Mathematics

http://www.aimspress.com/journal/Math

AIMS Mathematics, 8(11): 26045–26069.
DOI:10.3934/math.20231327
Received: 29 June 2023
Revised: 25 August 2023
Accepted: 30 August 2023
Published: 08 September 2023

Research article

An efficient outer space branch-and-bound algorithm for globally minimizing linear multiplicative problems

Xiaoli Huang[1,3] **and Yuelin Gao**[2,3,*]

[1] School of Mathematics and Statistics, Ningxia University, Yinchuan 750021, China

[2] Ningxia province cooperative innovation center of scientific computing and intelligent information processing, North Minzu University, Yinchuan 750021, China

[3] Nixngxia mathematics basic discipline research center, Ningxia University, Yinchuan 750021, China

* **Correspondence:** Email: gaoyuelin@263.net.

Abstract: We propose an efficient outer space branch-and-bound algorithm for minimizing linear multiplicative problems (LMP). First, by introducing auxiliary variables, LMP is transformed into an equivalent problem (ELMP), where the number of auxiliary variables is equal to the number of linear functions. Subsequently, based on the properties of exponential and logarithmic functions, further equivalent transformation of ELMP is performed. Next, a novel linear relaxation technique is used to obtain the linear relaxation problem, which provides a reliable lower bound for the global optimal value of LMP. Once more, branching operation takes place in the outer space of the linear function while embedding compression technique to remove infeasible regions to the maximum extent possible, which significantly reduces the computational cost. Therefore, an outer space branch-and-bound algorithm is proposed. In addition, we conduct convergence analysis and complexity proof for the algorithm. Finally, the computational performance of the algorithm is demonstrated based on the experimental results obtained by testing a series of problems.

Keywords: global optimization; linear multiplicative problem; linear relaxation; branch and bound
Mathematics Subject Classification: 90C26, 90C57

1. Introduction

We mainly consider the linear multiplicative problem of the following form:

$$(\text{LMP}):\begin{cases}\min & f(y)=\displaystyle\prod_{i=1}^{p}(c_i^T y+d_i)^{\alpha_i},\\ \text{s.t.} & y\in\mathcal{Y}=\{y\in\mathbb{R}^n|Ay\leq b\},\end{cases}$$

where c_i is an n-dimensional column vector, d_i is a real number and α_i is a real number different from zero. T represents the transpose of a vector, $A\in\mathbb{R}^{m\times n}$, $b\in\mathbb{R}^m$ and $\mathcal{Y}$ is a non-empty bounded set. In this article, for any $y\in\mathcal{Y}$, we suppose that $c_i^T y+d_i>0$, $i=1,...,p$.

LMP is a typical non-convex optimization problem with important applications in real life. It and its variants have been applied in various fields such as robust optimization [1], financial optimization [2], VLSI chip design [3], decision tree optimization [4], network flow optimization [5], supply chain problem [6], investment portfolio [7], etc. Moreover, LMP is an NP-hard problem [8] with multiple local solutions rather than global solutions, which increases the computational complexity. Hence, researching this problem holds great significance.

Various solution algorithms for LMP and its special forms have been proposed by numerous experts and scholars. These algorithms can be broadly categorized into the following groups: branch-and-bound algorithms [9–14], finite algorithm [15], heuristic method [16], approximation algorithm [17], polynomial time algorithm [18], parameterized simplex method [19], cutting-plane method [20], level set algorithm [21], etc. Despite the advancements made by these approaches, solving high-dimensional LMP remains a challenging task. In the past 20 years, searching for global solutions of LMP in the outer space using different relaxation methods has become a hot topic among scholars [22–29]. For example, the authors in references [22, 27] used new convex relaxation techniques to put forward different outer space branch-and-bound algorithms for solving LMP. References [23, 25, 26, 28–30] adopted various linear relaxation programming problems and proposed novel outer space branch-and-bound algorithms, respectively.

In this paper, an outer space branch-and-bound algorithm is designed to solve large-scale LMP. The major characteristics of the algorithm are as follows: First, p auxiliary variables are introduced to transform LMP into an equivalent problem, where p is the number of linear functions. Second, based on the properties of exponential and logarithmic functions, the second equivalent problem (ELMP1) is established. Third, a novel linear relaxation approach is employed to derive the linear relaxation programming problem for ELMP1. Moreover, the branching rule in p-dimensional outer space is given, and the corresponding outer space branch-and-bound algorithm is developed by embedding the rectangular compression technique into the branch-and-bound framework. Finally, the computational complexity of the algorithm is analyzed to estimate the number of iterations in the worst case, which also implies that our algorithm is convergent.

Compared with existing methods [10, 23, 26, 28], the proposed algorithm has the following advantages:

(1) The solved LMP is universal, and the exponents of its objective function are real numbers rather than being limited to just 1 or only positive values.

(2) After the first equivalent conversion, the exponents of the objective function are all positive. Therefore, when linearly approximating the objective function of equivalent problems, there is no need to consider the case where the coefficient is negative, which simplifies the implementation of linear relaxation.

(3) The branching process takes place only in the p-dimensional outer space of the linear function. This leads to cost savings compared to the branching operation in the n-dimensional decision variable space, as p is often much smaller than n in practical problems.

(4) To demonstrate the efficiency of the algorithm, we compared it with the methods in references [10,23,26,28], our algorithm is suitable for solving LMP with large-scale decision variables.

The rest of this paper is organized as follows: Section 2 presents the equivalent problems of LMP and establishes its linear relaxation problem. In Section 3, the branching operation and compression technology are discussed in detail. Moreover, the outer space branch-and-bound algorithm and its properties are described. Section 4 provides numerical comparison results for some problems. Finally, a brief summary of this paper is presented in Section 5.

2. Equivalent problem and its linear relaxation problem

In order to search the global optimal solution of LMP, we transform it into the equivalent problem (ELMP). For convenience, we first define the following sets:

$$I^+ = \{i|\alpha_i > 0, i \in \{1, ..., p\}\}, \quad I^- = \{i|\alpha_i < 0, i \in \{1, ..., p\}\}.$$

For any $i \in \{1, ..., p\}$, denote

$$0 < \underline{t}_i^0 = \min_{y\in\mathcal{Y}} c_i^T y + d_i, \qquad \bar{t}_i^0 = \max_{y\in\mathcal{Y}} c_i^T y + d_i, \qquad i \in I^+;$$
$$0 < \underline{t}_i^0 = \frac{1}{\max_{y\in\mathcal{Y}} c_i^T y + d_i}, \quad \bar{t}_i^0 = \frac{1}{\min_{y\in\mathcal{Y}} c_i^T y + d_i}, \quad i \in I^-.$$

Since $c_i^T y + d_i$ is a bounded linear function, by solving the above $2p$ linear programs, we can easily get that $\underline{t}_i^0$ and $\bar{t}_i^0$. Simultaneously, the initial hyper-rectangle

$$\mathcal{H}^0 = \{t \in \mathbb{R}^p | \underline{t}_i^0 \le t_i \le \bar{t}_i^0, i = 1, ..., p\}$$

is constructed. Thus, let us consider the following equivalent problem:

$$(\text{ELMP}): \begin{cases} \min & \prod\limits_{\alpha_i>0, i\in\{1,...,p\}} t_i^{\alpha_i} \prod\limits_{\alpha_i<0, i\in\{1,...,p\}} t_i^{-\alpha_i} \\ \text{s.t.} & t_i = c_i^T y + d_i, \;\; i \in I^+, \\ & t_i = \dfrac{1}{z_i}, \;\; i \in I^-, \\ & z_i = c_i^T y + d_i, \;\; i \in I^-, \\ & y \in \mathcal{Y}, \;\; t \in \mathcal{H}^0. \end{cases}$$

We denote the feasible region of ELMP as $\mathcal{V} = \{t_i = c_i^T y + d_i, i \in I^+, \; t_i = \frac{1}{z_i}, \; z_i = c_i^T y + d_i, \; i \in I^-, \; y \in \mathcal{Y}, \; t \in \mathcal{H}^0\}$. It is evident that $\mathcal{V}$ is a nonempty bounded compact set if and only if $\mathcal{Y} \neq \emptyset$. Theorem 1 below explains the equivalence between LMP and ELMP.

Theorem 1. (y^*, t^*, z^*) *is a global optimal solution for ELMP if and only if* y^* *is a global optimal solution for LMP, where* $t_i^* = c_i^T y^* + d_i$ *when* $i \in I^+$, $t_i^* = \frac{1}{z_i^*}$ *and* $z_i^* = c_i^T y^* + d_i$ *when* $i \in I^-$.

Proof. We will develop our proof in two aspects. On one hand, for any $y \in \mathcal{Y}$, let $t_i = c_i^T y + d_i$ for $i \in I^+$, $t_i = \frac{1}{z_i}$ and $z_i = c_i^T y + d_i$ for $i \in I^-$, thus $(y, t, z) \in \mathcal{V}$. If y^* is a global optimal solution for LMP, then $t_i^* = c_i^T y^* + d_i$ for $i \in I^+$, $t_i^* = \frac{1}{z_i^*}$ and $z_i^* = c_i^T y^* + d_i$ for $i \in I^-$, so $(y^*, t^*, z^*) \in \mathcal{V}$, which shows that (y^*, t^*, z^*) is a feasible solution to ELMP, and by the optimality of y^*, the following inequalities hold:

$$\begin{aligned}\prod_{i\in I^+}(t_i^*)^{\alpha_i}\prod_{i\in I^-}(t_i^*)^{-\alpha_i} &= \prod_{i\in I^+}(t_i^*)^{\alpha_i}\prod_{i\in I^-}(z_i^*)^{\alpha_i} = \prod_{i=1}^{p}(c_i^T y^* + d_i)\\ &< \prod_{i=1}^{p}(c_i^T y + d_i)^{\alpha_i} = \prod_{i\in I^+} t_i^{\alpha_i}\prod_{i\in I^-} z_i^{\alpha_i}\\ &= \prod_{i\in I^+}(t_i)^{\alpha_i}\prod_{i\in I^-}(t_i)^{-\alpha_i}.\end{aligned}$$

Thus, (y^*, t^*, z^*) is a global optimal solution for ELMP.

