mathematics

Article

Output-Space Branch-and-Bound Reduction Algorithm for a Class of Linear Multiplicative Programs

Bo Zhang [1], Yuelin Gao [2,3,*], Xia Liu [1] and Xiaoli Huang [2,3]

1 School of Mathematics and Statistics, Ningxia University, Yinchuan 750021, China; zbsdx121@163.com (B.Z.); lingxiaoyu911@163.com (X.L.)
2 Ningxia Province Cooperative Innovation Center of Scientific Computing and Intelligent Information Processing, North Minzu University, Yinchuan 750021, China; hxl1569501@163.com
3 Ningxia Province Key Laboratory of Intelligent Information and Data Processing, North Minzu University, Yinchuan 750021, China
* Correspondence: gaoyuelin@nmu.edu.cn; Tel.: +86-139-9510-0900

Received: 17 November 2019; Accepted: 24 February 2020; Published: 1 March 2020

Abstract: In this paper, a new relaxation bounding method is proposed for a class of linear multiplicative programs. Although the $2p-1$ variable is introduced in the construction of equivalence problem, the branch process of the algorithm is only carried out in $p-$dimensional space. In addition, a super-rectangular reduction technique is also given to greatly improve the convergence rate. Furthermore, we construct an output-space branch-and-bound reduction algorithm based on solving a series of linear programming sub-problems, and prove the convergence and computational complexity of the algorithm. Finally, to verify the feasibility and effectiveness of the algorithm, we carried out a series of numerical experiments and analyzed the advantages and disadvantages of the algorithm by numerical results.

Keywords: global optimization; linear multiplicative programming; branch-and-bound; output-space; linear relaxation

1. Introduction

In this study, we consider the following linear multiplicative programs (LMP):

$$(LMP) := \min f(x) = \prod_{j=1}^{p}(c_j^T x + d_j) \quad s.t. \quad Ax \leq b.$$

where the feasible domain $X = \{x \in R^n | Ax \leq b\}$ is n-dimensional, nonempty, and bounded; $p \geq 2$, $A \in R^{m\times n}$, $b \in R^m$, $c_j \in R^n$, $d_j \in R$, and $c_j^T x + d_j > 0$.

The linear multiplicative program problem comes from many application areas, for example, financial optimization [1,2], microeconomics [3], robust optimization [4], decision tree optimization [5], multiple-objective decision [6,7], VLSI chip design [8], optimal packing and layout [9], and control problems [10–14]. It is well known that the (LMP) problem does not contain some common properties (convexity or other properties), which is an important challenge and test for solving this kind of problems. In addition, we also note that the (LMP) problem is closely related to the linear maximum multiplicative programming problem (see [14–16]). Specifically, the linear maximum multiplicative programming problem can be obtained by changing the *min* in the objective function of the (LMP) problem to *max*. Some scholars have shown that the (LMP) problem is NP-hard [17], but the linear maximum multiplicative programming problem can be solved in polynomial time. In [18], Kuno pointed out that, without limiting the positive nature of each product term in the objective function, multiplying the negative product term by the negative sign "$-$" while considering the parity of p, the original problem can eventually be classified into two categories, namely the (LMP) problem and the linear maximum multiplication programming problem. Therefore, without loss of generality, as in [18], we investigate the (LMP) problem under the positive assumption of each linear product term. It is stated in advance that the method we mention can be generalized and applied to the maximization formal programming problem of such problems.

Over the past 20 years, some practical algorithms have been developed to solve this (LMP) problem, and, with the increasing reliance on modeling and optimization in real-world problems, great progress has been made in both local and global optimization theories and algorithms. Compared with the local optimization method, the global optimization methods for solving the (LMP) problem are still very few, but the previous scholars have also done a series of research on the global optimization method for (LMP) problem. The methods can be classified as branch-and-bound methods [18–26], outer-approximation methods [27,28], vertex enumeration methods [29], heuristic methods [30,31], an outcome-space cutting plane method [32], parameterization based methods [33–35], and level set algorithm [36,37]. Shao and Ehrgott also proposed a class of global optimization algorithms for solving the (LMP) problem by using multi-objective linear programming and primitive dual conditions [38]. However, despite this progress, solving the (LMP) globally is still a thorny problem. In view of (LMP) and its variation, several global solutions are proposed. For example, Jiao [22] establishes a reliable and efficient algorithm for a class of generalized linear multiplicative programming by using the linear approximation of exponential and logarithmic functions. Shen [24] proposed a new accelerating method for solving generalized linear multiplicative programming by combining the appropriate deletion technique with the branch-and-bound scheme. To solve the linear multiplicative programming problem with exponential form, Liu and Zhao [37] proposed a level set algorithm based on the research of Youness [36], while Shen et al. [39] proposed a complete polynomial time approximation algorithm. For linear programs with multiplicative constraints, Benson [40] proposed a branch-and-bound algorithm based on decomposition.

Aiming at the (LMP) problem, this paper proposes a branch-and-bound algorithm based on the rectangular branch of the output space. First, for this purpose, an equivalent optimization problem (EP) of the problem (LMP) is proposed. Secondly, the approximation theorem of binary bilinear functions is given and a relaxation subproblem of the problem (EP) is constructed. Finally, a branch-and-bound algorithm based on output space is designed for (EP) problem. Compared with the methods in the above literature, the proposed method has the following characteristics:

(a) Our relaxation method is easy to operate and can be extended to some more generalized optimization problems, especially for the linear maximum multiplicative programming problem, which can be directly generalized.

(b) The branch operation of the branch-and-bound algorithm proposed by us acts on the p-dimensional output space, which greatly reduces the running cost of the computer.

(c) We also propose a new hyper-rectangular compression and reduction technique, which greatly improves the convergence rate of the algorithm, and then analyzes the computational complexity of the algorithm.

(d) To better illustrate the effectiveness of the proposed algorithm, we performed many numerical experiments, and compared with the relevant references to illustrate the characteristics of the algorithm.

The remainder of this article is outlined as follows. Section 2 first gives the equivalent problem (EP) of (LMP) as well as the algorithm framework, and then analyzes the problem (EP) and gives the required bounding operation, Branching operation and rectangle-reducing operation of the algorithm, respectively, and finally gives the specific steps of the algorithm. In Section 3, the algorithm is analyzed, and it is shown that the algorithm can be terminated within finite iterations. In Section 4, some numerical experiments are presented and the characteristics of our algorithm are analyzed. Finally, the method of this paper is briefly reviewed.

2. Output-Space Branch-and-Bound Reduction Algorithm for (LMP)

2.1. Convert (LMP) into an Equivalent Problem (EP)

In this subsection, we show how to convert the problem (LMP) into a non-convex programming problem (EP), and introduce $(2p-1)$-dimension vector y to obtain the initial hyper-rectangle Y^0.

