as the set of all active nodes, and let $k = 0$.

Step 1. Subdivide the rectangle T^k into two new sub-rectangles $T^{k,1}$ and $T^{k,2}$, and denote by the set of new subdivided sub-rectangles $\overline{T}^k = \{T^{k,1}, T^{k,2}\}$.

Step 2. For each $T^{k,i} \in \overline{T}^k$, where $i = 1, 2$, use the image space region reduction technique to compress its interval, and still denote $\overline{T}^k$ as the set of the each remaining sub-rectangles.

Step 3. If $\overline{T}^k \neq \emptyset$, then: for $i = 1, 2$, we solve the following new linear relaxation problem:

$$
(\mathrm{NLRP}): \begin{cases} \min & \phi(t) = \sum_{j=1}^{p} \gamma_j \lambda_j, \\ \text{s.t.} & \lambda_j \le \ln l_j + k_{1j}(t_j - l_j), \text{ if } \gamma_j > 0, j = 1, 2, \ldots, p, \\ & \lambda_j \le \ln \frac{l_j + u_j}{2} + k_{2j}(t_j - \frac{l_j + u_j}{2}), \text{ if } \gamma_j > 0, j = 1, 2, \ldots, p, \\ & \lambda_j \le k_{1j} t_j - 1 - \ln k_{1j}, \text{ if } \gamma_j < 0, j = 1, 2, \ldots, p, \\ & \lambda_j \le k_{2j} t_j - 1 - \ln k_{2j}, \text{ if } \gamma_j < 0, j = 1, 2, \ldots, p, \\ & \sum_{j=1}^{p} \gamma_j \lambda_j \le UB_{k-1}, \\ & t_j = \sum_{i=1}^{n} c_{ji} x_i + d_j, j = 1, 2, \ldots, p, \\ & x \in X, \ t \in T^{k,i}, \end{cases}
$$

and obtain its optimal solution $(x(T^{k,i}), t(T^{k,i}))$ and optimal value $LB(T^{k,i})$.

If $LB(T^{k,i}) > UB_{k-1}$, then let $\overline{T}^k = \overline{T}^k \backslash T^{k,i}$.

Otherwise, update the upper bound by setting $UB_k = \min\{s(t(T^{k,i})), UB_{k-1}\}$, update the set of the feasible points by setting $F = F \bigcup \{(x(T^{k,i}), t(T^{k,i}))\}$, and denote (x^k, t^k) as the currently known best feasible solution.

Step 4. The remaining partition set is denoted by $Q_k = (Q_k \backslash T^k) \bigcup \overline{T}^k$, the new lower bound is denoted by $LB_k = \min_{T \in Q_k} LB(T)$.

Step 5. Let $Q_{k+1} = Q_k \backslash \{T | UB_k - LB(T) \le \epsilon, T \in Q_k\}$.

If $Q_{k+1} = \emptyset$, then the algorithm terminates, UB_k is a global ϵ-optimal value of the problem (EP1), and (x^k, t^k) is a global ϵ-optimal solution of the problem (EP1) over T^0.

Otherwise, select a sub-rectangle T^{k+1} satisfying that $T^{k+1} = \arg\min_{T \in Q_{k+1}} LB(T)$, let $k = k + 1$, and return to Step 1.

Remark 3.4. Obviously the objective function of the problem (LRP) is always less than or equal to the currently known upper bound, and we can obtain a new linear relaxation problem (NLRP) by adding this condition to the constraints of the problem (LRP).

3.4 Global convergence of the algorithm

In the following, we will prove the global convergence of the algorithm.

Theorem 3.5. *The proposed algorithm either terminates within a finite number of iterations, or generates an infinite sequence $\{(x^k, t^k)\}$ such that any accumulation point of the sequence $\{(x^k, t^k)\}$ is a global optimal solution for the problem (EP1).*

Proof. If the proposed algorithm is finite, without losing generality, we assume that the algorithm terminates after k iterations, then we can get the optimal solution (x^k, t^k) by solving the problem (LRP) over T^k, where

$$t_j^k = \sum_{i=1}^{n} c_{ji} x_i^k + d_j, j = 1, 2, \ldots, p.$$

From the convergence detection of the algorithm, the renewal methods of the upper and lower bounds, we can follow that

$$s(t^k) \leq LB_k + \epsilon, LB_k \leq \upsilon, \upsilon \leq s(t^k).$$

Therefore,

$$\upsilon \leq s(t^k) \leq \upsilon + \epsilon,$$

i.e. (x^k, t^k) is a global ϵ-optimal solution of the problem (EP1).

If the proposed algorithm is infinite, then it can generate an infinite solution sequence $\{x^k, t^k\}$ by solving the problem (LRP) over T^k. And obviously that (x^k, t^k) is a feasible solution sequence of the problem (EP1) over T^0. Let (x^*, t^*) be an accumulation point of $\{(x^k, t^k)\}$, without losing generality, we assume that $\lim_{k\to\infty}(x^k, t^k) = (x^*, t^*)$. Let $t_j^k = \sum_{i=1}^{n} c_{ji} x_i^k + d_j, j = 1, 2, \ldots, p$, we can obtain that

$$\lim_{k\to\infty} t_j^k = \lim_{k\to\infty} \sum_{i=1}^{n} c_{ji} x_i^k + d_j = \sum_{i=1}^{n} c_{ji} x_i^* + d_j = t_j^*, j = 1, 2, \ldots, p.$$