On the other hand, if (y^*, t^*, z^*) is a global optimal solution of ELMP, where t^*, z^* are satisfied: if $i \in I^+$, then $t_i^* = c_i^T y^* + d_i$; if $i \in I^-$, then $t_i^* = \frac{1}{z_i^*}$ and $z_i^* = c_i^T y^* + d_i$. Suppose y is a global optimal solution of LMP such that $\prod_{i=1}^{p}(c_i^T y + d_i)^{\alpha_i} < \prod_{i=1}^{p}(c_i^T y^* + d_i)^{\alpha_i}$ holds, for $i \in I^+$, let $t_i = c_i^T y + d_i$, for $i \in I^-$, let $t_i = \frac{1}{z_i}$ and $z_i = c_i^T y + d_i$, it follows that

$$\prod_{i\in I^+} t_i^{\alpha_i}\prod_{i\in I^-} t_i^{-\alpha_i} < \prod_{i\in I^+}(t_i^*)^{\alpha_i}\prod_{i\in I^-}(t_i^*)^{-\alpha_i}.$$

This contradicts the optimality of (y^*, t^*, z^*), thus y^* is a global optimal solution of LMP. □

For the sake of convenience, we denote $\beta_i = -\alpha_i > 0$ for $i \in I^-$. In the meantime, ln(•) and exp(•) represent the logarithmic and the exponential functions, respectively. The objective function of ELMP is further equivalently transformed according to the properties of the exponential and logarithmic functions, namely,

$$\begin{aligned}\prod_{i\in I^+} t_i^{\alpha_i}\prod_{i\in I^-} t_i^{-\alpha_i} &= \prod_{i\in I^+} t_i^{\alpha_i}\prod_{i\in I^-} z_i^{\alpha_i} = \prod_{i\in I^+} t_i^{\alpha_i}\prod_{\beta_i>0, i\in\{1,\dots,p\}} t_i^{\beta_i}\\ &= \exp\left(\ln\left(\prod_{i\in I^+} t_i^{\alpha_i}\prod_{\beta_i>0, i\in\{1,\dots,p\}} t_i^{\beta_i}\right)\right)\\ &= \exp\left(\sum_{i=1, i\in I^+}^{\kappa}\alpha_i \ln t_i + \sum_{i=\kappa+1, \beta_i>0}^{p}\beta_i \ln t_i\right)\\ &= \exp\left(\sum_{i=1}^{p}\zeta_i \ln t_i\right),\end{aligned}$$

where $\zeta \in \mathbb{R}^p$ and $\zeta = [\alpha_1, \alpha_2, \cdots, \alpha_\kappa, \beta_{\kappa+1}, \cdots, \beta_p]$. Hence, ELMP is reformulated to the following form:

$$(\text{ELMP1}): \begin{cases}\min & \mathcal{L}(y, t, z) = \sum_{i=1}^{p}\zeta_i \ln t_i\\ \text{s.t.} & t_i = c_i^T y + d_i, \quad i \in I^+,\\ & z_i = \dfrac{1}{t_i}, \quad z_i = c_i^T y + d_i, \quad i \in I^-,\\ & y \in \mathcal{Y}, \quad t \in \mathcal{H}^k,\end{cases}$$

where $\mathcal{H}^k$ represents $\mathcal{H}^0$ or the sub-rectangle of $\mathcal{H}^0$. Obviously, the optimal solution for ELMP1 is the same as that for ELMP. Therefore, we shift our focus to solving ELMP1, but ELMP1 cannot be solved directly due to the nonlinearity of the objective function and the constraints $z_i = \frac{1}{t_i}$ for $i \in I^-$. To address this, we propose a linear relaxation technique to obtain the lower bound problem of ELMP1.

Theorem 2. *For $i = 1, ..., p$, $t_i \in [\underline{t}_i, \bar{t}_i]$, define*

$$g(t_i) = \ln t_i, \;\; \underline{g}(t_i) = \ln \underline{t}_i + k_i(t_i - \underline{t}_i), \;\; \underline{\mathcal{L}}(y, t, z) = \sum_{i=1}^{p} \zeta_i \underline{g}(t_i),$$

where $k_i = \frac{\ln \bar{t}_i - \ln \underline{t}_i}{\bar{t}_i - \underline{t}_i}$. Then we have the following conclusions:
(i) $\underline{g}(t_i) \le g(t_i)$; (ii) $\underline{\mathcal{L}}(y, t, z) \le \mathcal{L}(y, t, z)$.

Proof. Since the function $g(t_i)$ is a monotonically increasing concave function on $[\underline{t}_i, \bar{t}_i]$ with respect to t_i, $\underline{g}(t_i)$ is its secant approximation, so (i) and (ii) obviously hold. □

Theorem 3. *For each $i \in I^-$, define*

$$\underline{\psi}(t_i) = -\frac{1}{\underline{t}_i \bar{t}_i} t_i + \frac{2}{\sqrt{\underline{t}_i \bar{t}_i}}, \;\; \overline{\psi}(t_i) = -\frac{1}{\underline{t}_i \bar{t}_i} t_i + \frac{1}{\underline{t}_i} + \frac{1}{\bar{t}_i}.$$

Then the functions $\underline{\psi}(t_i)$ and $\overline{\psi}(t_i)$ satisfy the conclusions below:
(i) $\underline{\psi}(t_i) \le \frac{1}{t_i} \le \overline{\psi}(t_i)$;
(ii) Denote $\Delta t_i = \bar{t}_i - \underline{t}_i$, then $\lim_{\Delta t_i \to 0} (\frac{1}{t_i} - \underline{\psi}(t_i)) = 0$, $\lim_{\Delta t_i \to 0} (\overline{\psi}(t_i) - \frac{1}{t_i}) = 0$.

Proof. (i) For each $i \in I^-$, since $t_i \in [\underline{t}_i, \bar{t}_i]$ and $t_i > 0$, it follows from the definitions of $\underline{\psi}(t_i)$ and $\overline{\psi}(t_i)$ that

$$\frac{1}{t_i} - \underline{\psi}(t_i) = \frac{1}{t_i} + \frac{1}{\underline{t}_i \bar{t}_i} t_i - \frac{2}{\sqrt{\underline{t}_i \bar{t}_i}} = \frac{1}{t_i} - \frac{1}{\sqrt{\underline{t}_i \bar{t}_i}} + \frac{1}{\underline{t}_i \bar{t}_i} t_i - \frac{1}{\sqrt{\underline{t}_i \bar{t}_i}} = \frac{\sqrt{\underline{t}_i \bar{t}_i} - t_i}{t_i \sqrt{\underline{t}_i \bar{t}_i}} + \frac{t_i - \sqrt{\underline{t}_i \bar{t}_i}}{\underline{t}_i \bar{t}_i} = \frac{(t_i - \sqrt{\underline{t}_i \bar{t}_i})^2}{t_i \underline{t}_i \bar{t}_i} \ge 0.$$

and

$$\overline{\psi}(t_i) - \frac{1}{t_i} = -\frac{1}{\underline{t}_i \bar{t}_i} t_i + \frac{1}{\underline{t}_i} + \frac{1}{\bar{t}_i} - \frac{1}{t_i} = \frac{-t_i^2 + \underline{t}_i t_i + \bar{t}_i t_i - \underline{t}_i \bar{t}_i}{\underline{t}_i \bar{t}_i t_i} = \frac{(\bar{t}_i - t_i)(t_i - \underline{t}_i)}{\underline{t}_i \bar{t}_i t_i} \ge 0.$$

(ii) From (i), when $\Delta t_i \to 0$ for each $i \in I^-$, the following inequalities are valid:

$$\begin{aligned} \lim_{\Delta t_i \to 0} \left(\frac{1}{t_i} - \underline{\psi}(t_i) \right) &= \lim_{\Delta t_i \to 0} \frac{\sqrt{\underline{t}_i \bar{t}_i} - t_i}{t_i \sqrt{\underline{t}_i \bar{t}_i}} + \frac{t_i - \sqrt{\underline{t}_i \bar{t}_i}}{\underline{t}_i \bar{t}_i} \\ &\le \lim_{\Delta t_i \to 0} \frac{|\bar{t}_i - \underline{t}_i|}{|t_i \sqrt{\underline{t}_i \bar{t}_i}|} + \frac{|\bar{t}_i - \underline{t}_i|}{|\underline{t}_i \bar{t}_i|} \end{aligned}$$

$$
\begin{aligned}
&= \lim_{\Delta t_i \to 0} \left(\frac{1}{|t_i \sqrt{\underline{t}_i \bar{t}_i}|} + \frac{1}{|\underline{t}_i \bar{t}_i|} \right) \Delta t_i \\
&\le \lim_{\Delta t_i \to 0} \frac{2\Delta t_i}{\min\{\underline{t}_i^2, \underline{t}_i \bar{t}_i\}} = 0.
\end{aligned}
$$

and

$$
\lim_{\Delta t_i \to 0} \left(\overline{\psi}(t_i) - \frac{1}{t_i} \right) = \lim_{\Delta t_i \to 0} \frac{(\bar{t}_i - t_i)(t_i - \underline{t}_i)}{\underline{t}_i \bar{t}_i t_i} \le \lim_{\Delta t_i \to 0} \frac{\Delta t_i^2}{\min\{\underline{t}_i^2 \bar{t}_i, \bar{t}_i^2 \underline{t}_i\}} = 0.
$$

□

Consequently, we obtain the linear relaxation program of ELMP1:

$$
(\text{LRP}) : \begin{cases}
\min & \underline{\mathcal{L}}(y, t, z) \\
\text{s.t.} & t_i = c_i^T y + d_i, \quad i \in I^+, \\
& \underline{\psi}(t_i) \le z_i \le \overline{\psi}(t_i), \quad z_i = c_i^T y + d_i, \quad i \in I^-, \\
& y \in \mathcal{Y}, \quad t \in \mathcal{H}^k.
\end{cases}
$$

In the constraint of LRP, we substitute $z_i = c_i^T y + d_i$ into the inequalities $z_i \le \overline{\psi}(t_i)$ and $\underline{\psi}(t_i) \le z_i$, respectively, that is

$$
c_i^T y + \frac{1}{\underline{t}_i \bar{t}_i} t_i \le \frac{1}{\underline{t}_i} + \frac{1}{\bar{t}_i} - d_i, \quad -c_i^T y - \frac{1}{\underline{t}_i \bar{t}_i} t_i \le d_i - \frac{2}{\sqrt{\underline{t}_i \bar{t}_i}}, \quad i \in I^-.
$$

For each $i = 1, ..., p$, $\zeta_i > 0$, LRP is reformulated as

$$
(\text{LRP}(\mathcal{H}^k)) : \begin{cases}
\min & \sum_{i=1}^{p} \zeta_i (\ln \underline{t}_i + k_i (t_i - \underline{t}_i)) \\
\text{s.t.} & -c_i^T y + t_i = d_i, \quad i \in I^+, \\
& c_i^T y + \frac{1}{\underline{t}_i \bar{t}_i} t_i \le \frac{1}{\underline{t}_i} + \frac{1}{\bar{t}_i} - d_i, \quad i \in I^-, \\
& -c_i^T y - \frac{1}{\underline{t}_i \bar{t}_i} t_i \le d_i - \frac{2}{\sqrt{\underline{t}_i \bar{t}_i}}, \quad i \in I^-, \\
& y \in \mathcal{Y}, \quad t \in \mathcal{H}^k.
\end{cases}
$$

We have derived the linear relaxation problem for ELMP1 through a series of relaxation processes. This relaxation enables us to simplify the problem by loosening certain constraints while providing a reliable lower bound for the global optimal value of ELMP1, facilitating informed decision-making in subsequent optimization steps.

3. Branch-and-bound algorithm and its property analysis

In this section, we present an efficient deterministic algorithm for solving ELMP1 by combining the linear relaxation problem proposed in Section 2 with subsequent branching operation in Section 3.1 and rectangle compression technique in Section 3.2. The algorithm requires solving a series of linear relaxation problems on the subdivided rectangles of $\mathcal{H}^0$. Furthermore, we provide rigorous proofs for the convergence and complexity of the algorithm based on the employed branching operation.