First, the new variable $y_j = c_j^T x + d_j (j = 1, 2, \cdots, p)$ is introduced for the item $\prod_{j=1}^{p}(c_j^T x + d_j)$ in the objective function of the problem (LMP), and then there is the formula

$$\prod_{j=1}^{p}(c_j^T x + d_j) = \prod_{j=1}^{p} y_j.$$

Let $y_{p+s} = y_{p+s-1}y_{p-s}$, $s = 1, 2, \cdots, p-1$ satisfy

$$y_{2p-1} = y_{2p-2}y_1 = y_{2p-3}y_2y_1 = \cdots = \prod_{j=1}^{p} y_j. \tag{1}$$

by taking advantage of the internal structure of $\prod_{j=1}^{p} y_j$. Thus, the $(2p-1)$-dimensional variable y is established, that is, $y = (y_1, y_2, \cdots, y_p, y_{p+1}, y_{p+2}, \cdots, y_{2p-1})^T$.

Secondly, two different operations are used to construct an initial hyper-rectangle Y^0 of variable y in two stages. In the first stage, we construct the hyper-rectangle $\hat{Y}^0$, and, in the second stage, the hyper-rectangle $\tilde{Y}^0$ is also constructed, thus the hyper-rectangle Y^0 can be represented as $Y^0 = \hat{Y}^0 \times \tilde{Y}^0$. To obtain $\hat{Y}^0$, let

$$\underline{y}_j^0 = \min_{x \in X} c_j^T x + d_j, \overline{y}_j^0 = \max_{x \in X} c_j^T x + d_j, j = 1, 2, \cdots, p, \tag{2}$$

and $\hat{Y}^0 = [\underline{\hat{y}}^0, \hat{\overline{y}}^0]$, $\underline{\hat{y}}^0 = (\underline{y}_1^0, \underline{y}_2^0, \cdots, \underline{y}_p^0)^T, \hat{\overline{y}}^0 = (\overline{y}_1^0, \overline{y}_2^0, \cdots, \overline{y}_p^0)^T$. The upper and lower bounds of each variable $y_{p+s} (s = 1, 2, \cdots, p-1)$ are defined by both ends of the following inequality Equation (3), that is,

$$0 < \underline{y}_{p+s}^0 = \underline{y}_{p+s-1}^0 \underline{y}_{p-s}^0 \leq y_{p+s} = y_{p+s-1}y_{p-s} \leq \overline{y}_{p+s-1}^0 \overline{y}_{p-s}^0 = \overline{y}_{p+s}^0. \tag{3}$$

Then, the initial hyper-rectangle $\tilde{Y}^0$ of the variable y_{p+s} can be recorded as $\tilde{Y}^0 = [\underline{\tilde{y}}^0, \tilde{\overline{y}}^0]$ with $\underline{\tilde{y}}^0 = (\underline{y}_{p+1}^0, \underline{y}_{p+2}^0, \cdots, \underline{y}_{2p-1}^0)^T, \tilde{\overline{y}}^0 = (\overline{y}_{p+1}^0, \overline{y}_{p+2}^0, \cdots, \overline{y}_{2p-1}^0)^T$. The initial hyper-rectangle Y^0 for the variable y can be represented as

$$Y^0 = \hat{Y}^0 \times \tilde{Y}^0 = \prod_{j=1}^{p}[\underline{y}_j^0, \overline{y}_j^0] \times \prod_{s=1}^{p-1}[\underline{y}_{p+s}^0, \overline{y}_{p+s}^0] = \prod_{j=1}^{2p-1}[\underline{y}_j^0, \overline{y}_j^0]$$

by using the representation of the Cartesian product. Of course, for each sub-rectangular $Y^k \subseteq Y^0$, we also define

$$0 < \underline{y}^k_{p+s} = \underline{y}^k_{p+s-1}\underline{y}^k_{p-s} \le y_{p+s} = y_{p+s-1}y_{p-s} \le \overline{y}^k_{p+s-1}\overline{y}^k_{p-s} = \overline{y}^k_{p+s}, \tag{4}$$

and

$$Y^k = \hat{Y}^k \times \tilde{Y}^k = \prod_{j=1}^{p}[\underline{y}^k_j, \overline{y}^k_j] \times \prod_{s=1}^{p-1}[\underline{y}^k_{p+s}, \overline{y}^k_{p+s}] = \prod_{j=1}^{2p-1}[\underline{y}^k_j, \overline{y}^k_j].$$

Finally, through the above content, we naturally establish the problem (LMP) in the following form of the equivalent optimization problem (EOP), that is,

$$(EOP) := \min y_{2p-1}, \quad s.t. \begin{cases} c_j^T x + d_j = y_j, \ j = 1, 2, \cdots, p, \\ y_{p+s} = y_{p+s-1}y_{p-s}, \ s = 1, 2, \cdots, p-1, \\ x \in X, y \in Y^0. \end{cases}$$

In particular, it is observed that (EOP) has $n + 2p - 1$ variables. We define the function $h(x, y) = C^T(x^T, y^T)^T = y_{2p-1}$ with $C = (0, 0, \cdots, 0, 1)^T \in R^{n+2p-1}$, and then (EOP) can be rewritten as the equivalent problem (EP):

$$(EP) := \min h(x, y) = C^T(x^T, y^T)^T, \quad s.t. \begin{cases} c_j^T x + d_j = y_j, \ j = 1, 2, \cdots, p, \\ y_{p+s} = y_{p+s-1}y_{p-s}, \ s = 1, 2, \cdots, p-1, \\ x \in X, y \in Y^0. \end{cases}$$

Theorem 1. *If x^* is the globally optimal solution of the problem (LMP), if and only if (x^*, y^*) is the globally optimal solution of the problem (EP), the equations $y_j^* = c_j^T x^* + d_j$, $j = 1, 2, \cdots, p$ and $y^*_{p+s} = y^*_{p+s-1}y^*_{p-s}$, $s = 1, 2, \cdots, p-1$ are also established.*

Proof of Theorem 1. If x^* is a globally optimal solution for problem (LMP), we have $\underline{y}_j \le y_j^* = c_j^T x^* + d_j \le \overline{y}_j$ and $y^*_{p+s} = y^*_{p+s-1}y^*_{p-s}$, $j = 1, 2, \cdots, p$, $s = 1, 2, \cdots, p-1$. Then, (x^*, y^*) is the feasible solution of (EP), and there is also the corresponding objective function value, that is,

$$h(x^*, y^*) = y^*_{2p-1} = \prod_{j=1}^{p} y_j^* = \prod_{j=1}^{p} c_j^T x^* + d_j = f(x^*).$$

Let (x, y) be any feasible solution of the problem (EP), and naturally there is the equation

$$\begin{aligned} & y_j = c_j^T x + d_j, j = 1, 2, \cdots, p, \\ & y_{p+s} = y_{p+s-1}y_{p-s}, s = 1, 2, \cdots, p-1, \end{aligned}$$

which also means

$$f(x) = \prod_{j=1}^{p} c_j^T x + d_j = \prod_{j=1}^{p} y_j = y_{2p-1} = h(x, y).$$

By using the optimal property of x^*,

$$y^*_{2p-1} = h(x^*, y^*) \leq h(x, y) = y_{2p-1}.$$

must be valid. Therefore, with $x^* \in X$ and $y^* \in Y$, a globally optimal solution of problem (EP) can be obtained, that is, (x^*, y^*).