From the continuity of the function $\sum_{i=1}^{n} c_{ji} x_i + d_j$, $\sum_{i=1}^{n} c_{ji} x_i^k + d_j = t_j^k \in [l_j^k, u_j^k], j = 1, 2, \ldots, p$, and the exhaustiveness of the branching rule, it follows that

$$\sum_{i=1}^{n} c_{ji} x_i^* + d_j = \lim_{k\to\infty} \sum_{i=1}^{n} c_{ji} x_i^k + d_j = \lim_{k\to\infty} t_j^k = \lim_{k\to\infty} [l_j^k, u_j^k] = \lim_{k\to\infty} \bigcap_k [l_j^k, u_j^k] = t_j^*.$$

Thus, (x^*, t^*) is a feasible solution of the problem (EP1) over T^0. Since $\{LB(T^k)\}$ is an increasing sequence and the updating process of lower bound, we have

$$\lim_{k\to\infty} LB(T^k) \leq s(t^*) \text{ and } \lim_{k\to\infty} LB(T^k) = \lim_{k\to\infty} \phi(t^k).$$

By Theorem 2.2, we can get that

$$\lim_{k\to\infty} \phi(t^k) = \lim_{k\to\infty} s(t^k) = \lim_{k\to\infty} \sum_{j=1}^{p} \gamma_j \ln t_j^k = s(t^*).$$

Therefore, we have

$$\lim_{k\to\infty} LB(T^k) = s(t^*).$$

Thus, (x^*, t^*) is a global optimum solution for the problem (EP1) over T^0. Meanwhile, it follows that x^* is a global optimum solution of the problem (MP), and the proof is completed. □

3.5 Computational complexity of the algorithm

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

$$T = \{T \in \mathbb{R}^p | l_j \le t_j \le u_j, j = 1, 2, \cdots, p\} \subseteq T^0$$

by letting

$$\Delta(T) := \max\{u_j - l_j|, j = 1, 2, \cdots, p\}.$$

In addition, for convenience, we denote by

$$\beta = \max_{j=1,\ldots,p} \frac{|\gamma_j|}{|l_j^0|}. \tag{3.1}$$

where $l_j^0 \in T^0$.

Theorem 3.6. *For any convergence tolerance $\epsilon > 0$, for any $T \subseteq T^0$, if $\Delta(T) \le \frac{\epsilon}{p\beta}$, then for any feasible solution (x, t) of the problem (LRP) over T, we have*

$$|s(t) - \phi(t)| \le \epsilon.$$

Proof. By Theorem 2.2, for any $T \subseteq T^0$, for each $j = 1, 2, \ldots, p$, we have

$$\max_{t\in T}[\delta_j^u(t_j) - \delta_j(t_j)] = \max_{t\in T}[\delta_j(t_j) - \delta_j^l(t_j)].$$

For any $T \subseteq T^0$, if $\Delta(T) \le \frac{\epsilon}{p\beta}$, then for any sufficiently small positive number ϵ, it follows that

$$\begin{aligned}
&|s(t) - \phi(t)| \\
&= |\textstyle\sum_{j=1,\gamma_j>0}^p \gamma_j[\delta_j(t_j) - \delta_j^l(t_j)] - \sum_{j=1,\gamma_j<0}^p \gamma_j[\delta_j^u(t_j) - \delta_j(t_j)]| \\
&\le |\textstyle\sum_{j=1,\gamma_j>0}^p \gamma_j \times \max[\delta_j(t_j) - \delta_j^l(t_j)] - \sum_{j=1,\gamma_j<0}^p \gamma_j \times \max[\delta_j^u(t_j) - \delta_j(t_j)]| \\
&\le \textstyle\sum_{j=1}^p |\gamma_j| \times \max\{|\delta_j(t_j) - \delta_j^l(t_j)|\} \\
&= \textstyle\sum_{j=1}^p |\gamma_j| \times \max\{|\ln t_j - \ln l_j - k_{1j}(t_j - l_j)|, |\ln t_j - \ln \frac{l_j+u_j}{2} - k_{2j}(t_j - \frac{l_j+u_j}{2})|\} \\
&\le \textstyle\sum_{j=1}^p |\gamma_j| \times \max\{|\ln \frac{l_j+u_j}{2} - \ln l_j| + |k_{1j}(\frac{l_j+u_j}{2} - l_j)|, |\ln u_j - \ln \frac{l_j+u_j}{2}| \\
&\quad + |k_{2j}(u_j - \frac{l_j+u_j}{2})|\} \\
&= 2 \times (\textstyle\sum_{j=1}^p |\gamma_j| \times \max\{|\ln \frac{l_j+u_j}{2} - \ln l_j|, |\ln u_j - \ln \frac{l_j+u_j}{2}|\}) \\
&\le 2 \times (\textstyle\sum_{j=1}^p |\gamma_j| \times \max\{\frac{1}{2}|\frac{1}{l_j}(u_j - l_j)|, \frac{1}{2}|\frac{1}{\frac{l_j+u_j}{2}}(u_j - l_j)|\}) \\
&\le 2 \times (\textstyle\sum_{j=1}^p |\gamma_j| \times \max\{\frac{1}{2}|\frac{1}{l_j}(u_j - l_j)|, \frac{1}{2}|\frac{1}{l_j}(u_j - l_j)|\}) \\
&\le \textstyle\sum_{j=1}^p |\gamma_j| \times \max\{|\frac{1}{l_j^0}(u_j - l_j)|, |\frac{1}{l_j^0}(u_j - l_j)|\} \\
&\le \textstyle\sum_{j=1}^p \frac{|\gamma_j|}{|l_j^0|} \times |u_j - l_j| \\
&\le \textstyle\sum_{i=1}^p (\beta\Delta(T)) \\
&= p\beta\Delta(T).
\end{aligned}$$

Therefore, from the above inequalities and $\Delta(T) \le \frac{\epsilon}{p\beta}$, we have that

$$|s(t) - \phi(t)| \le \sum_{i=1}^{p} (\Delta(T)\beta) \le \epsilon,$$

and the proof of the theorem is completed. □

Remark 3.7. From Theorem 3.6, we can follow that T will be removed if $\Delta(T) \leq \frac{\epsilon}{p\beta}$. Therefore, the proposed algorithm terminates if the sizes of all the sub-rectangles T satisfy $\Delta(T) \leq \frac{\epsilon}{p\beta}$.