3.1. Branching operation

For any selected sub-rectangle $\mathcal{H}^k = \{t \in \mathbb{R}^p | \underline{t}_i \leq t_i \leq \bar{t}_i\} \subseteq \mathcal{H}^0$, the following branching rules are given:

(i) Let $\tau = \operatorname{argmax}\{\bar{t}_i - \underline{t}_i, i = 1, ..., p\}$;

(ii) $\mathcal{H}^k$ is divided into two sub-rectangles

$$\mathcal{H}^{k1} = \Pi_{i=1}^{\tau-1}\mathcal{H}_i \times \left[\underline{t}_\tau, \frac{\underline{t}_\tau + \bar{t}_\tau}{2}\right] \times \Pi_{i=\tau+1}^{p}\mathcal{H}_i,$$

$$\mathcal{H}^{k2} = \Pi_{i=1}^{\tau-1}\mathcal{H}_i \times \left[\frac{\underline{t}_\tau + \bar{t}_\tau}{2}, \bar{t}_\tau\right] \times \Pi_{i=\tau+1}^{p}\mathcal{H}_i,$$

where $\mathcal{H}_i = [t_i \in \mathbb{R} | \underline{t}_i \leq t_i \leq \bar{t}_i, i = 1, ..., p, i \neq \tau]$.

3.2. Compression technology

We introduce a compression technique to enhance the convergence speed of the algorithm. When the algorithm iterates to the kth time, multiple sub-rectangles are obtained through rectangular subdivision. For any sub-rectangle $\widetilde{\mathcal{H}}^k \subseteq \mathcal{H}^k$, we further investigate whether ELMP1 over $\widetilde{\mathcal{H}}^k$ has a global minimum, where $\widetilde{\mathcal{H}}^k = \widetilde{\mathcal{H}}_1 \times \widetilde{\mathcal{H}}_2 \times \cdots \times \widetilde{\mathcal{H}}_p$ and $\widetilde{\mathcal{H}}_i = \{t_i \in \mathbb{R} | \underline{t}_i \leq t_i \leq \bar{t}_i, i = 1, ..., p\}$. The embedded compression technology in the algorithm involves replacing sub-rectangles with smaller rectangles $\widetilde{\mathcal{H}}^k$, while ensuring that the global optimal solution of ELMP1 remains intact and unaffected.

Theorem 4. *When the algorithm iterates to the kth time, let $\widehat{UB}$ be the current best upper bound of the global optimum of ELMP1. Denote*

$$\Xi = \sum_{i=1}^{p} \zeta_i \ln \underline{t}_i, \quad \pi_\iota = \exp\left(\frac{\widehat{UB} - \Xi + \zeta_\iota \ln \underline{t}_\iota}{\zeta_\iota}\right), \quad \iota \in \{1, 2, \cdots, p\}.$$

For any sub-rectangle $\widetilde{\mathcal{H}}^k \subseteq \mathcal{H}^k$, it can be inferred that

(i) If $\Xi > \widehat{UB}$, then there is no global optimal solution over $\widetilde{\mathcal{H}}^k$ for ELMP1;

(ii) If $\Xi < \widehat{UB}$, then ELMP1 has no global optimal solution over $\ddot{\mathcal{H}}$, where

$$\ddot{\mathcal{H}} = \ddot{\mathcal{H}}_1 \times \cdots \times \ddot{\mathcal{H}}_{\iota-1} \times \ddot{\mathcal{H}}_\iota \times \ddot{\mathcal{H}}_{\iota+1} \times \cdots \times \ddot{\mathcal{H}}_p$$

with $\ddot{\mathcal{H}}_\iota = \{t_\iota \in \mathbb{R} | \pi_\iota \leq t_\iota \leq \bar{t}_\iota\} \cap \widetilde{\mathcal{H}}_\iota$.

Proof. (i) If $\Xi > \widehat{UB}$, then

$$\min \sum_{i=1}^{p} \zeta_i \ln t_i = \sum_{i=1}^{p} \zeta_i \ln \underline{t}_i = \Xi > \widehat{UB}.$$

Therefore, $\widetilde{\mathcal{H}}^k$ does not contain a global optimal solution for ELMP1.

(ii) If $\Xi < \widehat{UB}$, for any $t \in \ddot{\mathcal{H}}$,

$$\begin{aligned}
\min_{t\in\ddot{\mathcal{H}}} \sum_{i=1}^{p} \zeta_i \ln t_i &= \min_{t\in\ddot{\mathcal{H}}} \sum_{i=1, i\neq\iota}^{p} \zeta_i \ln t_i + \min_{t\in\ddot{\mathcal{H}}} \zeta_\iota \ln t_\iota \\
&> \sum_{i=1, i\neq\iota}^{p} \zeta_i \ln \underline{t}_i + \zeta_\iota \ln \pi_\iota \\
&= \sum_{i=1, i\neq\iota}^{p} \zeta_i \ln \underline{t}_i + \zeta_\iota \ln \left(\exp \left(\frac{\widehat{UB} - \Xi + \zeta_\iota \ln \underline{t}_\iota}{\zeta_\iota} \right) \right) \\
&= \sum_{i=1, i\neq\iota}^{p} \zeta_i \ln \underline{t}_i + \zeta_\iota \ln \underline{t}_\iota + \widehat{UB} - \Xi \\
&= \widehat{UB}.
\end{aligned}$$

Therefore, $\ddot{\mathcal{H}}$ does not contain a global optimal solution for ELMP1. □

3.3. Branch-and-bound algorithm

The branching operation proposed in Section 3.1 partitions the initial rectangle $\mathcal{H}^0$ into smaller sub-rectangles, enabling the algorithm to search for local optimal solutions of ELMP1 over $\mathcal{V}$ that necessarily include the global minimal solution of ELMP1. During the kth iteration of the algorithm, we provide some relevant notations. Q_k denotes the list of active nodes. For each branching node $\mathcal{H} \in Q_k$, $(y(\mathcal{H}), t(\mathcal{H}))$ and $LB(\mathcal{H})$ represent the optimal solution and the optimal value of LRP($\mathcal{H}$), respectively. The current best lower bound for ELMP1 is noted as $LB_k = \min\{LB(\mathcal{H}), \mathcal{H} \in Q_k\}$. v_k represents the objective function value corresponding to the best feasible solution of ELMP1, and the current best upper bound of v_k is denoted as UB. We choose a divided rectangle $\mathcal{H}^k$ from Q_k that satisfies $LB(\mathcal{H}) = LB_k$, and segment it into two sub-rectangles $\mathcal{H}^{k1}$ and $\mathcal{H}^{k2}$ by branching operation. These sub-rectangles are then added to the set $\mathcal{T}$, and the set $\mathcal{T}$ is updated as $\mathcal{T} = \{\mathcal{T} \backslash \mathcal{H}^k\} \bigcup \{\mathcal{H}^{k1}, \mathcal{H}^{k2}\}$. Let $\mathcal{F}$ be the set of feasible points, and ϵ denotes algorithmic tolerance. In a more precise manner, we can describe the presented outer space branch-and-bound algorithm as follows:

Step 0. Initialize the best known solution as null and the best known upper bound as infinity. Create a root node and initialize the list of active nodes with this root node. Set the algorithmic tolerance to the desired value.

$$\mathcal{F} = \emptyset, \quad UB = +\infty, \quad Q_0 = \{\mathcal{H}^0\}, \quad \epsilon \geq 0, \quad k = 0.$$

Step 1. Solve a relaxation problem LRP($\mathcal{H}^0$) in order to get a lower bound (or prove infeasibility). If problem is feasible, update the incumbent if necessary. Let

$$LB_0 = LB(\mathcal{H}^0), \quad (y^0, t^0, z^0) = (y(\mathcal{H}^0), t(\mathcal{H}^0), z(\mathcal{H}^0)).$$

If (y^0, t^0, z^0) is a feasible solution of ELMP1, then let $UB = \mathcal{L}(y^0, t^0, z^0)$, $\mathcal{F} = \mathcal{F} \bigcup \{y^0, t^0, z^0\}$. If $UB - LB_0 \leq \epsilon$, the algorithm terminates and obtains the global ϵ-optimal solution (y^0, t^0, z^0) for ELMP1. Otherwise, denote $\mathcal{T} = \{\mathcal{H}^0\}$.

Step 2. Split the current node $\mathcal{H}^k$ into two new nodes $\mathcal{H}^{kj}(j = 1, 2)$ and reduce them by using the compression technique, the reduced rectangle is still denoted as $\mathcal{H}^{kj}(j = 1, 2)$ and set $\mathcal{T} = \{\mathcal{T} \backslash \mathcal{H}^k\} \bigcup \{\mathcal{H}^{k1}, \mathcal{H}^{k2}\}$.

Step 3. For each child node $\mathcal{H}^{kj} \in \mathcal{T}(j = 1, 2)$, the corresponding optimal value $LB(\mathcal{H}^{kj})$ and optimal solution $(y(\mathcal{H}^{kj}), t(\mathcal{H}^{kj}))$ are obtained by solving LRP$(\mathcal{H}^{kj})$. Set $\mathcal{F} = \mathcal{F} \bigcup (y(\mathcal{H}^{kj}), t(\mathcal{H}^{kj}), z(\mathcal{H}^{kj}))$, $(\hat{y}^k, \hat{t}^k, \hat{z}^k) = \text{argmin}_{(y,t,z)\in\mathcal{F}} \mathcal{L}(y, t, z)$, set $v_k = \mathcal{L}(\hat{y}^k, \hat{t}^k, \hat{z}^k)$. If $v_k < UB$, then update the upper bound $UB = v_k$, the current best solution for ELMP1 is updated as $(y^k, t^k, z^k) = (\hat{y}^k, \hat{t}^k, \hat{z}^k)$, and set $\mathcal{F} = \emptyset$. If $LB(\mathcal{H}^{kj}) > UB$, then remove it from $\mathcal{T}$, i.e. $\mathcal{T} = \mathcal{T} \backslash \{\mathcal{H}^{kj}\}$. Set $Q_k = (Q_k \backslash \mathcal{H}^k) \bigcup \mathcal{T}$ and update the lower bound $LB_k = \min\{LB(\mathcal{H}) | \mathcal{H} \in Q_k\}$.

Step 4. Let the list of active nodes $Q_{k+1} = \{\mathcal{H} | UB - LB(\mathcal{H}) > \epsilon, \mathcal{H} \in Q_k\}$. If Q_{k+1} is empty, return the best known solution as a global ϵ-optimal solution. Otherwise select an active node $\mathcal{H}^{k+1} \in \text{argmin}\{LB(\mathcal{H}), \mathcal{H} \in Q_{k+1}\}$. Set $k = k + 1$ and go back to Step 2.

Definition 1 provides the concept of the global ϵ-optimal solution involved in the proposed algorithm.