On the other hand, we assume that (x^*, y^*) is a globally optimal solution of the (EP) problem, then

$$\begin{aligned} &y^*_j = c_j^T x^* + d_j,\ j = 1, 2, \cdots, p, \\ &y^*_{p+s} = y^*_{p+s-1} y^*_{p-s}, s = 1, 2, \cdots, p-1. \end{aligned}$$

and

$$h(x^*, y^*) = y^*_{2p-1} = \prod_{j=1}^{p} y^*_j = \prod_{j=1}^{p} c_j^T x^* + d_j = f(x^*).$$

must also hold. For the problem (LMP) arbitrary feasible solution x, if

$$\begin{aligned} &y_j = c_j^T x + d_j, j = 1, 2, \cdots, p, \\ &y_{p+s} = y_{p+s-1} y_{p-s}, s = 1, 2, \cdots, p-1, \end{aligned}$$

then (x, y) is a feasible solution of the (EP) problem and the objective function value is $h(x, y)$. Through the optimality of (x^*, y^*) and the feasibility of x, there is

$$h(x) = \prod_{j=1}^{p} c_j^T x + d_j = y_{2p-1} \geq y^*_{2p-1} = h(x^*, y^*) = f(x^*).$$

According to the above inequality, we can obtain that x^* is a globally optimal solution of the problem (LMP). The conclusion is completely proved. □

Combined with (1), we can notice that the objective function of the problem (EP) has such a property, that is, $h(x, y) = C^T(x^T, y^T)^T = y_{2p-1} = y_{2p-2} y_1 = \cdots = \prod_{j=1}^{p} y_j = \prod_{j=1}^{p}(c_j^T x + d_j) = f(x)$. The above property ensures that $h(x, y)$ in the following sections can be replaced by $f(x)$, that is, $h(x, y) = f(x)$.

Although the constraint $y_{p+s} = y_{p+s-1} y_{p-s},\ s = 1, 2, \cdots, p-1$ of (EP) is still nonlinear, we can also solve the problem (EP) with a branch-and-bound algorithm based on rectangular subdivisions of the set $Y^0 = \hat{Y}^0 \times \tilde{Y}^0$. Let $\Xi = \{Y^k : k = 1, 2, \cdots, K\}$ denote a rectangular partition of Y^0, i.e., $Y^k = \hat{Y}^k \times \tilde{Y}^k = \prod_{j=1}^{p}[\underline{y}_j^k, \overline{y}_j^k] \times \prod_{s=1}^{p-1}[\underline{y}_{p+s}^k, \overline{y}_{p+s}^k]$, $\bigcup_{k=1}^{K} Y^k = Y^0, \mathrm{int} Y^k \cap \mathrm{int} Y^z = \emptyset \text{if } k \neq z$. In the process of subdivision of rectangular $Y^k \in Y^0$, we must point out that the rectangle $\hat{Y}^k$ of the first part of Y^k is subdivided by the standard dichotomous method, and then $\tilde{Y}^k$ of the second half is compressed directly. This compression method is described in detail below. Our rectangular branch-and-bound algorithm systematically reduces the rectangular region containing y^* by repeating seven basic steps.

Output-Space Branch-and-Bound Reduction Procedure

Step 0. (*Initialization*) Set the tolerance $\epsilon > 0$. The initial upper bound UB^0, lower bound LB^0, best solution x^*, and (x^*, y^*) are obtained by solving the initial lower bound subproblem over Y^0.
Step 1. (*Termination criteria*) If $UB^k - LB^k \leq \epsilon$ or $\Xi = \emptyset$, then the algorithm is terminated, and the ϵ-globally optimal solution (x^*, y^*) and the ϵ-globally optimal value $h(x^*, y^*)$ of the (EP) problem are

output immediately, and the ϵ-globally optimal solution x^* and the ϵ-globally optimal value $f(x^*) = h(x^*, y^*)$ of (LMP) are also output immediately. Otherwise, go to Step 2.
Step 2. Select an appropriate Y^k from Ξ and set $\Xi = \Xi \backslash Y^k$, Y^k satisfies $LB^k < UB^k - \epsilon$ and (x^*, y^*) is the best feasible solution thus far.
Step 3. ($Branching\ operation$) Divide Y^k into two rectangles Y^{k1} and Y^{k2}.
Step 4. ($rectangle - reducing\ operation$) Using the rectangle-reduction technique, the two rectangles generated by $Step$ 3 are refined separately, and the index set of the remaining rectangle after the reduction is represented by Γ, obviously $|\Gamma| \le 2$.
Step 5. ($Bounding\ operation$) Compute the lower bound $LB(Y^{ki})(i = \Gamma)$ on the minimum value of h over $Y^{ki} \cap \{y \in R^{n+2p-1} : y_j = c_j^T x + d_j, y_{p+s} = y_{p+s-1} y_{p-s}, j = 1, 2, \cdots, p, s = 1, 2, \cdots, p-1, x \in X\}$. If $LB(Y^{ki}) < UB^k - \epsilon$ with $UB^k = h(x^*, y^*) = f(x^*)$, put Y^{ki} into Ξ, i.e., $\Xi = \Xi \cup Y^{ki}$.
Step 6. ($Updating\ the\ upper\ and\ lower\ bound$) The new feasible solution is used to update the upper bound. Thus far, the least optimal value of all known sub-problems is chosen as the new lower bound.

Obviously, Step 5 is the key to improve the efficiency of the algorithm. In the next subsection, we give a linear relaxation-subproblem of the problem (EP) to provide an efficient lower bound for the optimal value.

2.2. Novel Linear Relaxation Approach

An important operation of the branch-and-bound procedure for solving (EP) is to establish and solve a series of lower bound relaxer problems of (EP) on $Y^k \in \Xi = \{Y^k : k = 1, 2, \cdots, K\}$. We then define the associated sub-problems of (EP) as follows:

$$(EP^k) := \min h(x, y) = C^T (x^T, y^T)^T, \quad s.t. \left\{ \begin{array}{c} c_j^T x + d_j = y_j,\ j = 1, 2, \cdots, p, \\ y_{p+s} = y_{p+s-1} y_{p-s},\ s = 1, 2, \cdots, p-1, \\ x \in X, y \in Y^k. \end{array} \right.$$

The main idea of our relaxation method is to linearize the nonlinear constraint of (EP^k) and finally obtain its linear lower bound relaxation problem. Next, we give Theorem 2, which is associated with the proposed linearization-method.