Theorem 3.8. *Given the convergence tolerance $\epsilon > 0$, the maximum number of iterations required by the proposed algorithm in order to obtain the optimal solution to the problem (MP) is*

$$M = 2^{\sum_{j=1}^{p} \lceil \log_2 \frac{p\beta(u_j^0 - l_j^0)}{\epsilon} \rceil} - 1,$$

where β is given by (3.1), and $T^0 = \prod_{j=1}^{p} T_j^0$ with $T_j^0 = [l_j^0, u_j^0]$.

Proof. We assume that a rectangle $T = \prod_{j=1}^{p} T_j$ with $T_j = [l_j, u_j] \subseteq T_j^0$ is selected from T^0 starting to subdivide in Step 1 of the algorithm for each iteration. Without loss of generality, from Theorem 3.6, suppose that, after m_j iterations, there is a subinterval $T_j^{m_j} = [l_j^{m_j}, u_j^{m_j}]$ of $T_j^0 = [l_j^0, u_j^0]$ satisfying

$$u_j^{m_j} - l_j^{m_j} \leq \frac{\epsilon}{p\beta}, \quad j = 1, 2, \cdots, p. \tag{3.2}$$

According to the branching process, we can get that

$$u_j^{m_j} - l_j^{m_j} = \frac{1}{2^{m_j}}(u_j^0 - l_j^0), \quad j = 1, 2, \cdots, p.$$

Therefore, we have

$$\frac{1}{2^{m_j}}(u_j^0 - l_j^0) \leq \frac{\epsilon}{p\beta}, \quad j = 1, 2, \cdots, p.$$

That is

$$m_j \geq \log_2 \frac{p\beta(u_j^0 - l_j^0)}{\epsilon}, \quad j = 1, 2, \cdots, p.$$

Let

$$\bar{m}_j = \lceil \log_2 \frac{p\beta(u_j^0 - l_j^0)}{\epsilon} \rceil, \quad j = 1, 2, \cdots, p.$$

Then, after $M_1 = \sum_{j=1}^{p} \bar{m}_j$ iterations, the proposed algorithm will generate at most $M_1 + 1$ rectangles, denoted by $T^1, T^2, \ldots, T^{M_1+1}$, and they satisfy that

$$\Delta(T^{M_1}) = \Delta(T^{M_1+1}) = \max\{u_j^{\bar{m}_j} - l_j^{\bar{m}_j}, j = 1, 2, \ldots, p\}$$

and

$$T^0 = \bigcup_{\eta=1}^{M_1+1} T^\eta.$$

Next, put the $M_1 + 1$ sub-rectangles into the set S^{M_1+1}, i.e.

$$S^{M_1+1} = \{T^\eta, \eta = 1, 2, \ldots, M_1 + 1\}.$$

According to (3.2), we have

$$\Delta(T^{M_1}) = \Delta(T^{M_1+1}) \leq \frac{\epsilon}{p\beta}.$$

For the convenience of description, let $\bar{\Delta} = \Delta(T^{M_1}) = \Delta(T^{M_1+1})$, we can obtain that

$$\bar{\Delta} \leq \frac{\epsilon}{p\beta}.$$

From Theorem 3.6 and Step 4, we will discard T^{M_1} and T^{M_1+1} from the set S^{M_1+1}, because there is no optimal solution to the problem (EP) in T^{M_1} and T^{M_1+1}. Next, the remaining sub-rectangles are placed in the set S^{M_1}, where

$$S^{M_1} = S^{M_1+1} \setminus \{T^{M_1}, T^{M_1+1}\} = \{T^\eta, \eta = 1, 2, \ldots, M_1 - 1\}.$$

And the remaining rectangles T^η $(\eta = 1, 2, \ldots, M_1 - 1)$ will continue to be considered.

Next, we consider T^{M_1-1}. According to the branching rule, T^{M_1-1} will be immediately divided into two sub-rectangles $T^{M_1-1,1}$ and $T^{M_1-1,2}$, such that

$$T^{M_1-1} = T^{M_1-1,1} \bigcup T^{M_1-1,2}$$

and

$$\Delta(T^{M_1-1,1}) = \Delta(T^{M_1-1,2}) = \bar{\Delta}.$$

Hence, after $M_1 + (2^1 - 1)$ iterations, T^{M_1-1} will be discarded from the set S^{M_1} by the above equations, and the remaining sub-rectangles will be put into the set S^{M_1-1}, where

$$S^{M_1-1} = S^{M_1+1} \setminus \{T^{M_1-1}, T^{M_1}, T^{M_1+1}\} = \{T^\eta, \eta = 1, 2, \ldots, M_1 - 2\}.$$