Definition 1. *Given $\epsilon > 0$, the feasible solution $(\hat{y}, \hat{t}, \hat{z})$ is considered a global ϵ-optimal solution for ELMP1, if $\mathcal{L}(\hat{y}, \hat{t}, \hat{z}) \leq v + \varepsilon$, where v represents the global optimal value of ELMP1.*

3.4. Convergence analysis

This subsection discusses the convergence analysis of the algorithm. Supposing the algorithm is infinite, according to the branching operation, there exists an infinite rectangular sequence $\{\mathcal{H}^k\}_{k=1}^{\infty}$ such that for each $k = 1, 2, ...,$ we have $\mathcal{H}^{k+1} \subseteq \mathcal{H}^k \subseteq \cdots \subseteq \mathcal{H}^0$, where $\mathcal{H}^k \in \mathbb{R}^p$. The following Lemma 1 provides a theoretical basis for Theorem 5.

Lemma 1. *For any $t \in \mathcal{H}$, when $\bar{t}_i - \underline{t}_i \to 0$, $i = 1, ..., p$, for which we have $\mathcal{L}(y, t, z) - \underline{\mathcal{L}}(y, t, z) \to 0$.*

Proof. It follows from Theorem 2 that

$$\begin{aligned}
\mathcal{L}(y, t, z) - \underline{\mathcal{L}}(y, t, z) &= \sum_{i=1}^{p} \zeta_i[\ln t_i - (\ln \underline{t}_i + k_i(t_i - \underline{t}_i)] \\
&\leq \max\{\sum_{i=1}^{p} \zeta_i[\ln t_i - (\ln \underline{t}_i + k_i(t_i - \underline{t}_i)]\} \\
&\leq \sum_{i=1}^{p} \max\{\zeta_i\} \max_{t_i \in \mathcal{H}_i}\{\ln t_i - (\ln \underline{t}_i + k_i(t_i - \underline{t}_i))\} \\
&= \sum_{i=1}^{p} \max\{\zeta_i\}[\ln \bar{t}_i - \ln \underline{t}_i + \frac{\ln \bar{t}_i - \ln \underline{t}_i}{\bar{t}_i - \underline{t}_i}(\bar{t}_i - \underline{t}_i)] \\
&= \sum_{i=1}^{p} \max\{\zeta_i\} \cdot 2(\ln \bar{t}_i - \ln \underline{t}_i) \\
&\leq \sum_{i=1}^{p} \frac{2 \max\{\zeta_i\}}{\underline{t}_i}(\bar{t}_i - \underline{t}_i).
\end{aligned}$$

Therefore, for any $t \in \mathcal{H}$, $\mathcal{L}(y,t,z) - \underline{\mathcal{L}}(y,t,z) \to 0$ holds while $\bar{t}_i - \underline{t}_i \to 0$, $i = 1, ..., p$. □

Theorem 5. *Given $\epsilon > 0$, assuming that the feasible domain of ELMP1 is non-empty, the algorithm either obtains a global ϵ-optimal solution of ELMP1 at a finite number of iterations, or produces a sequence of feasible solutions $\{(y^k, t^k, z^k)\}$, each of whose accumulation points is a global ϵ-optimal solution of ELMP1.*

Proof. Assuming that the algorithm terminates within a finite number of iterations, without loss of generality, let us assume that the algorithm terminates at the kth iteration, which gets $UB - LB_k \le \epsilon$. According to Step 3 in the algorithm, we have $UB = v_k = \mathcal{L}(\hat{y}^k, \hat{t}^k, \hat{z}^k)$, thus

$$\mathcal{L}(\hat{y}^k, \hat{t}^k, \hat{z}^k) - LB_k \le \epsilon. \tag{3.1}$$

It follows from (3.1) that

$$\mathcal{L}(\hat{y}^k, \hat{t}^k, \hat{z}^k) \le LB_k + \epsilon \le v + \epsilon.$$

Thereby, $(\hat{y}^k, \hat{t}^k, \hat{z}^k)$ is a global ϵ-optimal solution to ELMP1.

If the algorithm iterates an infinite number of times and produces an infinite sequence of rectangles $\{\mathcal{H}^k\}_{k=1}^{\infty}$, where $\mathcal{H}^k = \prod_{i=1}^{p}[\underline{t}_i^k, \bar{t}_i^k] \in \mathbb{R}^p$. Without losing generality, suppose that $\lim\limits_{k\to\infty} \mathcal{H}^k = H^{\infty}$, then for the subsequences K of sequence $\{1, 2, ...\}$ we have

$$\lim_{k\in K} \mathcal{H}^k = H^{\infty}. \tag{3.2}$$

For any $k \in K$, depending on Step 3 of the algorithm, the lower bound is updated to

$$LB(\mathcal{H}^k) = \min_{t\in\mathcal{V}} \underline{\mathcal{L}}(y^k, t^k, z^k) \le v_k \le \mathcal{L}(\hat{y}^k, \hat{t}^k, \hat{z}^k) \le \mathcal{L}(y^k, t^k, z^k).$$

For any $k \in K$, it follows that $t^k \in \mathcal{H}^k \subseteq \mathcal{H}$. Therefore, $\{t^k\}_{k=1}^{\infty}$ exists a convergent subsequence $\{t^k\}_{k\in K}$, by formula (3.2), the limit of $\{t^k\}_{k\in K}$ is in $\mathcal{H}^{\infty}$, let

$$\lim_{k\in K} t^k = \hat{t}, \tag{3.3}$$

where $\hat{t}$ is a accumulation point of $\{t^k\}_{k\in K}$. Since $\mathcal{L}(y,t,z)$ is continuous, combining with (3.3) we have

$$\lim_{k\in K} \mathcal{L}(y^k, t^k, z^k) = \mathcal{L}(\hat{y}, \hat{t}, \hat{z}). \tag{3.4}$$

For any $k \in K$, $t^k \in \mathcal{H}^k$, it follows from Lemma 1 that

$$\lim_{k\in K} (\mathcal{L}(y^k, t^k, z^k) - \underline{\mathcal{L}}(y^k, t^k, z^k)) = 0. \tag{3.5}$$

Hence, we have

$$\lim_{k\in K} \underline{\mathcal{L}}(y^k, t^k, z^k) = \mathcal{L}(\hat{y}, \hat{t}, \hat{z}). \tag{3.6}$$

Integrating (3.4)–(3.6), we obtain $\mathcal{L}(\hat{y}, \hat{t}, \hat{z}) = \lim_{k\in K} \mathcal{L}(y^k, t^k, z^k) = \lim_{k\in K} \underline{\mathcal{L}}(y^k, t^k, z^k)$. For each $k \in K$ we get $\mathcal{L}(\hat{y}^k, \hat{t}^k, \hat{z}^k) = \sum_{i=1}^{p} \zeta_i \ln(t_i^k)$, therefore

$$\lim_{k\to\infty} v^k = \lim_{k\to\infty} \sum_{i=1}^{p} \zeta_i \ln(t_i^k) = v. \tag{3.7}$$

Because $\{\sum_{i=1}^{p}\zeta_i \ln(t_i^k)\}_{k\in K}$ is a subsequence of $\{\sum_{i=1}^{p}\zeta_i \ln(t_i^k)\}_{k=1}^{\infty}$, following from (3.7), we then obtain $\lim_{k\in K}\sum_{i=1}^{p}\zeta_i \ln(t_i^k) = v$. From the continuity of $\mathcal{L}(y,t,z)$ and formula (3.5), we have

$$\lim_{k\in K}\sum_{i=1}^{p}\zeta_i \ln(t_i^k) = \sum_{i=1}^{p}\zeta_i \ln(\hat{t}_i^k).$$

So we get

$$\sum_{i=1}^{p}\zeta_i \ln(\hat{t}_i^k) = v. \tag{3.8}$$

Combining Eq (3.8), we conclude that $(\hat{y}^k, \hat{t}^k, \hat{z}^k)$ is a globally optimal solution to ELMP1. □

3.5. Complexity analysis

We use $\Omega(\mathcal{H})$ to define the size of the sub-rectangle $\mathcal{H} = \{t \in \mathbb{R}^p, \underline{t}_i \le t_i \le \bar{t}_i, i = 1, ..., p\}$, i.e.,

$$\Omega(\mathcal{H}) = \max\{\bar{t}_i - \underline{t}_i, i = 1, ..., p\}.$$

In addition,

$$\theta = 2\max\{\zeta_i, i = 1, ..., p\},\ \ \lambda = \max\left\{\frac{1}{\underline{t}_i}, i = 1, ..., p\right\}.$$

Lemma 2. *Given any $\epsilon > 0$, for a sub-rectangle $\mathcal{H} \subseteq \mathcal{H}^0$, if $\Omega(\mathcal{H}) < \epsilon$, then for any optimal solution (y,t) of LRP($\mathcal{H}$), we have*

$$|\mathcal{L}(y,t,z) - \underline{\mathcal{L}}(y,t,z)| \le p\delta\theta\lambda\epsilon.$$

Proof. Clearly, (y,t,z) is a feasible solution to ELMP1, it follows from Lemma 1 that

$$\mathcal{L}(y,t,z) - \underline{\mathcal{L}}(y,t,z) \le \sum_{i=1}^{p}\frac{2\max\{\zeta_i\}}{\underline{t}_i}(\bar{t}_i - \underline{t}_i) \le p\theta\lambda\Omega(\mathcal{H}) \le p\theta\lambda\epsilon.$$

□

The following Theorem 6 illustrates an estimation of the maximum number of iterations for the proposed algorithm in the worst case, indirectly indicating that the algorithm will eventually find the global optimal solution for LMP.

Theorem 6. *For any $\epsilon_0 \in (0,1)$, the algorithm iterates at most*

$$2^{\sum_{i=1}^{p}\left\lceil \log_2 \frac{p\theta\lambda\Omega(\mathcal{H}^0)}{\epsilon_0}\right\rceil} - 1$$

times to obtain a global ϵ_0-optimal solution in a worst case.