Theorem 2. *Let $\Omega = \{(z, w) \in R^2 | -\infty \le \underline{z} \le z \le \overline{z} \le \infty, -\infty \le \underline{w} \le w \le \overline{w} \le \infty\}$. For any $\delta > 0$, define:*

$$\begin{aligned}
&\varphi(z) = z^2,\ \varphi^u(z) = (\underline{z} + \overline{z})z - \underline{z}\overline{z},\ \varphi^l(z) = (\underline{z} + \overline{z})z - \frac{(\underline{z} + \overline{z})^2}{4}, \\
&\Delta(z) = \varphi(z) - \varphi^l(z), \nabla(z) = \varphi^u(z) - \varphi(z),\ \psi(z, w) = zw, \\
&\psi^l(z, w) = \frac{1}{2}[(\overline{w} + \underline{w})z + (\overline{z} + \underline{z})w + \frac{(\underline{z} - \delta\overline{w})(\overline{z} - \delta\underline{w})}{2\delta} - \frac{(\overline{z} + \underline{z} + \delta(\overline{w} + \underline{w}))^2}{8\delta}], \\
&\psi^u(z, w) = \frac{1}{2}[(\overline{w} + \underline{w})z + (\overline{z} + \underline{z})w + \frac{(\overline{z} + \underline{z} - \delta(\overline{w} + \underline{w}))^2}{8\delta} - \frac{(\underline{z} + \delta\underline{w})(\overline{z} + \delta\overline{w})}{2\delta}], \\
&\Delta(z, w) = \psi(z, w) - \psi^l(z, w), \nabla(z, w) = \psi^u(z, w) - \psi(z, w).
\end{aligned}$$

Then, the following conclusion holds:
(a) $\varphi^l(z) \le \varphi(z) \le \varphi^u(z)$;
(b) $\psi^l(z, w) \le \psi(z, w) \le \psi^u(z, w)$; and
(c) $\Delta(z) \to 0$, $\nabla(z) \to 0$, $\Delta(z, w) \to 0$, $\nabla(z, w) \to 0$, as $\overline{z} - \underline{z} \to 0$, $\overline{w} - \underline{w} \to 0$.

Proof of Theorem 2. (a) Through the linear underestimation and overestimation functions defined by the single variable quadratic function $\varphi(z) = z^2$ over the interval $[\underline{z}, \overline{z}]$, we have

$$\varphi^u(z) = (\underline{z} + \overline{z})z - \underline{z}\overline{z} \geq \varphi(z) \geq \varphi^l(z) = (\underline{z} + \overline{z})z - \frac{(\underline{z} + \overline{z})^2}{4}, \tag{5}$$

which is $\varphi^l(y) \leq \varphi(y) \leq \varphi^u(y)$.

(b) Define

$$\begin{aligned}
&\varphi(z + \delta w) = (z + \delta w)^2, \varphi(z - \delta w) = (z - \delta w)^2,\\
&\varphi^l(z + \delta w) = [\underline{z} + \overline{z} + \delta(\underline{w} + \overline{w})](z + \delta w) - \frac{(\underline{z} + \overline{z} + \delta(\underline{w} + \overline{w}))^2}{4},\\
&\varphi^u(z + \delta w) = [\underline{z} + \overline{z} + \delta(\underline{w} + \overline{w})](z + \delta w) - (\underline{z} + \delta\underline{w})(\overline{z} + \delta\overline{w}),\\
&\varphi^l(z - \delta w) = [\underline{z} + \overline{z} - \delta(\underline{w} + \overline{w})](z - \delta w) - \frac{(\underline{z} + \overline{z} - \delta(\underline{w} + \overline{w}))^2}{4},\\
&\varphi^u(z - \delta w) = [\underline{z} + \overline{z} - \delta(\underline{w} + \overline{w})](z - \delta w) - (\overline{z} - \delta\underline{w})(\underline{z} - \delta\overline{w}),
\end{aligned}$$

Suppose that $z + \delta w$ and $z - \delta w$ are univariate, $\varphi(z + \delta w) = (z + \delta w)^2$ and $\varphi(z - \delta w) = (z - \delta w)^2$ are convex functions of variables $z + \delta w$ and $z - \delta w$ defined on interval $[\underline{z} + \delta\underline{w}, \overline{z} + \delta\overline{w}]$ and $[\underline{z} - \delta\overline{w}, \overline{z} - \delta\underline{w}]$, respectively. Then, using inequality (5), we have

$$\varphi^l(z + \delta w) \leq \varphi(z + \delta w) \leq \varphi^u(z + \delta w), \tag{6}$$

and

$$\varphi^l(z - \delta w) \leq \varphi(z - \delta w) \leq \varphi^u(z - \delta w). \tag{7}$$

By (5)–(7), we can find that

$$\begin{aligned}
\psi(z, w) &= \frac{1}{4\delta}[(z + \delta w)^2 - (z - \delta w)^2] = \frac{1}{4\delta}[\varphi(z + \delta w) - \varphi(z - \delta w)],\\
&\geq \frac{1}{4\delta}[(\overline{z} + \delta\overline{w} + \underline{z} + \delta\underline{w})(z + \delta w) - \frac{(\overline{z} + \delta\overline{w} + \underline{z} + \delta\underline{w})^2}{4} -\\
&((\overline{z} - \delta\underline{w} + \underline{z} - \delta\overline{w})(z - \delta w) - (\overline{z} - \delta\underline{w})(\underline{z} - \delta\overline{w}))],\\
&= \frac{1}{4\delta}[\varphi^l(z + \delta w) - \varphi^u(z - \delta w)],\\
&= \frac{1}{2}[(\overline{w} + \underline{w})z + (\overline{z} + \underline{z})w + \frac{(\underline{z} - \delta\overline{w})(\overline{z} - \delta\underline{w})}{2\delta} - \frac{(\overline{z} + \underline{z} + \delta(\overline{w} + \underline{w}))^2}{8\delta}],\\
&= \psi^l(z, w),
\end{aligned}$$

and

$$\begin{aligned}\psi(z,w) &= \frac{1}{4\delta}[(z+\delta w)^2-(z-\delta w)^2] = \frac{1}{4\delta}[\varphi(z+\delta w)-\varphi(z-\delta w)],\\ &\le \frac{1}{4\delta}[(\overline{z}+\delta\overline{w}+\underline{z}+\delta\underline{w})(z+\delta w)-(\overline{z}+\delta\overline{w})(\underline{z}+\delta\underline{w})-\\ &((\overline{z}-\delta\underline{w}+\underline{z}-\delta\overline{w})(z-\delta w)-\frac{(\overline{z}-\delta\underline{w}+\underline{z}-\delta\overline{w})^2}{4})],\\ &= \frac{1}{4\delta}[\varphi^u(z+\delta w)-\varphi^l(z-\delta w)],\\ &= \frac{1}{2}[(\overline{w}+\underline{w})z+(\overline{z}+\underline{z})w+\frac{(\overline{z}+\underline{z}-\delta(\overline{w}+\underline{w}))^2}{8\delta}-\frac{(\underline{z}+\delta\underline{w})(\overline{z}+\delta\overline{w})}{2\delta}],\\ &= \psi^u(z,w).\end{aligned}$$

Therefore, we have $\psi^l(z,w)\le\psi(z,w)\le\psi^u(z,w)$.