Similarly, after the algorithm executed $M_1 + (2^1 - 1) + (2^2 - 1)$ iterations, T^{M_1-2} will be thrown out of the set S^{M_1-1} . And we move the remaining sub-rectangles into the set S^{M_1-2}, where

$$S^{M_1-2} = S^{M_1+1} \setminus \{T^{M_1-2}, T^{M_1-1}, T^{M_1}, T^{M_1+1}\} = \{T^\eta, \eta = 1, 2, \ldots, M_1 - 3\}.$$

Repeat the above process, until all $T^\eta(\eta = 1, 2, \ldots, M_1 + 1)$ are deleted from T^0. Therefore, this algorithm iterates at most

$$M = M_1 + (2^1 - 1) + (2^2 - 1) + \ldots + (2^{M_1-1} - 1) = 2^{M_1} - 1 = 2^{\sum_{j=1}^{p} \lceil \log_2 \frac{p\beta(u_j^0 - l_j^0)}{\epsilon} \rceil} - 1$$

times, and the proof of the theorem is completed. □

4 Numerical Experiments

In this section, we numerically compare the image space branch-and-bound algorithm with BARON [8] and the existing branch-and-bound algorithms (Refs. [12, 26, 23]). All numerical tests are implemented in MATLAB R2018a and run on a microcomputer with Win 7, Intel(R) Core(TM) i5-4590S CPU @3.00GHz processor, and 4 GB RAM. The maximum CPU time limit of all algorithms is set at $3600s$. For all test examples, we present statistics of the numerical results.

First of all, some small-size examples in Appendix were tested with our algorithm for comparison with the known existing algorithms (Refs. [12, 26, 23]) and BARON, and the corresponding numerical results are reported in Table 1 with the given convergence tolerance, where some notations have been used for column headers: Iter.: the number of iterations of the algorithm; Time: the CPU execution time of the algorithm in seconds.

From the numerical results in Table 1, for all small-size examples A.1-A.5, we can follow that our algorithm can obtain almost the same global optimal solutions and optimal values as BARON and the existing algorithms in Refs.[12, 26, 23] with almost the same computational efficiency.

Next, we chose some large-size examples generated randomly to verify our algorithm further, see the following Examples 1-4 for details.

Example 1 (Ref.[25]).

$$\begin{cases} \min & \prod_{j=1}^{2}(\sum_{i=1}^{n} c_{ji}x_i + 1), \\ \text{s.t.} & Ax \le b, x \ge 0. \end{cases}$$

where c_{ji}, $j = 1, 2,\ i = 1, \ldots, n$, is randomly generated in $[0, 1]$, each element of the matrix A is randomly generated in $[-1, 1]$, and b_i, $i = 1, \ldots, n$, is randomly generated by setting $b_i = \sum_{j=1}^{n} a_{ij} + 2\mu$, where μ is randomly generated in $[0, 1]$.

Example 2 (Ref.[28]).

$$\begin{cases} \min & \prod_{j=1}^{p} c_j^{\top} x, \\ \text{s.t.} & \sum_{i=1}^{n} A_{si}x_i \le b_s, s = 1, 2, \ldots, m, \\ & 0 \le x_i \le 1, i = 1, 2, \cdots, n. \end{cases}$$

where c_j, $j = 1, 2, \ldots, p$, is randomly generated in $[0, 1]$, A_{si}, $s = 1, 2, \ldots, m, i = 1, 2, \ldots, n$, is randomly generated in $[-1, 1]$, and b_s, $s = 1, 2, \ldots, m$, is generated by setting $b_s = \sum_{i=1}^{n} A_{si} + 2\mu$, where μ is randomly generated in $[0, 1]$.

Example 3.

$$\begin{cases} \min & \prod_{j=1}^{p}(\sum_{i=1}^{n} c_{ji}x_i + d_j)^{\gamma_j}, \\ \text{s.t.} & Ax \le b, x \ge 0. \end{cases}$$

where c_{ji} and d_j, $j = 1, 2, \ldots, p, i = 1, 2, \ldots, n$, are all randomly generated in $[0, 1]$, each element a_{ij} of the matrix A and $\gamma_j, j = 1, 2, \ldots, p$, are generated in $[-1, 1]$, and each element b_i of the vector b is generated by setting $b_i = \sum_{j=1}^{n} a_{ij} + 2\mu$, where μ is randomly generated in $[0, 1]$.

Example 4.

$$\begin{cases} \min & \prod_{j=1}^{p}(\sum_{i=1}^{n} c_{ji}x_i + n + 1), \\ \text{s.t.} & Ax \le b, x \in [-1, 1]. \end{cases}$$

where c_{ji}, $j = 1, 2,\ i = 1, \ldots, n$, is randomly generated in $[-1, 1]$, each element of the matrix A is randomly generated in $[-1, 1]$, and b_i, $i = 1, \ldots, n$, is randomly generated by setting $b_i = \sum_{j=1}^{n} a_{ij} + 2\mu$, where μ is randomly generated in $[0, 1]$.

Obviously, when $p = 2, n = 2$, $c_{11} = 1, c_{12} = 1, c_{21} = 1, c_{22} = -1$, $A = 0$, and $b = 0$, Example 4 can be reduced to

$$\begin{cases} \min & f(x) = (x_1 + x_2 + 2)(x_1 - x_2 + 3) = x_1^2 - x_2^2 + 5x_1 + x_2 + 6, \\ \text{s.t.} & x \in [-1, 1], \end{cases}$$

which is a nonconvex optimization problem.

The numerical results of Examples 1-4 are listed in Tables 2-7. In Tables 2-7, some notations are given as follows: Avg.N denotes by the average number of iterations; Std.N denotes by the standard deviation of number of iterations; Avg.T denotes by the average execution CPU time of the algorithm in seconds; Std.T denotes by the standard deviation of execution CPU time of the algorithm; and "−" represents that some of ten random examples failed to terminate in $3600s$.