Proof. Let $\epsilon = \epsilon_0$, suppose that during the kth iteration, when the algorithm reaches Step 3, $\mathcal{H} \subseteq \mathcal{H}^0$ is a rectangle selected by the branching rule to be dissected, which satisfies

$$\bar{t}_i - \underline{t}_i \le \Omega(\mathcal{H}) \le \frac{\epsilon_0}{p\theta\lambda},\ \ i = 1, \cdots, p,$$

then, the algorithm is terminated. In the worst case, when the algorithm terminates at Step 4 on the kth iteration, splitting the initial rectangle $\mathcal{H}^0$ results in $k+1$ sub-rectangles, with the assumption that $\mathcal{H}^\iota$ is any one of these sub-rectangles and satisfies

$$\bar{t}_i^\iota - \underline{t}_i^\iota \le \Omega(\mathcal{H}^\iota) \le \frac{1}{2^{\iota_i}}\Omega(\mathcal{H}^0),\ \ i = 1, \cdots, p,$$

here, ι_i denotes the initial interval $[\underline{t}_i^0, \bar{t}_i^0]$ after ι_i subdivision to produce $[\underline{t}_i^\iota, \bar{t}_i^\iota]$. From Lemma 2, it follows that

$$\frac{1}{2^{\iota_i}}\Omega(\mathcal{H}^0) \le \frac{\epsilon_0}{p\theta\lambda},\ \ i = 1, \cdots, p. \tag{3.9}$$

Since the subdivision $\mathcal{H}^0$ yields no more than $\prod_{i=1}^p 2^{\iota_i}$ sub-rectangles, if every sub-rectangle satisfies (3.9), the algorithm must terminate. By Eq (3.9) we have

$$\iota_i \ge \log_2 \frac{p\theta\lambda\Omega(\mathcal{H}^0)}{\epsilon_0},\ \ i = 1, ..., p.$$

Let $\chi_i = \lceil \log_2 \frac{p\theta\lambda\Omega(\mathcal{H}^0)}{\epsilon_0} \rceil, i = 1, ..., p$. The initial rectangle is split into $k+1$ sub-rectangles and $k+1 = \prod_{i=1}^p 2^{\chi_i}$. At this point the algorithm terminates. Thus, the algorithm iterates at most $2^{\sum_{i=1}^p \lceil \log_2 \frac{p\theta\lambda\Omega(\mathcal{H}^0)}{\epsilon_0} \rceil} - 1$ times. Furthermore, it can be derived that

$$\epsilon_0 \ge |\mathcal{L}(y,t,z) - \underline{\mathcal{L}}(y,t,z)| \ge \mathcal{L}(y,t,z) - LB(\mathcal{H}) \ge \mathcal{L}(y,t,z) - \mathcal{L}(y^*,t^*,z^*) \ge 0, \tag{3.10}$$

where (y^*, t^*, z^*) is a global optimal solution of ELMP1. For $i \in I^+$, $t_i^* = c_i^T y^* + d_i$. For $i \in I^-$, $z_i^* = \frac{1}{t_i^*}$ and $z_i^* = c_i^T y^* + d_i$. Based on the bounding process, let $\hat{y}^k$ be the best feasible solution obtained so far, and denote that $\{f(\hat{y}^k)\}$ is the decreasing sequence such that $f(\hat{y}^k) \le f(y)$. Combined with (3.10) can be obtained

$$\epsilon_0 \ge f(y) - f(y^*) \ge f(\hat{y}^k) - f(y^*).$$

Therefore, the algorithm terminates, and $\hat{y}^k$ is a global ϵ_0-optimal solution to LMP. □

4. Numerical experiments

In this section, several problems are employed to demonstrate the feasibility and effectiveness of the proposed algorithm in this paper. All linear programming problems are solved by the dual-simplex method, all tests of the algorithm are carried out using MATLAB9.2.0.538062 (R2017a) on an Inter(R)Core(TM) i5-8250U, CPU@1.60GHz, 4GB memory and 64 bit Windows10 operating system.

Initially, we utilize the existing branch and bound algorithms [10, 23, 26, 28] and the proposed algorithm to compute the deterministic Problems 1–8 with a predefined convergence tolerance. This is to demonstrate the feasibility of our algorithm. To validate the efficiency of the proposed algorithm, we conduct tests on random Problems 9–11 with a tolerance of 10^{-6}.

Problem 1 [10,26,28]

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

Problem 2 [10, 26, 28]

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

Problem 3 [10, 26, 28]

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

Problem 4 [26, 28]

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

Problem 5 [10, 26, 28]

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

Problem 6 [26, 28]

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

Problem 7 [23, 26, 28]

$$\begin{cases} \min & (-4y_1 - 4y_4 + 3y_5 + 21)(4y_1 + 2y_2 + 3y_3 - 4y_4 + 4y_5 - 3) \\ & \times (3y_1 + 4y_2 + 2y_3 - 2y_4 + 2y_5 - 7)(-2y_1 + y_2 - 2y_3 + 2y_5 + 11) \\ \text{s.t.} & 4y_1 + 4y_2 + 5y_3 + 3y_4 + y_5 \leq 25,\ \ -y_1 - 5y_2 + 2y_3 + 3y_4 + y_5 \leq 2, \\ & y_1 + 2y_2 + y_3 - 2y_4 + 2y_5 \geq 6,\ \ 4y_2 + 3y_3 - 8y_4 + 11y_5 \leq 8, \\ & y_1 + y_2 + y_3 + y_4 + y_5 \leq 6,\ \ y_1 + y_2 + 7y_3 \geq 9, \\ & y_1, y_2, y_3, y_4, y_5 \geq 1. \end{cases}$$

Problem 8 [23,26,28]

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

Problem 9 [10,23]

$$\min \prod_{i=1}^{2}\left(\sum_{j=1}^{n} \hat{c}_{ij}y_j + 1\right) \quad \text{s.t.} \quad \hat{A}y \le \hat{b}, y \ge 0.$$

where $\hat{c}_{ij}$ is generated randomly in the interval [0,1], $i = 1, 2, j = 1, ..., n$. All elements $\hat{a}_{ij}$ of the matrix $\hat{A}$ are randomly generated in the interval [-1,1], i.e., $\hat{a}_{ij} = 2\hat{\lambda} - 1$, where $\hat{\lambda}$ is randomly generated in [0,1]. The components of $\hat{b}$ is set to $\sum_{j=1}^{n} \hat{a}_{ij} + 2\hat{t}$, where $\hat{t}$ is randomly generated at [0,1].

Problem 10 [23,26]

$$\begin{cases} \min & \prod_{i=1}^{p}\sum_{j=1}^{n} \widetilde{c}_{ij}y_j \\ \text{s.t.} & \sum_{j=1}^{n} \widetilde{a}_{mj}y_j \le \widetilde{b}_m, m = 1, ..., M. \\ & 0 \le y_j \le 1, j = 1, ..., n. \end{cases}$$

where $\widetilde{c}_{ij}$ is randomly generated in [0,1]. $\widetilde{a}_{mj}$ is randomly generated at [-1,1], $m = 1, ..., M, j = 1, ..., n$. $\widetilde{b}_m = \sum_{j=1}^{n} \widetilde{a}_{mj} + 2\widetilde{t}$, where $\widetilde{t}$ is randomly generated at [0,1].

Problem 11 [26,28]

$$\begin{cases} \min & \prod_{i=1}^{p}\left(\sum_{j=1}^{n} \overline{c}_{ij}y_j + \overline{d}_i\right)^{\overline{\alpha}^i} \\ \text{s.t.} & \overline{A}y \le \overline{b}, y \ge 0. \end{cases}$$

where $\overline{c}_{ij}$, and $\overline{d}_i$ are randomly generated in [0,1], $i = 1, ..., p, j = 1, ..., n$. Each element of the matrix $\overline{A}$ and $\overline{\alpha}^i$ are randomly generated in [-1,1]. The components of the vector $\overline{b}$ are generated by $\sum_{j=1}^{n} \overline{a}_{ij} + 2\overline{t}$, where $\overline{t}$ is generated randomly at [0,1].

Table 1 shows the numerical comparison between some algorithms and our algorithm on Problems 1–8.

Table 1. Numerical comparisons among some other algorithms and our algorithm on Problems 1–8.

Problems	Algorithms	Optimal solution	Opt.val	Iter	Time	Tolerance
1	Algorithm in [10]	(0,0)	0.5333	290	41.4735	10^{-3}
	Algorithm in [26]	(0,0)	0.5333	68	1.1780	10^{-6}
	Algorithm in [28]	(0,0)	0.5333	67	1.2418	10^{-6}
	Our algorithm	(0,0)	0.5333	3	0.0350	10^{-6}
2	Algorithm in [10]	(1.9975,8)	9.9725	122	17.1740	10^{-3}
	Algorithm in [26]	(2,8)	10	4	0.05146	10^{-6}
	Algorithm in [28]	(2,8)	10	1	0.00003	10^{-6}
	Our algorithm	(2,8)	10	1	0.00004	10^{-6}
3	Algorithm in [10]	(1,1)	997.6613	29	4.0872	10^{-3}
	Algorithm in [26]	(1.0,1.0)	997.6613	1	0.0000351	10^{-6}
	Algorithm in [28]	(1.0,1.0)	997.6613	1	0.0000403	10^{-6}
	Our algorithm	(1.0,1.0)	997.6613	1	0.0000389	10^{-6}
4	Algorithm in [26]	(1.25,1.00)	263.78893	2	0.010338	10^{-6}
	Algorithm in [28]	(1.25,1.00)	263.78893	2	0.013769	10^{-6}
	Our algorithm	(1.25,1.00)	263.78893	2	0.00876	10^{-6}
5	Algorithm in [10]	(3.0000,1.9990)	5.01105	48	6.66627	10^{-3}
	Algorithm in [26]	(3,2)	5.00931	1	0.0000338	10^{-6}
	Algorithm in [28]	(3,2)	5.00931	1	0.0000348	10^{-6}
	Our algorithm	(3,2)	5.00931	1	0.0000317	10^{-6}
6	Algorithm in [26]	(0,8,1)	0.90123	10	0.1977365	10^{-6}
	Algorithm in [28]	(8,0,1)	0.90123	9	0.17455	10^{-6}
	Our algorithm	(8,0,1)	0.90123	9	0.1405001	10^{-6}
7	Algorithm in [26]	(1,2.000,1,1,1)	9504.0	5	0.0826865	10^{-6}
	Algorithm in [28]	(1,2,1,1,1)	9504.0	1	0.0000387	10^{-6}
	Algorithm in [23]	(1,2,1,1,1)	9503.9999	2	0.069	10^{-6}
	Our algorithm	(1,2,1,1,1)	9504.0	1	0.0000542	10^{-6}
8	Algorithm in [26]	(1.3148,0.1396,0,0.4233)	0.8902	2	0.0194	10^{-6}
	Algorithm in [28]	(1.3148,0.1396,0,0.4233)	0.8902	2	0.0394	10^{-6}
	Algorithm in [23]	(1.3148,0.1396,0,0.4233)	0.8902	1	0.0266	10^{-6}
	Our algorithm	(1.3148,0.1396,0,0.4233)	0.8902	2	0.0093	10^{-6}

For stochastic Problems 9–11, we solve 10 randomly generated problems for each set of parameters (p, m, n) and place their average number of iterations and average CPU time in Tables 2–6. Specifically, Problem 9 represents an LMP with only two linear functions and exponents of 1. Table 2 shows the results of numerical comparisons between our algorithm and the algorithms proposed in [10, 23]. Problem 10 is an LMP with multiple linear functions and exponents of 1. Tables 3 and 4 display the numerical results of our algorithm compared with the algorithms in [23, 26]. Additionally, Figures 1 and 2 plot some of the data results in Table 3. Problem 11 is an LMP with real exponents and multiple linear functions. Tables 5 and 6 show the numerical results of our algorithm compared with the algorithms in [26, 28]. Figures 3–6 depict some of the data results from Tables 5 and 6.

For convenience, the symbols in the table headers in Tables 1–6 are specified as follows: Opt.val: the global optimum of the tested problem; Iter: the number of iterations of the algorithm; Time: the CPU time in seconds; Avg.Iter: the average number of iterations of the 10 randomly generated problems; Std.Iter: the standard deviation of the number of iterations; Avg.Time: the average CPU time of the 10 randomly generated problems; Std.Time: the standard deviation of the average CPU time; p: the number of linear functions; m: the number of constraints; n: the dimensionality of decision variables.