(c) Since function $\Delta(z)=\varphi(z)-\varphi^l(z)=z^2-(\underline{z}+\overline{z})z+\frac{(\underline{z}+\overline{z})^2}{4}$ is a convex function defined by variable z over the interval $[\underline{z},\overline{z}]$, the maximum value of function $\Delta(z)$ can reach at the point $\underline{z}$ or $\overline{z}$, that is,

$$0\le\Delta(z)=\varphi(z)-\varphi^l(z)\le\frac{(\overline{z}-\underline{z})^2}{4}=\max_{z\in[\underline{z},\overline{z}]}\Delta(z). \quad (8)$$

Similarly,

$$0\le\nabla(z)=\varphi^u(z)-\varphi(z)\le\frac{(\overline{z}-\underline{z})^2}{4}=\max_{z\in[\underline{z},\overline{z}]}\nabla(z). \quad (9)$$

By using (8) and (9), we have $\max_{z\in[\underline{z},\overline{z}]}\Delta(z)=\max_{z\in[\underline{z},\overline{z}]}\nabla(z)\to 0$, with $\overline{z}-\underline{z}\to 0$, and of course, $\Delta(z)\to 0, \nabla(z)\to 0$, with $\overline{z}-\underline{z}\to 0$.

Now, let us define

$$\begin{aligned}&\Delta(z+\delta w)=\varphi(z+\delta w)-\varphi^l(z+\delta w), \nabla(z+\delta w)=\varphi^u(z+\delta w)-\varphi(z+\delta w),\\ &\Delta(z-\delta w)=\varphi(z-\delta w)-\varphi^l(z-\delta w), \nabla(z-\delta w)=\varphi^u(z-\delta w)-\varphi(z-\delta w).\end{aligned}$$

Through Equation (8) and the definition of functions $\varphi(z+\delta w)$, $\varphi^l(z+\delta w)$, $\varphi^u(z-\delta w)$, and $\varphi(z-\delta w)$, we can draw the following conclusion:

$$\begin{aligned}\Delta(z,w) &= \psi(z,w)-\psi^l(z,w),\\ &= \frac{1}{4\delta}[(\varphi(z+\delta w)-\varphi^l(z+\delta w))+(\varphi^u(z-\delta w)-\varphi(z-\delta w))],\\ &= \frac{1}{4\delta}[\Delta(z+\delta w)+\nabla(z-\delta w)],\\ &\le \frac{1}{4\delta}[\max_{z+\delta w\in[\underline{z}+\delta\underline{w},\overline{z}+\delta\overline{w}]}\Delta(z+\delta w)+\max_{z-\delta w\in[\underline{z}+\delta\overline{w},\overline{z}+\delta\underline{w}]}\nabla(z-\delta w)],\\ &= \frac{(\overline{z}-\underline{z}+\delta(\overline{w}-\underline{w}))^2}{8\delta}=\max_{z\in[\underline{z},\overline{z}],w\in[\underline{w},\overline{w}]}\Delta(z,w).\end{aligned} \quad (10)$$

Similarly, through Equation (9) and the definition of functions $\varphi(z+\delta w)$, $\varphi^u(z+\delta w)$, $\varphi^l(z-\delta w)$, and $\varphi(z-\delta w)$, we also have

$$\nabla(z,w) = \frac{1}{4\delta}[\nabla(z+\delta w) + \Delta(z-\delta w)] \le \frac{(\overline{z}-\underline{z}+\delta(\overline{w}-\underline{w}))^2}{8\delta} = \max_{z\in[\underline{z},\overline{z}],w\in[\underline{w},\overline{w}]} \nabla(z,w). \tag{11}$$

Then, by using (10) and (11), we have $\max_{z\in[\underline{z},\overline{z}],w\in[\underline{w},\overline{w}]} \Delta(z,w) = \max_{z\in[\underline{z},\overline{z}],w\in[\underline{w},\overline{w}]} \nabla(z,w) \to 0$ with $\overline{z}-\underline{z} \to 0, \overline{w}-\underline{w} \to 0$, and, of course, $\Delta(z,w) \to 0$, $\nabla(z,w) \to 0$ with $\overline{z}-\underline{z} \to 0$, $\overline{w}-\underline{w} \to 0$. At this point, the conclusion is proved. □

It is noted that at the right most end of inequalities (10) and (11), the size of positive number δ has an effect on the upper and lower estimates of function $\psi(z,w)$. Let $a = \overline{z}-\underline{z}$, $b = \overline{w}-\underline{w}$, $g(\delta) = \frac{(a+\delta b)^2}{8\delta}$, obviously $a > 0$, $b > 0$. According to the derivative function $g'(\delta) = \frac{\delta^2 b^2 - a^2}{8\delta^2}$ of $g(\delta)$, we can know that, when $g(\delta)$ is defined on the interval $(0,+\infty)$, it is a convex function, and it is also easy to know that, if $g(\delta)$ gets the minimum value $\frac{(\overline{z}-\underline{z})(\overline{w}-\underline{w})}{2}$, then $\delta = \frac{a}{b} = \frac{\overline{z}-\underline{z}}{\overline{w}-\underline{w}}$. At this time, $\max_{z\in[\underline{z},\overline{z}],w\in[\underline{w},\overline{w}]} \Delta(z,w)$ and $\max_{z\in[\underline{z},\overline{z}],w\in[\underline{w},\overline{w}]} \Delta(z,w)$ take the minimum value $\frac{(\overline{z}-\underline{z})(\overline{w}-\underline{w})}{2}$, and the gap between $\psi(z,w)$ and its upper and lower estimation functions reaches the minimum value range, and the upper and lower estimation values are more stable.

For any sub-rectangle $Y^k \subseteq \Xi$ and each $s = 1,2,\cdots,p-1$, there is no loss of generality, with the following definition:

$$\psi_s(y_{p-s},y_{p+s-1}) = y_{p-s}y_{p+s-1} = \frac{(y_{p-s}+\delta_s^k y_{p+s-1})^2 - (y_{p-s}-\delta_s^k y_{p+s-1})^2}{4\delta_s^k}, \delta_s^k = \frac{\overline{y}_{p-s}^k - \underline{y}_{p-s}^k}{\overline{y}_{p+s-1}^k - \underline{y}_{p+s-1}^k},$$

$$\begin{aligned}\underline{\psi}_s^k(y_{p-s},y_{p+s-1}) = \frac{1}{2}[&(\overline{y}_{p+s-1}^k + \underline{y}_{p+s-1}^k)y_{p-s} + (\overline{y}_{p-s}^k + \underline{y}_{p-s}^k)y_{p+s-1} + \\ &\frac{(\underline{y}_{p-s}^k - \delta_s^k \overline{y}_{p+s-1}^k)(\overline{y}_{p-s}^k - \delta_s^k \underline{y}_{p+s-1}^k)}{2\delta_s^k} - \frac{(\overline{y}_{p-s}^k + \underline{y}_{p-s}^k + \delta_s^k(\overline{y}_{p+s-1}^k + \underline{y}_{p+s-1}^k))^2}{8\delta_s^k}],\end{aligned}$$

$$\begin{aligned}\overline{\psi}_s^k(y_{p-s},y_{p+s-1}) = \frac{1}{2}[&(\overline{y}_{p+s-1}^k + \underline{y}_{p+s-1}^k)y_{p-s} + (\overline{y}_{p-s}^k + \underline{y}_{p-s}^k)y_{p+s-1} + \\ &\frac{(\overline{y}_{p-s}^k + \underline{y}_{p-s}^k - \delta_s^k(\overline{y}_{p+s-1}^k + \underline{y}_{p+s-1}^k))^2}{8\delta_s^k} - \frac{(\underline{y}_{p-s}^k + \delta_s^k \underline{y}_{p+s-1}^k)(\overline{y}_{p-s}^k + \delta_s^k \overline{y}_{p+s-1}^k)}{2\delta_s^k}].\end{aligned}$$