For each random Examples 1-4, with the given convergence error $\epsilon = 10^{-6}$, we solved ten independently generated problems and recorded the average numerical results among these ten test problems. For the large-size Example 2, since the software BARON failed to solve some of ten random examples in $3600s$, we only report the numerical comparisons between the algorithm presented in Zhang et al. [28] and our algorithm in Table 4. For the large-size Examples 3 and 4, since the software BARON also failed to solve some of ten random examples in $3600s$, we only report the numerical results of our algorithm in Tables 6 and 7.

By the numerical results in Table 2, first of all, we can observe that the algorithm presented in Wang and Liu [25] takes more average time with more average iterations than our algorithm, so that our algorithm highly outperforms the algorithm presented in Wang and Liu [25]. Secondly, our algorithm highly outperforms the algorithm presented in Wang and Liu [25] and the software BARON in computational performance.

By the numerical results in Tables 3, 6 and 7, we observe that our algorithm can solve the large-size Examples 2-4, but the software BARON failed to solve some of ten random examples in $3600s$. Thus, this indicates the robustness and stability of our algorithm.

By the numerical results in Table 4, compared with the algorithm presented in Zhang et al.[28], we can see that our algorithm uses fewer iterations and CPU running time. Thus, our algorithm outperforms the algorithm presented in Zhang et al.[28].

By the numerical results in Table 5, for Example 3, when $p \geq 2$, $m \geq 20$, and $n \geq 20$, the software BARON failed to solve some of ten random examples in $3600s$, but our algorithm can obtain the global optimal solution for such a problem. Thus, for Example 3, our algorithm has higher computational performance than the software BARON.

By the numerical results in Tables 1-7, the software BARON is the most efficient one for the small-size problem (MP). Moreover, we can observe that Examples 1-4 with the large-size variables seem to be more difficult to be solved by the algorithm presented in Wang and Liu [25], Zhang et al.[28] and the software BARON. It is expected to become new test examples for validating the global algorithm of a generalized affine multiplicative programming problem.

5 Conclusion

We study a class of multiplicative problems (MP) and propose an image space branch-and-bound algorithm. In this algorithm, by introducing new variables and equivalent transformation, we firstly convert the problem (MP) into the equivalent problem (EP1). Next, we propose a new piecewise linear relaxation method. By using the linear relaxation method, the problem (EP1) is transformed into a linear relaxation problem. Based on the image space branch-and-bound search, the linear relaxation problem, and the image space region reduction techniques, we design an image space branch-and-bound algorithm for globally solving the problem (MP). In contrast to some existing branch-and-bound algorithms, the proposed algorithm can achieve an approximate global ϵ-optimal solution in at most $2^{\sum_{j=1}^{p}\lceil \log_2 \frac{p\beta(u_j^0 - l_j^0)}{\epsilon}\rceil} - 1$ iterations. Numerical results indicate higher efficiency of the algo-

Table 4.1: Numerical comparisons among the software BARON, Algorithms of Refs.[12, 26, 23], and our new algorithm on Examples A.1-A.5.

Example	Refs.	Optimal value	Optimal solution	Iter	CPU time
1	Our	0.89019	(1.3148, 0.1396, 0.0, 0.4233)	1	0.03
	[26]	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)	30	0.75
	[23]	0.53333	(0.0, 0.0)	16	0.05
	BARON	0.5333	(0.0, 0.0)	1	0.03
3	Our	997.66127	(1, 1)	1	0.01
	[12]	997.66127	(1, 1)	3	0.09
	BARON	997.6613	(1, 1)	1	0.03
4	Our	263.78893	(1.25, 1)	1	0.03
	[12]	263.7889	(1.25, 1)	11	0.30
	BARON	263.7889	(1.25, 1)	3	0.03
5	Our	5.00931	(3, 2)	58	1.22
	[12]	5.00931	(3, 2)	2	0.05
	BARON	5.0093	(3, 2)	1	0.03

Table 4.2: Numerical computational comparisons among the software BARON, the algorithm of Wang and Liu [25], and our algorithm on Example 1.

(m, n)	Algorithm of Wang and Liu [25]		Our algorithm		BARON	
	Avg(Std).N	Avg(Std).T	Avg(Std).N	Avg(Std).T	Avg(Std).N	Avg(Std).T
(10,20)	14.2(1.5492)	0.6062(0.0695)	26.0(5.4160)	0.6254(0.1216)	8.8(9.21)	0.1(0.03)
(20,20)	17.4(1.7127)	0.8368(0.0756)	21.3(4.7152)	0.5200(0.1128)	-	-
(22,20)	18.5(1.9003)	0.9460(0.1235)	14.5(9.3956)	0.3606(0.2306)	-	-
(20,30)	19.9(0.5676)	1.0781(0.0674)	24.0(2.8284)	0.5899(0.0867)	-	-
(35,50)	21.2(0.4316)	1.8415(0.1338)	19.8(5.2026)	0.5314(0.1374)	-	-
(45,60)	23.0(0.6667)	2.4338(0.1016)	22.8(2.7809)	0.6563(0.0650)	-	-
(45,100)	35.7(1.1595)	5.1287(0.0935)	23.9(1.3703)	0.8225(0.0573)	-	-
(60,100)	36.1(0.7379)	6.8143(0.1713)	23.7(2.0028)	0.8955(0.0838)	-	-
(70,100)	36.6(1.2649)	8.1967(0.2121)	23.6(0.8433)	0.9462(0.0415)	-	-
(70,120)	39.1(1.6633)	9.5642(0.2975)	24.1(1.6633)	1.0991(0.0660)	-	-
(100,100)	37.5(2.1731)	13.0578(0.3543)	17.8(5.3500)	0.8810(0.2339)	-	-

Table 4.3: Numerical computational results for our algorithm on Example 2.