As can be seen from the numerical results in Table 1, our algorithm effectively calculates the global optimal solutions and optimal values for low-dimensional Problems 1–8. In comparison to the algorithms proposed in [10, 26, 28], our algorithm demonstrates shorter computation time and fewer iterations. Most deterministic problems require only one iteration, with a maximum of nine iterations, indicating the feasibility of our algorithm.

Upon observing Table 2, it is evident that our algorithm exhibits a lower average number of iterations and shorter average CPU time compared to the algorithm proposed in [10] for medium-scale Problem 9. The primary reason for this disparity is that our algorithm solves the problem in the p-dimensional space, whereas the algorithm in [10] tackles it in the n-dimensional space. In comparison to the algorithm presented in [23], it is apparent that the iterations of the algorithm in [23] is generally lower than our algorithm. However, when considering the average CPU time, our algorithm outperforms in terms of efficiency.

Table 2. Numerical comparisons among the algorithms in [10, 23] and our algorithm on Problem 9.

(m,n)	Our algorithm		Algorithm in [10]		Algorithm in [23]	
	Avg(Std).Iter	Avg(Std).Time	Avg(Std).Iter	Avg(Std).Time	Avg(Std).Iter	Avg(Std).Time
(10,20)	10.3(3.4655)	0.1712(0.0666)	14.2 (1.5492)	0.6062 (0.0695)	2.6 (6.2561)	0.2083 (0.3861)
(20,20)	8.9(3.7802)	0.2018(0.1548)	17.4 (1.7127)	0.8368 (0.0756)	4.8 (6.9793)	0.2814 (0.5504)
(22,20)	8.8(3.8678)	0.1375(0.0726)	18.5 (1.9003)	0.9460 (0.1235)	6.0 (12.2564)	0.3231 (0.9257)
(20,30)	12.3(3.8223)	0.2059(0.0737)	19.9 (0.5676)	1.0781 (0.0674)	6.4 (7.4951)	0.3302 (0.4899)
(35,50)	11.6(1.8)	0.2234(0.0404)	21.2 (0.4316)	1.8415 (0.1338)	8.1 (11.6772)	0.4267(0.8646)
(45,60)	11.3(2.1471)	0.3004(0.1370)	23.0 (0.6667)	2.4338 (0.1016)	8.7 (14.2688)	0.4867 (0.8930)
(40,100)	11.3(0.9)	0.3790(0.1208)	35.7 (1.1595)	5.1287 (0.0935)	11.9 (12.3809)	0.6049 (0.9664)
(60,100)	11.9(2.3854)	0.4781(0.1569)	36.1 (0.7379)	6.8143 (0.1713)	9.7 (12.9822)	0.7955 (1.2783)
(70,100)	11.1(1.7578)	0.4682(0.1673)	36.6 (1.2649)	8.1967 (0.2121)	8.3 (11.6638)	0.8152 (1.3057)
(70,120)	11.9(3.7802)	0.5736(0.4449)	39.1 (1.6633)	9.5642 (0.2975)	10.1 (14.6462)	0.9693 (1.3529)
(100,100)	9.1(1.8682)	0.5069(0.2585)	37.5 (2.1731)	13.0578 (0.3543)	11.1 (9.0549)	1.1889 (1.2506)

From Table 3, it is easy to see that the proposed algorithm is more efficient in solving Problem 10. First, when compared to the algorithm in [26], our algorithm consumes less time and requires fewer iterations. Second, in comparison to the algorithm in [23], our algorithm exhibits a lower number of iterations for parameter sets where p is fixed at 7. Although the algorithm in [23] may have less iterations than ours for some parameter groups, our average CPU time is still lower than that of [23].

For instance, when $p = 4$, our algorithm demonstrates higher iterations than [23] as m and n vary, yet our CPU time is shorter than theirs.

Table 3. Numerical comparisons among the algorithms in [23, 26] and our algorithm on medium-sized Problem 10.

p	(m,n)	Our algorithm		Algorithm in [23]		Algorithm in [26]	
		Avg.Iter	Avg.Time	Avg.Iter	Avg.Time	Avg.Iter	Avg.Time
4	(10,20)	45.3	0.5224	10.8	0.6654	66.1	0.8137
	(20,40)	56.4	0.7212	25.1	1.1668	86.8	1.1674
	(30,60)	61.1	0.8918	32.6	1.8145	89.4	1.3799
	(40,80)	54.9	0.9787	46.4	2.2780	89.1	1.6630
	(50,100)	59.8	1.3421	42.6	2.5277	89.3	2.1077
	(60,120)	64.1	1.8703	76.5	6.5797	96.7	2.9471
	(70,140)	62.9	2.3383	62.2	8.1493	95.1	3.6654
	(80,160)	73.7	3.4148	43.8	14.7688	109.5	5.2437
	(90,180)	70.1	4.1754	37.2	19.2281	110.7	6.8075
	(100,200)	69.7	5.2313	79.2	46.8494	111.7	8.6532
5	(10,20)	77.8	0.9546	45.7	1.2609	120.9	1.5939
	(20,40)	89.5	1.2246	45.7	1.3929	142.1	2.0816
	(30,60)	104.8	1.6322	184.6	7.6766	180.5	3.0199
	(40,80)	100.8	1.9369	117.6	10.4108	154.9	3.1245
	(50,100)	100	2.3850	167	15.5902	178.9	4.4852
	(60,120)	117.2	3.5787	143.4	18.3711	203.8	6.5421
	(70,140)	123.3	4.5434	212.5	38.7537	231.6	9.0070
	(80,160)	97.6	4.6827	310.2	81.1816	175.8	8.7630
	(90,180)	116.1	6.8339	298.4	93.8722	211.9	12.8401
	(100,200)	98.8	7.4864	230.6	114.7671	176.9	13.7859
6	(10,20)	88.5	1.0468	43.8	1.7287	158.5	1.9954
	(20,40)	116.9	1.5515	77.0	6.6506	204.6	2.8660
	(30,60)	171	2.5808	101.9	11.1125	334.9	5.3399
	(40,80)	175.5	3.2212	133.1	16.3945	342.3	6.6506
	(50,100)	163.8	3.8391	271.2	22.0331	307.7	7.5644
	(60,120)	147.3	4.3689	387.1	25.2491	289.2	9.0304
	(70,140)	168.6	6.3815	766.5	63.8334	327.5	12.8848
	(80,160)	187.2	8.7670	315.4	98.5674	384	18.5743
	(90,180)	166.5	9.8465	444.2	110.7434	310.2	18.9719
	(100,200)	173.4	13.2959	657.8	120.0034	323.5	25.7361
7	(10,20)	133.6	1.5839	458.8	19.2006	284.1	3.6194
	(20,40)	140.4	1.8923	496.6	23.8151	265.2	3.7999
	(30,60)	182.1	2.8005	1013.5	64.3865	330.2	5.3898
	(40,80)	202.7	3.7863	1177.3	86.1229	470.7	9.3001
	(50,100)	250.7	5.8942	1398	127.1028	542.5	13.4583
	(60,120)	225.4	6.8489	640.2	155.5248	456.0	14.5520
	(70,140)	275.6	10.6270	1197.4	185.5591	604.2	24.1463
	(80,160)	258.9	12.7324	891.1	278.3208	542.2	27.5974
	(90,180)	211.3	13.1138	816.7	293.8722	463.1	29.6745
	(100,200)	234.4	18.4294	469.3	321.4063	510.6	41.1299

To visualize the effectiveness of the proposed algorithm in this paper, Figures 1 and 2 depict the trends of the average number of iterations and average CPU time for fixing (m, n) to (10,20), (100,200) and p to change from 4 to 7 in Problem 10, respectively. From Figure 1, when $(m, n) = (10, 20)$ (indicated by the solid line), the green solid line is always above the blue solid line, indicating that the number of iterations of the algorithm in [26] is higher than ours. On the other hand, the red solid line is above the blue solid line only after $p = 6$, which means that the iterations of the algorithm in [23] are higher than ours after $p > 6$. When $(m, n) = (100, 200)$ (indicated by the dashed line), both the red dashed line and the green dashed line are above the blue dashed line, indicating that the number of iterations of our algorithm is lower than the other two algorithms. In the vertical direction, the blue dashed line is always higher than the blue solid line, but the vertical distance between them is shorter than the corresponding vertical distance of the other two algorithms. This implies that as (m, n) increases, the number of iterations of our algorithm exhibits a slight increase, but the magnitude of the increase is not significant. Based on Figure 2, we observe that when $(m, n) = (10, 20)$, $p = 4, 5, 6$, the three solid lines approximately coincide and when $p = 7$, it is obvious that our algorithm takes less time. When $(m, n) = (100, 200)$, the distance between the red dashed line and the blue dashed line becomes larger as p increases. The time taken by our algorithm increases as (m, n) increases, but not significantly.

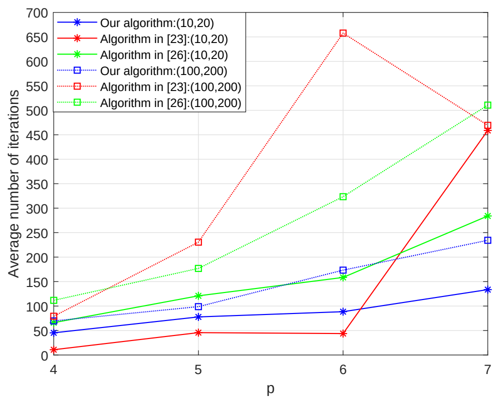

Figure 1. Comparison of average number of iterations in Problem 10.

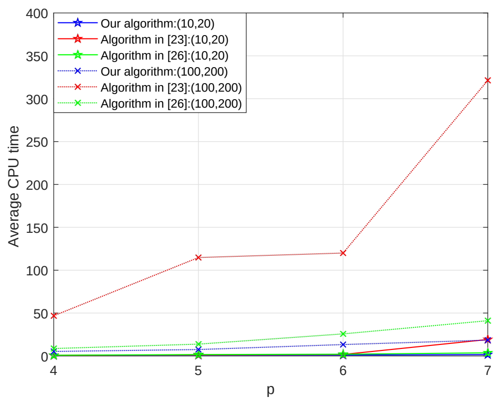

Figure 2. Comparison of average CPU time in Problem 10.

Table 4 gives the numerical results for solving the large-scale Problem 10. It can be observed that our algorithm is more time-saving compared to the algorithm in [26]. In contrast to the algorithm in [23], the first two sets of parameters yield better results than ours. However, for the case of $(m, n) = (3, 10, 1000)$, the number of iterations in our algorithm is comparable to that of [23], but our algorithm exhibits lower time consumption. When $(p, m, m) = (3, 10, 2000)$, although our algorithm requires fewer iterations than that of [23], the CPU time is slightly longer. When $(p, m, n) = (4, 10, 1000), (4, 10, 2000)$, respectively, our algorithm demonstrates clear advantages. Specifically, the algorithm in [23] requires twice as many iterations and three times as much CPU time as ours.