Using Theorem 2, for each $s = 1,2,\cdots,p-1$, let $z = y_{p-s}$, $w = y_{p+s-1}$, the function $\psi_s(y_{p-s},y_{p+s-1})$, $\underline{\psi}_s^k(y_{p-s},y_{p+s-1})$ and $\overline{\psi}_s^k(y_{p-s},y_{p+s-1})$ also satisfies $0 \le \psi_s(y_{p-s},y_{p+s-1}) - \underline{\psi}_s^k(y_{p-s},y_{p+s-1}) \to 0$, $0 \le \overline{\psi}_s^k(y_{p-s},y_{p+s-1}) - \psi_s(y_{p-s},y_{p+s-1}) \to 0$, as $\overline{y}_{p-s}^k - \underline{y}_{p-s}^k \to 0, \overline{y}_{p+s-1}^k - \underline{y}_{p+s-1}^k \to 0$.

Theorem 3. *For any sub-rectangle $Y^k \subseteq \Xi$ and $y \in Y^k$, let $\varepsilon = \max\{\overline{y}_j^k - \underline{y}_j^k : j = 1,2,\cdots,p\}$, we have $y_{p+s} - \underline{\psi}_s^k(y_{p-s},y_{p+s-1}) \to 0$, $s = 1,2,\cdots,p-1$ as $\varepsilon \to 0$.*

Proof of Theorem 3. According to Theorem 2 and the definition of $\underline{\psi}_s^k(y_{p-s}, y_{p+s-1})$, we easily know that

$$0 \le y_{p+s} - \underline{\psi}_s^k(y_{p-s}, y_{p+s-1}) \le \frac{(\overline{y}_{p-s}^k - \underline{y}_{p-s}^k + \delta_s^k(\overline{y}_{p+s-1}^k - \underline{y}_{p+s-1}^k))^2}{8\delta_s^k} = \frac{|\overline{y}_{p-s}^k - \underline{y}_{p-s}^k| \cdot |\overline{y}_{p+s-1}^k - \underline{y}_{p+s-1}^k|}{2}. \tag{12}$$

If $s = 1$, the conclusion is obviously true, thus we only discuss the case of $s \in \{2, 3, \cdots, p-1\}$. For each $s = 2, 3, \cdots, p-1$, we have

$$\begin{aligned}
|y_{p+s} - \underline{y}_{p+s}^k| &= |y_{p-s}y_{p+s-1} - \underline{y}_{p-s}^k \underline{y}_{p+s-1}^k|, \\
&\le |y_{p-s}y_{p+s-1} - y_{p-s}\underline{y}_{p+s-1}^k| + |y_{p-s}\underline{y}_{p+s-1}^k - \underline{y}_{p-s}^k \underline{y}_{p+s-1}^k|, \\
&\le |y_{p-s}| \cdot |y_{p+s-1} - \underline{y}_{p+s-1}^k| + |y_{p-s} - \underline{y}_{p-s}^k| \cdot |\underline{y}_{p+s-1}^k|, \\
&\le |\overline{y}_{p-s}^k| \cdot |\overline{y}_{p+s-1}^k - \underline{y}_{p+s-1}^k| + |\overline{y}_{p-s}^k - \underline{y}_{p-s}^k| \cdot |\underline{y}_{p+s-1}^k|, \\
&\le |\overline{y}_{p-s}^0| \cdot |\overline{y}_{p+s-1}^k - \underline{y}_{p+s-1}^k| + |\overline{y}_{p-s}^k - \underline{y}_{p-s}^k| \cdot |\overline{y}_{p+s-1}^0|, \\
&= |\overline{y}_{p-s}^0| \cdot |\overline{y}_{p+s-1}^k - \underline{y}_{p+s-1}^k| + |\overline{y}_{p-s}^k - \underline{y}_{p-s}^k| \cdot \prod_{j=p-s+1}^{p} |\overline{y}_j^0|,
\end{aligned} \tag{13}$$

and

$$\begin{aligned}
|\overline{y}_{p+s}^k - y_{p+s}| &= |\overline{y}_{p-s}^k \overline{y}_{p+s-1}^k - y_{p-s}y_{p+s-1}|, \\
&\le |\overline{y}_{p-s}^k \overline{y}_{p+s-1}^k - \overline{y}_{p-s}^k y_{p+s-1}| + |\overline{y}_{p-s}^k y_{p+s-1} - y_{p-s}y_{p+s-1}|, \\
&\le |\overline{y}_{p-s}^k| \cdot |\overline{y}_{p+s-1}^k - y_{p+s-1}| + |\overline{y}_{p-s}^k - y_{p-s}| \cdot |y_{p+s-1}|, \\
&\le |\overline{y}_{p-s}^k| \cdot |\overline{y}_{p+s-1}^k - \underline{y}_{p+s-1}^k| + |\overline{y}_{p+s-1}^k| \cdot |\overline{y}_{p-s}^k - \underline{y}_{p-s}^k|, \\
&\le |\overline{y}_{p-s}^0| \cdot |\overline{y}_{p+s-1}^k - \underline{y}_{p+s-1}^k| + |\overline{y}_{p+s-1}^0| \cdot |\overline{y}_{p-s}^k - \underline{y}_{p-s}^k|, \\
&= |\overline{y}_{p-s}^0| \cdot |\overline{y}_{p+s-1}^k - \underline{y}_{p+s-1}^k| + |\overline{y}_{p-s}^k - \underline{y}_{p-s}^k| \cdot \prod_{j=p-s+1}^{p} |\overline{y}_j^0|.
\end{aligned} \tag{14}$$

Then, by using trigonometric inequalities to combine inequalities (13) and (14), we obtain the following recurrence formulas:

$$|\overline{y}_{p+s}^k - \underline{y}_{p+s}^k| \le |\overline{y}_{p+s}^k - y_{p+s}| + |y_{p+s} - \underline{y}_{p+s}^k| \le N_1^s \cdot |\overline{y}_{p+s-1}^k - \underline{y}_{p+s-1}^k| + N_2^s \cdot |\overline{y}_{p-s}^k - \underline{y}_{p-s}^k|, \tag{15}$$

where

$$N_1^s = 2|\overline{y}_{p-s}^0|,\ N_2^s = 2\prod_{j=p-s+1}^{p} |\overline{y}_j^0|, s = 2, 3, \cdots, p-1.$$