(m,n)	Avg(Std).N(T)	$p=4$	$p=5$	$p=6$	$p=7$
(10,20)	Avg(Std).N	11.500(3.9229)	13.900(4.7945)	17.200(5.5337)	19.000(6.3596)
	Avg(Std).T	0.3037(0.1045)	0.3604(0.1303)	0.4522(0.1325)	0.4707(0.1551)
(20,40)	Avg(Std).N	11.900(3.1780)	14.500(5.3385)	18.200(3.3928)	19.100(4.8637)
	Avg(Std).T	0.3437(0.0866)	0.3957(0.1458)	0.4993(0.0769)	0.5149(0.1224)
(30,60)	Avg(Std).N	11.600(2.0656)	15.400(2.5033)	18.500(5.3177)	22.900(3.2472)
	Avg(Std).T	0.3423(0.0595)	0.4435(0.0751)	0.5533(0.1512)	0.6640(0.1017)
(40,80)	Avg(Std).N	11.000(2.6667)	15.200(3.6148)	17.300(4.0291)	20.300(4.5717)
	Avg(Std).T	0.3561(0.0648)	0.4771(0.1078)	0.5526(0.1086)	0.6498(0.1447)
(50,100)	Avg(Std).N	11.400(2.0111)	16.400(1.5776)	15.900(3.7845)	18.200(3.5839)
	Avg(Std).T	0.3997(0.0560)	0.5675(0.0542)	0.5712(0.1357)	0.6521(0.1162)
(60,120)	Avg(Std).N	11.900(2.0248)	13.800(2.9740)	17.300(3.8020)	20.900(2.6437)
	Avg(Std).T	0.4682(0.0577)	0.5606(0.1214)	0.6982(0.1103)	0.8813(0.1342)
(70,140)	Avg(Std).N	12.400(2.9889)	14.900(3.6347)	17.900(2.9231)	19.800(5.5337)
	Avg(Std).T	0.5874(0.1375)	0.6740(0.1400)	0.8417(0.1459)	0.9474(0.2458)
(80,160)	Avg(Std).N	11.300(1.5670)	13.600(2.6750)	15.800(2.5298)	20.600(4.0056)
	Avg(Std).T	0.6113(0.0728)	0.7755(0.1293)	0.9197(0.1307)	1.1214(0.1751)
(90,180)	Avg(Std).N	11.900(2.1833)	13.900(2.3310)	15.400(2.7568)	21.000(3.3333)
	Avg(Std).T	0.7373(0.0949)	0.9333(0.1186)	1.0481(0.2041)	1.4688(0.2734)
(100,200)	Avg(Std).N	12.200(3.0840)	13.800(3.9384)	14.300(1.8886)	19.800(4.2374)
	Avg(Std).T	0.9037(0.1969)	1.0944(0.3071)	1.1815(0.2029)	1.6473(0.3446)

Table 4.4: Numerical computational comparisons between the algorithm presented in Zhang et al. [28] and our algorithm on Example 2.

(p,m,n)	Algorithm of Zhang et al. [28]		Our new algorithm	
	Avg.N	Avg.T	Avg.N	Avg.T
(2,10,1000)	15.5	2.6293	10.9	0.6586
(2,10,2000)	28.5	14.0012	11.1	1.5257
(3,10,1000)	101.8	19.3235	14.9	1.1001
(3,10,2000)	185.4	90.3898	18.0	3.3383
(4,10,1000)	757.6	156.5649	23.2	1.6665
(4,10,2000)	1352.1	995.4707	25.9	4.5790

Table 4.5: Numerical computational comparisons between the software BARON and our algorithm on Example 3.

(p,m,n)	Our algorithm		BARON	
	Avg(Std).N	Avg(Std).T	Avg(Std).N	Avg(Std).T
(2,10,20)	16.2(10.7889)	0.3971(0.2567)	3.6(3.8930)	0.0760(0.0455)
(2,20,20)	7.4(8.0166)	0.1890(0.1985)	-	-
(2,22,20)	10.1(7.7524)	0.2688(0.1959)	-	-
(2,20,30)	6.8(9.7388)	0.1868(0.2420)	-	-
(2,35,50)	11.5(10.5541)	0.3154(0.2841)	-	-
(2,45,60)	10.6(10.4158)	0.3190(0.3044)	-	-
(2,45,100)	8.8(6.6131)	0.3130(0.2490)	-	-
(2,60,100)	8.2(7.8287)	0.3286(0.2929)	-	-
(2,70,100)	7.6(9.6747)	0.3247(0.4013)	-	-
(2,70,120)	15.8(10.9423)	0.7730(0.5219)	-	-
(2,100,100)	13.6(8.3160)	0.7146(0.3986)	-	-

Table 4.6: Numerical computational results for our algorithm on Example 3.