Table 4. Numerical comparisons among the algorithms in [23, 26] and our algorithm on large-scale Problem 10.

(p, m, n)	Our algorithm		Algorithm in [23]		Algorithm in [26]	
	Avg.Iter	Avg.Time	Avg.Iter	Avg.Time	Avg.Iter	Avg.Time
(2,10,1000)	27.7	4.1565	15.5	2.6293	39.7	6.2742
(2,10,2000)	33.9	18.1454	28.5	14.0012	48.6	26.5702
(3,10,1000)	106.1	16.8598	101.8	19.3235	193.7	31.8366
(3,10,2000)	173.4	98.8698	185.4	90.3898	352.7	204.3262
(4,10,1000)	361.8	59.9727	757.6	156.5649	878.4	149.7120
(4,10,2000)	552.6	332.8818	1352.1	995.4707	1519.3	921.7023

Table 5 shows the results of a medium-scale test for Problem 11. Compared to the algorithms in [26, 28], our algorithm exhibits a distinct advantage. Figures 3 and 4 show the average number of iterations and the average CPU time among the algorithms in [26, 28] and our algorithm when (m, n) is fixed (10,20), (100,200) on Problem 11, respectively. From Figure 3, it can be observed that the average number of iterations in the algorithm proposed by [26] is higher than our algorithm when $(m, n) = (10, 20)$ and lower than our algorithm only when $p = 5$. When $(m, n) = (100, 200)$, the average number of iterations in our algorithm is lower than those of the algorithms in [26, 28]. Moreover, our algorithm significantly outperforms the algorithms in [26, 28] when $(m, n) = (100, 200)$. From Figure 4, we find that while $(m, n) = (10, 20), (100, 200)$, the average CPU time of the proposed algorithm is less than the other two algorithms, and the trend of time change is not obvious as p increases.

Table 5. Numerical comparisons among the algorithms in [26, 28] and our algorithm on medium-scale Problem 11.

p	(m,n)	our		[26]		[28]	
		Avg(Std).Iter	Avg(Std).Time	Avg(Std).Iter	Avg(Std).Time	Avg(Std).Iter	Avg(Std).Time
4	(10,20)	31.9(32.8312)	0.4150(0.4459)	120.1(118.6612)	2.0917(2.0855)	112.2(97.3764)	1.8487(1.5777)
	(20,40)	35.4(28.3944)	0.5178(0.4354)	142(109.9582)	2.6908(2.1186)	114.3(92.8041)	2.0269(1.6408)
	(30,60)	46.6(56.5194)	0.8169(1.0093)	93.1(58.7409)	1.9714(1.3202)	93.0(69.5054)	1.8041(1.2743)
	(40,80)	34.5(33.5894)	0.7204(0.7070)	85.6(74.8441)	2.0867(1.8616)	63.5(54.9641)	1.4810(1.3050)
	(50,100)	40.8(39.5596)	1.0568(1.0764)	91.8(99.0372)	2.6335(2.8371)	71.1(54.9735)	1.8772(1.4905)
	(60,120)	21.8(17.0458)	0.6746(0.5693)	89.2(61.4684)	3.2295(2.2926)	68.5(51.9870)	2.3788(1.8911)
	(70,140)	16.0(11.1893)	0.6336(0.4974)	78.7(80.9519)	3.6159(3.7190)	63.7(76.3375)	2.7302(3.2902)
	(80,160)	27.8(19.9038)	1.4628(1.0860)	88.7(53.3555)	5.1639(3.3558)	74.6(49.2020)	4.2010(2.9670)
	(90,180)	21.2(21.6601)	1.3846(1.5300)	67.0(62.3217)	4.6670(4.5688)	59.6(54.8584)	3.9347(3.8613)
	(100,200)	37.3(30.3942)	3.3410(3.0137)	122.5(55.1693)	11.0229(5.1801)	96.9(53.5807)	8.3233(4.7203)
5	(10,20)	123.2(173.7980)	1.6461(2.3781)	160.5(122.7992)	2.6874(2.0730)	98.4(70.0303)	1.5646(1.0756)
	(20,40)	57.8(77.3586)	0.8393(1.16481)	101.5(90.8947)	1.8805(1.7787)	71.6(67.9591)	1.3059(1.3715)
	(30,60)	63.2(39.3568)	1.1174(0.7884)	155.5(135.708)	3.3789(3.0502)	98.4(88.7437)	2.0061(1.7969)
	(40,80)	54.9(95.2979)	1.3205(2.362)	203.9(146.9084)	5.4791(4.0027)	145.5(118.9128)	3.7432(3.0424)
	(50,100)	42.3(33.77)	1.1217(0.9049)	343.5(382.1019)	10.8698(11.9164)	267.8(361.9276)	8.0793(10.2894)
	(60,120)	51.0(47.8623)	1.8003(1.8125)	268.0(146.4281)	10.6335(6.2604)	191(131.6716)	7.2388(5.0948)
	(70,140)	31.8(40.5038)	1.3226(1.8188)	117.8(114.8893)	5.4516(5.4995)	86.9(82.3619)	3.8601(3.8865)
	(80,160)	30.4(29.2684)	1.5684(1.5981)	67.5(55.9897)	3.86(3.2299)	46.0(21.5639)	2.4507(1.2596)
	(90,180)	31.9(31.3989)	2.0009(2.0368)	96.6(77.3139)	6.8571(5.7645)	68.4(65.7529)	4.6769(4.6839)
	(100,200)	33.4(33.6309)	3.1802(3.3664)	198.4(169.6061)	19.0294(16.5882)	119.5(105.2675)	11.0382(9.6674)
6	(10,20)	99.3(117.1632)	2.1315(2.2123)	466.3(514.9322)	12.2852(14.0562)	356(416.9149)	8.8802(9.4876)
	(20,40)	348.7(651.2854)	5.6616(10.6349)	439.7(617.1514)	9.5253(14.0514)	287.9(355.6457)	4.7303(5.8901)
	(30,60)	83.8(62.7117)	1.3889(1.0594)	197.5(149.0156)	4.0838(3.1896)	109.3(72.8190)	2.1971(1.5108)
	(40,80)	89.3(116.0233)	2.8242(3.4539)	385.6(404.1607)	15.9517(16.6636)	267.0(311.3747)	10.6983(12.9213
	(50,100)	215.1(205.3667)	10.4319(10.0666)	418.3(258.3912)	19.7742(13.2598)	280.8(200.1099)	13.2487(9.9695)
	(60,120)	64.0(104.2929)	3.7668(6.0989)	536.4(451.0098)	32.1704(27.5837)	386.8(327.6687)	23.1508(20.6228)
	(70,140)	127.5(136.0230)	9.7495(10.2623)	239.6(159.1353)	16.9048(11.5614)	143.4(101.6545)	10.1235(7.9464)
	(80,160)	128.0(247.0591)	12.2207(23.8867)	255.10(288.4623)	23.7336(27.8770)	168.7(192.1182)	15.9113(19.6485)
	(90,180)	140.2(191.3007)	17.0936(23.5297)	392.10(326.3545)	46.8629(40.9758)	256.1(248.8777)	30.1672(29.1070)
	(100,200)	65.1(60.5532)	5.5949(5.6379)	214.8(257.7537)	18.8967(23.1908)	133.2(188.4955)	11.3892(16.4153)
7	(10,20)	134.3(171.9087)	1.9250(2.4624)	434.7(817.5430)	7.6743(14.2808)	268.8(489.3136)	4.7123(8.6148)
	(20,40)	379.6(645.6239)	6.1832(9.991)	1029.2 (2443.5000)	19.9616(46.747)	492.3(1137.0000)	8.843(19.9798)
	(30,60)	359.5(780.0254)	6.8852(14.7182)	294.5(595.5877)	6.9729(14.0042)	209.1(425.2718)	4.4931(8.9504)
	(40,80)	72.0(152.2281)	1.5577(3.3768)	416.8(624.1418)	10.5708(15.7193)	240.5(412.5630)	5.9042(10.0912)
	(50,100)	39.8(61.3560)	0.9973(1.5611)	281.7(481.0314)	8.1550(14.1022)	192.1(366.9682)	5.2928(10.2618)
	(60,120)	164.5(227.7056)	5.9599(8.0338)	672.5(906.5232)	26.0214(34.4929)	444.9(682.5063)	16.7892(24.8233)
	(70,140)	77.3(98.5231)	2.6876(3.4708)	276.2(303.7498)	10.102(11.0515)	139.6(142.7895)	5.13(5.3474)
	(80,160)	74.1(122.941)	5.2043(8.8393)	474.8(780.9027)	27.3175(45.6091)	289.6(552.5047)	16.8792(32.7485)
	(90,180)	70.0(140.4179)	5.2749(10.985)	429.5(705.4637)	30.0896(48.8796)	318.2(562.72)	22.3532(39.7114)
	(100,200)	70.5(96.0742)	6.8665(9.7942)	694.6(1073.2000)	68.3707(108.5662)	384.0(623.8142)	34.5013(56.7705)

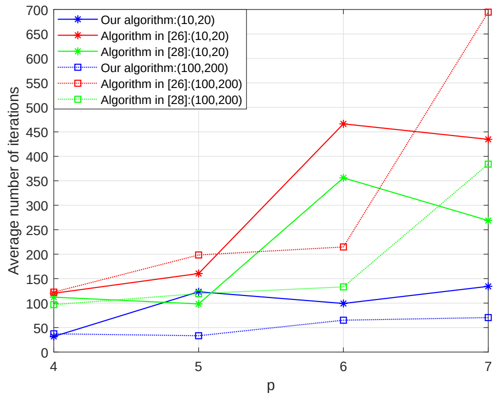

Figure 3. Comparison of average number of iterations in problem 11.

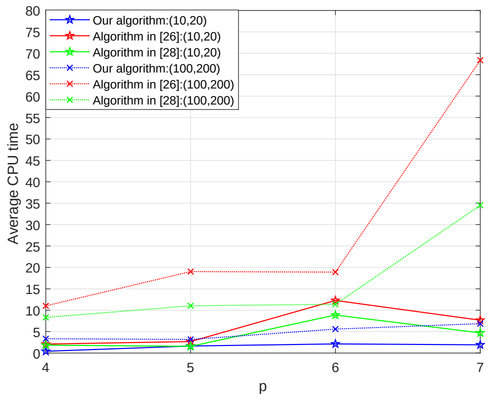

Figure 4. Comparison of average CPU time in problem 11.

Table 6 shows the numerical results of a large-scale Problem 11, we can see that the number of linear functions p is much smaller than n. Fixing $(p, m) = (2, 10)$, the number of iterations of our algorithm decreases as the decision variable n increases, while the number of iterations of the algorithms in the references [26, 28] decreases and then increases as the decision variable n increases, as reflected in Figure 5. Moreover, we find that the algorithm in [26,28] consume more CPU time than our algorithm. Furthermore, we conclude that CPU time increases with an increase in the decision variable or linear function. Figure 6 provides a clearer picture of this conclusion.

Table 6. Numerical comparisons among the algorithms in [26, 28] and our algorithm on large-scale Problem 11.