Furthermore, according to Equation (15), it is necessary to have

$$\begin{aligned}|\overline{y}^k_{p+s}-\underline{y}^k_{p+s}| &\le N_1^s\cdot|\overline{y}^k_{p+s-1}-\underline{y}^k_{p+s-1}|+N_2^s\cdot|\overline{y}^k_{p-s}-\underline{y}^k_{p-s}|,\\ &\le N_1^s\cdot[N_1^{s-1}\cdot|\overline{y}^k_{p+s-2}-\underline{y}^k_{p+s-2}|+N_2^{s-1}\cdot|\overline{y}^k_{p-s+1}-\underline{y}^k_{p-s+1}|]+N_2^s\cdot|\overline{y}^k_{p-s}-\underline{y}^k_{p-s}|,\\ &\le\cdots,\\ &\le\prod_{q=1}^{s}N_1^q\cdot|\overline{y}^k_p-\underline{y}^k_p|+\sum_{j=p-s+1}^{p-1}(\prod_{u=p+1-j}^{s}N_1^u\cdot N_2^{p-j})\cdot|\overline{y}^k_j-\underline{y}^k_j|+N_2^s\cdot|\overline{y}^k_{p-s}-\underline{y}^k_{p-s}|,\\ &\le N_s\cdot\sum_{j=p-s}^{p}|\overline{y}^k_j-\underline{y}^k_j|.\end{aligned}\tag{16}$$

where

$$N_s=\max\{\prod_{q=1}^{s}N_1^q,N_2^s,\prod_{u=p+1-j}^{s}N_1^u\cdot N_2^{p-j}:j=p-s+1,p-s+2,\cdots,p-1\}.$$

By combining Equations (12) and (16), we can deduce

$$\begin{aligned}y_{p+s}-\underline{\psi}^k_s(y_{p-s},y_{p+s-1})&\le\frac{1}{2}|\overline{y}^k_{p-s}-\underline{y}^k_{p-s}|\cdot|\overline{y}^k_{p+s-1}-\underline{y}^k_{p+s-1}|,\\ &\le\frac{1}{2}N_{s-1}\cdot|\overline{y}^k_{p-s}-\underline{y}^k_{p-s}|\cdot\sum_{j=p-s+1}^{p}|\overline{y}^k_j-\underline{y}^k_j|.\end{aligned}\tag{17}$$

It can be noted that, in the most right of inequality (17), when $s=2,3,\cdots,p-1$, there is $(p-s)\in\{1,2,\cdots,p-2\}$. Because $\varepsilon\to 0$ means $\overline{y}^k_j-\underline{y}^k_j\to 0$ for each $j=1,2,\cdots,p$, then the right side of inequality (17) tends to zero, and then $y_{p+s}-\underline{\psi}^k_s(y_{p-s},y_{p+s-1})\to 0, s=1,2,\cdots,p-1$. The proof is complete. □

Below, by using the above pre-given conclusions, the linear relaxation problem (LRP^k) is obtained by relaxing the nonlinear constraints of the equivalent problem (EP^k), which is expressed as follows:

$$(LRP^k):=\min\underline{f}^k(x,y)=C^T(x^T,y^T)^T,\quad s.t.\left\{\begin{array}{c}c_j^Tx+d_j=y_j,\ j=1,2,\cdots,p,\\ \underline{\psi}^k_s(y_{p-s},y_{p+s-1})\le y_{p+s},\ s=1,2,\cdots,p-1,\\ x\in X,y\in Y^k.\end{array}\right.$$

Of course, the relaxation subproblem (LRP^0) defined on the rectangle Y^0 is shown as follows:

$$(LRP^0):=\min\underline{f}^0(x,y)=C^T(x^T,y^T)^T,\quad s.t.\left\{\begin{array}{c}c_j^Tx+d_j=y_j,\ j=1,2,\cdots,p,\\ \underline{\psi}^0_s(y_{p-s},y_{p+s-1})\le y_{p+s},\ s=1,2,\cdots,p-1,\\ x\in X,y\in Y^0.\end{array}\right.$$

Theorem 3 shows that the feasible domain of the linear relaxation subproblem described above will gradually approximate the feasible domain of the equivalent problem (EP) as the algorithm gradually refines the first part $\hat{Y}^0$ of the hyper-rectangular Y^0.

There is a needless to say fact: if (LRP^k) is not feasible, then (EP^k) is also not feasible; otherwise, for any optimal solution $(x^k,\tilde{y}^k)$ of (LRP^k), $f(x^k)=h(x^k,y^k)\ge\underline{f}^k(x^k,\tilde{y}^k)$ is obvious. In particular, if $\tilde{y}^k$ and y^k are defined as $\tilde{y}^k=(\tilde{y}^k_1,\tilde{y}^k_2,\cdots,\tilde{y}^k_p,\tilde{y}^k_{p+1},\tilde{y}^k_{p+2},\cdots,\tilde{y}^k_{2p-1})^T$ and

$y^k = (y_1^k, y_2^k, \cdots, y_p^k, y_{p+1}^k, y_{p+2}^k, \cdots, y_{2p-1}^k)^T = (\tilde{y}_1^k, \tilde{y}_2^k, \cdots, \tilde{y}_p^k, y_{p+1}^k, y_{p+2}^k, \cdots, y_{2p-1}^k)^T$, then $y_{p+s}^k = y_{p+s-1}^k \tilde{y}_{p-s}^k, s = 1, 2, \cdots, p-1$.

Finally, our bounding operation of the branch-and-bound procedure can be expressed as:
Step 5.(*Bounding operation*) Set $LB(Y^k) := \underline{f}^k(x^k, y^k)$. If $LB(Y^k) \le UB^k - \epsilon$ with $UB^k = h(x^*, y^*) = f(x^*)$, put Y^k into Ξ, i.e., $\Xi = \Xi \cup Y^k$.

Remark 1. *In this paper, we obtain the lower bound relaxation problem (LRP^k) by using $\underline{\psi}_s^k(y_{p-s}, y_{p+s-1}) \le y_{p+s}$ to relax the feasible domain of the equivalence problem (EP^k). If we solve the linear maximum multiplicative programming problem, we only need to use $\overline{\psi}_s^k(y_{p-s}, y_{p+s-1}) \ge y_{p+s}$ to do similar upper bound relaxation.*

Remark 2. *If a linear function is added to the objective function of the problem (LMP), then there is a similar equivalent transformation and proof to Section 2.2, which is because we only make the equivalent transformation of the product term of the objective function. In this section, of course, there is a similar relaxation subproblem, and the rectangular partition method in the next subsection is similar, but then the reduction method of the hyper-rectangle is a little different, and we give it after Section 2.4.*

2.3. Subdivision and Refinement of Hyper-Rectangle

Branching operation is also indispensable in Branch-and-bound procedure. In this subsection, we give the branch-refinement rule of any $Y^k = [\underline{y}^k, \overline{y}^k] \in \Xi$, where $Y^k = \hat{Y}^k \times \tilde{Y}^k$.