(p, m, n)	Avg(Std).N	Avg(Std).T
(10,100,1000)	6.9(11.1798)	19.9324(33.2809)
(10,100,2000)	4.0(3.7118)	35.7294(34.9124)
(10,100,3000)	5.8(9.9978)	113.5503(210.4960)
(10,100,4000)	11.8(19.9321)	372.0691(520.8177)
(10,100,5000)	8.8(11.7927)	642.7141(1030.4993)
(20,100,1000)	87.0(135.0547)	152.6044(241.3521)
(20,100,2000)	35.6(91.1643)	232.1741(601.8485)
(20,100,3000)	14.6(22.2521)	213.4133(325.8330)
(20,100,4000)	5.9(14.8058)	124.8827(307.8063)

Table 4.7: Numerical computational results for our algorithm on Example 4.

(p, m, n)	Avg(Std).NT	Avg(Std).Time
(5,100,100)	109.3(1.4944)	3.7017(0.1228)
(5,100,500)	115.2(0.6325)	32.3005(0.3738)
(5,100,1000)	116.0(0.4714)	104.0153(1.7031)
(5,100,2000)	116.3(0.8233)	351.5042(7.5839)
(5,100,3000)	116.9(0.3162)	737.9607(13.6290)
(5,100,4000)	116.9(0.3162)	1258.1859(23.4237)
(10,100,100)	227.8(3.3267)	8.3797(0.2369)
(10,100,500)	237.8(1.5492)	74.6187(0.9019)
(10,100,1000)	241.0(1.6997)	235.8993(2.2644)
(10,100,2000)	241.7(0.6749)	794.3962(5.6962)
(10,100,3000)	241.8(1.1353)	1655.3296(20.8176)
(10,100,4000)	242.3(0.4830)	2831.4333(44.3357)

rithm. The future work is to extend the algorithm to a generalized convex (or concave) multiplicative programming problem.

Acknowledgements

All authors are very grateful to the responsible editors and the anonymous reviewers for their valuable comments and suggestions, which have greatly improved the earlier version of our paper.

References

[1] H.P. Benson and G.M. Boger, Outcome-space cutting-plane algorithm for linear multiplicative programming, *J. Optimiz. Theory App.* 104 (2000) 301–322.

[2] K.P. Bennett, Global tree optimization: a non-greedy decision tree algorithm, *Comput. Sci. Stat.* 26 (1994) 156–160.

[3] Y. Chen and H. Jiao, A nonisolated optimal solution of general linear multiplicative programming problems, *Comput. Oper. Res.* 36 (2009) 2573–2579.

[4] M.C. Dorneich and N.V. Sahinidis, Global optimization algorithms for chip design and compaction, *Optim. Eng.* 25 (1995) 131–154.

[5] Y.L. Gao, C.X. Xu and Y.T. Yang, Outcome-space branch and bound algorithm for solving linear multiplicative programming, *Comput. Inform. Sci.* 3801 (2005) 675–681.

[6] H. Jiao, S. Liu and Y. Chen, Global optimization algorithm of a generalized linear multiplicative programming, *J. Appl. Math. Comput.* 40 (2012) 551–568.

[7] H. Jiao, W. Wang and R. Chen, An efficient outer space algorithm for generalized linear multiplicative programming problem, *Ieee Access* 8 (2020) 80629–80637.

[8] A. Khajavirad and N.V. Sahinidis, A hybrid LP/NLP paradigm for global optimization relaxations, *Math. Program. Comput.* 10(3) (2018) 383–421.

[9] F. Kahl, S. Agarwal and M.K. Chandraker, Practical global optimization for multiview geometry, *Int. J. Comput. Vision.* 79 (2008) 271–284.

[10] H. Konno and H. Wantanabe, Bond portfolio optimization problems and their applications to index tracking, *J. Oper. Res. Soc. Jpn.* 39 (1996) 295–306.

[11] 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.

[12] S. Liu and Y. Zhao, An efficient algorithm for globally solving generalized linear multiplicative programming, *J. Comput. Appl. Math.* 296 (2016) 840–847.

[13] X. Liu, T. Umegaki and Y. Yamamoto, Heuristic methods for linear multiplicative programming, *J. Global. Optim.* 15 (1999) 433–447.

[14] C.D. Maranas, I.P. Androulakis and C.A. Floudas, Solving long-term financial planning problems via global optimization, *J. Econ. Dyn. Control.* 21 (1997) 1405–1425.

[15] J. Mulvey, R. Vanderbei and S. Zenios, Robust optimization of large-scale systems, *Oper. Res.* 43 (1995) 264–281.

[16] T. Matsui, NP-hardness of linear multiplicative programming and related problem, *J. Global. Optim.* 9 (1996) 113–119.

[17] P. Shen, X. Bai and W. Li, A new accelerating method for globally solving a class of nonconvex programming problems, *Nonlinear Anal.* 71 (2009) 2866–2876.

[18] P. Shen and B. Huang, Global algorithm for solving linear multiplicative programming problems, *Optim. Lett.* 14 (2020) 693–710.

[19] P. Shen and L. Wang, A fully polynomial time approximation algorithm for generalized linear multiplicative programming, *Math. Appl.* 31 (2018) 208–213.

[20] P. Shen, K. Wang and T. Lu, Outer space branch and bound algorithm for solving linear multiplicative programming problems, *J. Global. Optim.* 78 (2020) 453–482.

[21] P. Shen, K. Wang and T. Lu, Global optimization algorithm for solving linear multiplicative programming problems, *Optimization*, (2020), DOI: 10.1080/02331934.2020.1812603.

[22] P. Shen, L. Yang and Y. Liang, Range division and contraction algorithm for a class of global optimization problems, *Appl. Math. Comput.* 242 (2014) 116–126.

[23] N.V. Thoai, A global optimization approach for solving the convex multiplicative programming problems, *J. Global. Optim.* 1 (1991) 341–357.