(p, m, n)	Our algorithm		Algorithm in [26]		Algorithm in [28]	
	Avg(Std).Iter	Avg(Std).Time	Avg(Std).Iter	Avg(Std).Time	Avg(Std).Iter	Avg(Std).Time
(2,10,1000)	11.3(5.6223)	1.8230(1.0482)	28.1(29.4192)	4.9508(5.5044)	33.2(47.4042)	4.9860(6.9212)
(2,10,2000)	13.1(8.8707)	7.6748(5.8547)	38.1(35.5006)	24.5339(24.0695)	45(45.1597)	24.0173(23.0733)
(2,10,3000)	10.3(4.9406)	13.2076(7.6065)	21.1(14.1736)	29.3624(21.9892)	66.9(60.9843)	71.8731(62.9537)
(2,10,4000)	8.8(8.0100)	18.9312(20.8423)	35.9(29.7740)	90.5241(79.8744)	59.1(58.9024)	117.6200(111.2727)
(3,10,1000)	77.1(61.2576)	15.0725(12.4141)	216.1(156.9888)	41.7395(30.3254)	243.6(199.4228)	40.3785(32.9021)
(3,10,2000)	161.8(394.4629)	114.8995(283.4199)	250.1(365.6067)	174.4885(259.0985)	349.7(438.3820)	202.5456(251.7850)
(4,10,1000)	80.5(84.4562)	14.3938(15.3816)	887.9(706.9598)	163.7237(130.5653)	1047.7(944.1833)	168.5456(152.1507)
(4,10,2000)	290.2(348.2453)	169.9271(208.815)	739.2(749.6237)	423.9934(419.0576)	387.2(331.2992)	208.3637(176.7977)

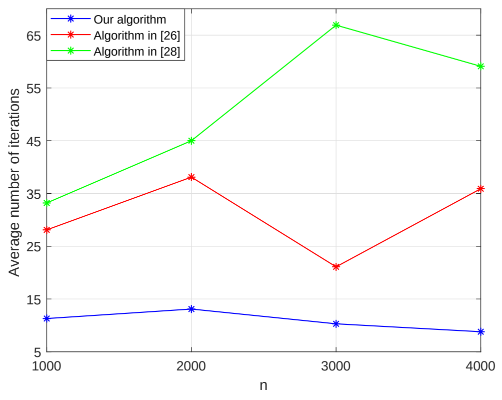

Figure 5. Comparison of average number of iterations in problem 11.

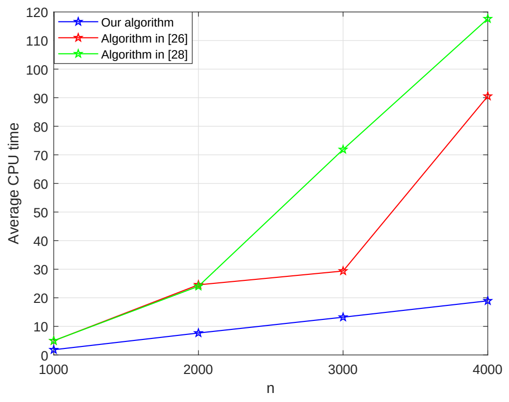

Figure 6. Comparison of average CPU time in problem 11.

5. Conclusions

We propose a new branch-and-bound algorithm for solving LMP in outer space. First, the original problem is reduced to an equivalent problem by introducing auxiliary variables. Then, the equivalent problem is further simplified by leveraging the properties of exponential and logarithmic functions.The focus switches to resolving the second equivalent problem. Afterwards, the nonlinear constraints are linearly relaxed, and the objective function of the equivalent problem is linearly approximated to obtain the linear relaxation programming problem. Consequently, the proposed rectangular compression technique is embedded into the branch-and-bound framework, leading to the development of the outer space branch-and-bound algorithm. Furthermore, we demonstrate the computational complexity to estimate the maximum number of iterations of the algorithm in the worst case. Finally, in order to verify the feasibility and efficiency of the algorithm, some deterministic and random problems are tested. The experimental results show that the algorithm exhibits good computational performance, particularly for large-scale random LMP where the number of linear functions p is significantly smaller than the decision variable n.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (11961001), the Construction Project of first-class subjects in Ningxia higher Education(NXYLXK2017B09), the Basic discipline research projects supported by Nanjing Securities(NJZQJCXK202201).

Conflict of interest

The authors declare no conflicts of interest.

References

1. J. Mulvey, R. Vanderbei, S. Zenios, Robust optimization of large-scale systems, *Oper. Res.*, **43** (1995), 264–281. https://doi.org/10.1287/opre.43.2.264

2. H. Konno, H. Shirakawa, H. Yamazaki, A mean-absolute deviation-skewness portfolio optimization model, *Ann. Oper. Res.*, **45** (1993), 205–220. https://doi.org/10.1007/BF02282050

3. M. Domeich, N. Sahinidis, Global optimization algorithms for chip design and compaction, *Eng. Optimiz.*, **25** (1995), 131–154. https://doi.org/10.1080/03052159508941259

4. K. Bennett, Global tree optimization: a non-greedy decision tree algorithm, *Computing Science and Statistics*, **26** (1994), 156–161.

5. R. Cambini, C. Sodini, On the minimization of a class of generalized linear functions on a flow polytope, *Optimization*, **63** (2014), 1449–1464. https://doi.org/10.1080/02331934.2013.852548

6. S. Qu, L. Shu, J. Yao, Optimal pricing and service level in supply chain considering misreport behavior and fairness concern, *Comput. Ind. Eng.*, **174** (2022), 108759. https://doi.org/10.1016/j.cie.2022.108759

7. C. Maranas, I. Androulakis, C. Floudas, A. Berger, J. Mulvey, Solving long-term financial planning problems via global optimization, *Comput. Ind. Eng.*, **21** (1997), 1405–1425. https://doi.org/10.1016/S0165-1889(97)00032-8

8. T. Matsui, NP-hardness of linear multiplicative programming and related problems, *J. Glob. Optim.*, **9** (1996), 113–119. https://doi.org/10.1007/bf00121658

9. H. Ryoo, N. Sahinidis, Global optimization of multiplicative programs, *J. Glob. Optim.*, **26** (2003), 387–418. https://doi.org/10.1023/A:1024700901538

10. C. Wang, S. Liu, A new linearization method for generalized linear multiplicative programming, *Comput. Oper. Res.*, **38** (2011), 1008–1013. https://doi.org/10.1016/j.cor.2010.10.016

11. Y. Zhao, S. Liu, An efficient method for generalized linear multiplicative programming problem with multiplicative constraints, *SpringerPlus*, **5** (2016), 1302. https://doi.org/10.1186/s40064-016-2984-9

12. P. Shen, H. Jiao, Linearization method for a class of multiplicative programming with exponent, *Appl. Math. Comput.*, **183** (2006), 328–336. https://doi.org/10.1016/j.amc.2006.05.074

13. C. Wang, Y. Bai, P. Shen, A practicable branch-and-bound algorithm for globally solving linear multiplicative programming, *Optimization*, **66** (2017), 397–405. https://doi.org/10.1080/02331934.2016.1269765

14. P. Shen, B. Huang, Global algorithm for solving linear multiplicative programming problems, *Optim. Lett.*, **14** (2020), 693–710. https://doi.org/10.1007/s11590-018-1378-z

15. S. Schaible, C. Sodini, Finite algorithm for generalized linear multiplicative programming, *J. Optim. Theory Appl.*, **87** (1995), 441–455. https://doi.org/10.1007/bf02192573

16. X. Liu, T. Umegaki, Y. Yamamoto, Heuristic methods for linear multiplicative programming, *J. Glob. Optim.*, **15** (1999), 433–447. https://doi.org/10.1023/A:1008308913266

17. Y. Zhao, J. Yang, Inner approximation algorithm for generalized linear multiplicative programming problems, *J. Inequal. Appl.*, **2018** (2018), 354. https://doi.org/10.1186/s13660-018-1947-9

18. B. Zhang, Y. Gao, X. Liu, X. Huang, An efficient polynomial time algorithm for a class of generalized linear multiplicative programs with positive exponents, *Math. Probl. Eng.*, **2021** (2021), 8877037. https://doi.org/10.1155/2021/8877037

19. H. Konno, Y. Yajima, T. Matsui, Parametric simplex algorithms for solving a special class of nonconvex minimization problems, *J. Glob. Optim.*, **1** (1991), 65–81. https://doi.org/10.1007/BF00120666

20. P. Bonami, A. Lodi, J. Schweiger, A. Tramontani, Solving quadratic programming by cutting planes, *SIAM J. Optimiz.*, **29** (2019), 1076–1105. https://doi.org/10.1137/16M107428X

21. E. Youness, Level set algorithm for solving convex multiplicative programming problems, *Appl. Math. Comput.*, **167** (2005), 1412–1417. https://doi.org/10.1016/j.amc.2004.08.028

22. Y. Gao, C. Xu, Y. Yang, An outcome-space finite algorithm for solving linear multiplicative programming, *Appl. Math. Comput.*, **179** (2006), 494–505. https://doi.org/10.1016/j.amc.2005.11.111
23. B. Zhang, Y. Gao, X. Liu, X. Huang, Output-space branch-and-bound reduction algorithm for a class of linear multiplicative programs, *Mathematics*, **8** (2020), 315. https://doi.org/10.3390/math8030315
24. Y. Zhao, T. Zhao, Global optimization for generalized linear multiplicative programming using convex relaxation, *Math. Probl. Eng.*, **2018** (2018), 9146309. https://doi.org/10.1155/2018/9146309
25. Z. Hou, S. Liu, Global algorithm for a class of multiplicative programs using piecewise linear approximation technique, *Numer. Algor.*, **92** (2023), 1063–1082. https://doi.org/10.1007/s11075-022-01330-x
26. H. Jiao, W. Wang, J. Yin, Y. Shang, Image space branch-reduction-bound algorithm for globally minimizing a class of multiplicative problems, *RAIRO-Oper Res.*, **56** (2022), 1533–1552. https://doi.org/10.1051/ro/2022061
27. C. Wang, Y. Deng, P. Shen, A novel convex relaxation-strategy-based algorithm for solving linear multiplicative problems, *J. Comput. Appl. Math.*, **407** (2022), 114080. https://doi.org/10.1016/j.cam.2021.114080
28. H. Jiao, W. Wang, Y. Shang, Outer space branch-reduction-bound algorithm for solving generalized affine multiplicative problems, *J. Comput. Appl. Math.*, **419** (2023), 114784. https://doi.org/10.1016/j.cam.2022.114784
29. H. Zhou, G. Li, X. Gao, Z. Hou, Image space accelerating algorithm for solving a class of multiplicative programming problems, *Math. Probl. Eng.*, **2022** (2022), 1565764. https://doi.org/10.1155/2022/1565764
30. H. Jiao, S. Liu, Y. Zhao, Effective algorithm for solving the generalized linear multiplicative problem with generalized polynomial constraints, *Appl. Math. Model.*, **39** (2015), 7568–7582. https://doi.org/10.1016/j.apm.2015.03.025