[24] C. Wang, Y. Bai and P. Shen, A practicable branch-and-bound algorithm for globally solving multiplicative programming, *Optimization* 66 (2017) 397–405.

[25] C. Wang and S. Liu, A new linearization method for generalized linear multiplicative programming, *Comput. Oper. Res.* 38 (2011) 1008–1013.

[26] C. Wang, S. Liu and P. Shen, Global minimization of a generalized linear multiplicative programming, *Appl. Math. Model*, 36 (2012) 2446–2451.

[27] L. Yang, P. Shen and Y. Pei, A global optimization approach for solving generalized nonlinear multiplicative programming problem, *Abstr. Appl. Anal.* (2014), DOI:10.1155/2014/641909.

[28] B. Zhang, Y. Gao and X. Liu, Output-space branch-and-bound reduction algorithm for a class of linear multiplicative programs, *Mathematics*, 8 (2020): 315.

[29] B. Zhang, Y. Gao and X. Liu, An efficient polynomial time algorithm for a class of generalized linear multiplicative programs with positive exponents, *Math. Probl. Eng.* 2021 (2021).

[30] Y. Zhao and T. Zhao, Global optimization for generalized linear multiplicative programming using convex relaxation, *Math. Probl. Eng.* (2018), DOI:10.1155/2018/9146309.

[31] Y. Zhao and S. Liu, An efficient method for generalized linear multiplicative programming problem with multiplicative constraints, *SpringerPlus*, 5 (2016) 1–14.

Appendix

A.1. (Ref.[26]).

$$\begin{cases} \min & (0.813396x_1 + 0.67440x_2 + 0.305038x_3 + 0.129742x_4 + 0.217796) \\ & \times(0.224508x_1 + 0.063458x_2 + 0.932230x_3 + 0.528736x_4 + 0.091947) \\ \text{s.t.} & 0.488509x_1 + 0.063458x_2 + 0.945686x_3 + 0.210704x_4 \le 3.562809, \\ & -0.324014x_1 - 0.501754x_2 - 0.719204x_3 + 0.099562x_4 \le -0.052215, \\ & 0.445225x_1 - 0.346896x_2 + 0.637939x_3 - 0.257623x_4 \le 0.427920, \\ & -0.202821x_1 + 0.647361x_2 + 0.920135x_3 - 0.983091x_4 \le 0.840950, \\ & -0.886420x_1 - 0.802444x_2 - 0.305441x_3 - 0.180123x_4 \le -1.353686, \\ & -0.515399x_1 - 0.424820x_2 + 0.897498x_3 + 0.187268x_4 \le 2.137251, \\ & -0.591515x_1 + 0.060581x_2 - 0.427365x_3 + 0.579388x_4 \le -0.290987, \\ & 0.423524x_1 + 0.940496x_2 - 0.437944x_3 - 0.742941x_4 \le 0.373620, \\ & x_1 \ge 0, x_2 \ge 0, x_3 \ge 0, x_4 \ge 0. \end{cases}$$

A.2. (Ref.[23]).

$$\begin{cases} \min & (-x_1 + 2x_2 + 2)(4x_1 - 3x_2 + 4)(3x_1 - 4x_2 + 5)^{-1}(-2x_1 + x_2 + 3)^{-1} \\ \text{s.t.} & x_1 + x_2 \le 1.5,\ 0 \le x_1 \le 1,\ 0 \le x_2 \le 1. \end{cases}$$

A.3. (Ref.[12]).

$$\begin{cases} \min & (x_1 + x_2 + 1)^{2.5}(2x_1 + x_2 + 1)^{1.1}(x_1 + 2x_2 + 1)^{1.9} \\ \text{s.t.} & x_1 + 2x_2 \le 6,\ \ 2x_1 + 2x_2 \le 8, \\ & 1 \le x_1 \le 3,\ 1 \le x_2 \le 3. \end{cases}$$

A.4. (Ref.[12]).

$$\begin{cases} \min & (3x_1 - 4x_2 + 5)(x_1 + 2x_2 - 1)^{0.5}(2x_1 - x_2 + 4)(x_1 - 2x_2 + 8)^{0.5}(2x_1 + x_2 - 1) \\ \text{s.t.} & 5x_1 - 8x_2 \ge -24,\ \ 5x_1 + 8x_2 \le 44, \\ & 6x_1 - 3x_2 \le 15,\ \ 4x_1 + 5x_2 \ge 10, \\ & 1 \le x_1 \le 3,\ \ 0 \le x_2 \le 1. \end{cases}$$

A.5. (Ref.[12]).

$$\begin{cases} \min & (3x_1 - 2x_2 - 2)^{\frac{2}{3}}(x_1 + 2x_2 + 2)^{\frac{2}{5}} \\ \text{s.t.} & 2x_1 - x_2 \ge 2,\ \ x_1 - 2x_2 \le 2, \\ & x_1 + x_2 \le 5,\ 3 \le x_1 \le 5,\ 1 \le x_2 \le 3. \end{cases}$$

Manuscript received 8 September 2021
revised 20 November 2021
accepted for publication 19 January 2022

Hongwei Jiao
School of Mathematical Sciences
Henan Institute of Science and Technology
Xinxiang 453003, China
E-mail address: jiaohongwei@126.com

Wenjie Wang
School of Mathematical Sciences
Henan Institute of Science and Technology
Xinxiang 453003, China
E-mail address: wangwenjie0780@163.com

Peiping Shen
School of Mathematics and Statistics
North China University of Water Resources and Electric Power
Zhengzhou 450046, China
E-mail address: shenpeiping@163.com