According to Equations (1) and (4) and $y_{p+s} = y_{p+s-1} y_{p-s},\ s \in \{1, 2, \cdots, p-1\}$, the generation of rectangular $\tilde{Y}^k$ mainly depends on the successive multiplication of the coordinate components of the lower left vertex and the upper right vertex of $\hat{Y}^k$. Therefore, we only use the standard dichotomy to segment the former part $\hat{Y}^k$ of the sub-rectangle Y^k, and then refine the latter part $\tilde{Y}^k$ according to Equation (4). The specific operations of Step 3 are as follows:

(i) For the rectangle $\hat{Y}^k = \prod_{j=1}^{p} [\underline{y}_j^k, \overline{y}_j^k]$, let $\overline{y}_\mu^k - \underline{y}_\mu^k = \max\{\overline{y}_j^k - \underline{y}_j^k : j = 1, 2, \cdots, p\}$, $y_\mu^k = \frac{\underline{y}_\mu^k + \overline{y}_\mu^k}{2}$. By using y_μ^k, the interval $[\underline{y}_\mu^k, \overline{y}_\mu^k]$ corresponding to the μ-edge of rectangle $\hat{Y}^k$ is divided into two intervals $[\underline{y}_\mu^k, y_\mu^k]$ and $[y_\mu^k, \overline{y}_\mu^k]$, and then $\hat{Y}^k$ is also divided into two sub-rectangles $\hat{Y}^{k1}$ and $\hat{Y}^{k2}$. Their forms can be expressed as

$$\hat{Y}^{k1} = \prod_{j=1}^{\mu-1} [\underline{y}_j^k, \overline{y}_j^k] \times [\underline{y}_\mu^k, y_\mu^k] \times \prod_{j=\mu+1}^{p} [\underline{y}_j^k, \overline{y}_j^k] = \prod_{j=1}^{p} [\underline{y}_j^{k1}, \overline{y}_j^{k1}],$$
$$\hat{Y}^{k2} = \prod_{j=1}^{\mu-1} [\underline{y}_j^k, \overline{y}_j^k] \times [y_\mu^k, \overline{y}_\mu^k] \times \prod_{j=\mu+1}^{p} [\underline{y}_j^k, \overline{y}_j^k] = \prod_{j=1}^{p} [\underline{y}_j^{k2}, \overline{y}_j^{k2}].$$

by the Cartesian product.

(ii) For the segmentation and thinning of hyper-rectangle $\tilde{Y}^k$, the upper right vertex of $\hat{Y}^{k1}$ and the lower left vertex of $\hat{Y}^{k2}$ are used, respectively. In this way, we finally get two hyper-rectangles, $\tilde{Y}^{k1}$ and $\tilde{Y}^{k2}$. According to $y_{p+s} = y_{p+s-1} y_{p-s},\ s \in \{1, 2, \cdots, p-1\}$, the Cartesian product forms of hyper-rectangle $\tilde{Y}^{k1}$ and $\tilde{Y}^{k2}$ are

$$\tilde{Y}^{k1} = \prod_{s=1}^{p-1} [\underline{y}_{p+s}^k, \min\{\overline{y}_{p+s}^k, \overline{y}_{p+s-1}^{k1} \overline{y}_{p-s}^{k1}\}] = \prod_{s=1}^{p-1} [\underline{y}_{p+s}^{k1}, \overline{y}_{p+s}^{k1}],$$
$$\tilde{Y}^{k2} = \prod_{s=1}^{p-1} [\max\{\underline{y}_{p+s-1}^k, \underline{y}_{p+s-1}^{k2} \underline{y}_{p-s}^{k2}\}, \overline{y}_{p+s}^k] = \prod_{s=1}^{p-1} [\underline{y}_{p+s}^{k2}, \overline{y}_{p+s}^{k2}].$$

Obviously, $Y^{k1} \cap Y^{k2} = \emptyset$.

Although Y^k is a hyper-rectangular space of $2p-1-$dimension, it can be seen from the Branch-refinement method of hyper-rectangle Y^k mentioned above that we only branch the rectangle $\hat{Y}^k$, and the boundary of $\tilde{Y}^{k1}(\tilde{Y}^{k2})$ can be obtained directly according to the boundary of $\hat{Y}^{k1}(\hat{Y}^{k2})$, thus the branching process of the branch-and-bound algorithm is completed in $p-$dimensional space $\hat{Y}^k$.

2.4. Reduction of the Hyper-Rectangle

In this subsection, we give a reduction technique for hyper-rectangles to delete the sub-rectangle Y^k that do not contain the globally optimal solution or to delete a part of the sub-rectangle Y^k that do not have a globally optimal solution. In this way, the number of rectangles in the set Ξ will be reduced or the effect of refinement of rectangles in Ξ will be achieved, and then the bounding operation will be accelerated.

Without losing the generality, it is assumed that the current hyper-rectangle to be reduced is $Y^k = \hat{Y}^k \times \tilde{Y}^k = \prod_{j=1}^{p}[\underline{y}_j^k, \overline{y}_j^k] \times \prod_{s=1}^{p-1}[\underline{y}_{p+s}^k, \overline{y}_{p+s}^k] \in \Xi$, and the best objective function value obtained by the algorithm, thus far, is UB^k. Because $y_{2p-1} = \prod_{j=1}^{p} y_j \le UB^k$, for each $t = 1, 2, \cdots, p$, $v = 1, 2, \cdots, p-2$, define

$$\gamma^k = \min\{UB^k, \overline{y}_{2p-1}^k\}, \alpha_t^k = \frac{\gamma^k}{\prod\limits_{j=1, j\neq t}^{p} \underline{y}_j^k}, \beta_0^k = \gamma^k, \beta_v^k = \frac{\beta_{v-1}^k}{\underline{y}_v^k}.$$

It is easy to know that, if the problem (EP) has a globally optimal solution in the rectangular Y^k, there must be a necessary condition, that is, $\underline{y}_{2p-1}^k \le y_{2p-1} \le \gamma^k$. This necessary condition is also used in the following two rectangular reduction theorems.

In view of the characteristics that the super rectangle Y^k consists of two parts $\hat{Y}^k$ and $\tilde{Y}^k$, we reduce it in two steps. For this reason, we give Theorems 4 and 5, and prove that they have set forth the super-rectangular reduction technique in this section.

Theorem 4. *For each $t = 1, 2, \cdots, p$, if $\alpha_t^k < \underline{y}_t^k$, the original problem (EP) has no globally optimal solution on the rectangle Y^k; otherwise, if $\alpha_t^k < \overline{y}_t^k$, the rectangle Y^{k1} does not contain the globally optimal solution of the problem (EP), where*

$$Y^{k1} = \hat{Y}^{k1} \times \tilde{Y}^k \subseteq Y^k \,\&\&\, \hat{Y}_j^{k1} = \begin{cases} \hat{Y}_j^k, j \neq t, \\ (\alpha_t^k, \overline{y}_t^k] \cap \hat{Y}_j^k, j = t. \end{cases}$$

Proof of Theorem 4. If there is a $t \in \{1, 2, \cdots, p\}$ that satisfies $\alpha_t^k < \underline{y}_t^k$, there will be

$$\gamma^k = \min\{UB^k, \overline{y}_{2p-1}^k\} = \alpha^k \prod_{j=1, j\neq t}^{p} \underline{y}_j^k < \prod_{j=1}^{p} \underline{y}_j^k \le \prod_{j=1}^{p} y_j = y_{2p-1},$$

then, there is no globally optimal solution of (EP) on Y^k. Next, we prove that there is $\gamma^k < y_{2p-1}$ for each $y \in Y^{k1}$. When $y \in Y^{k1}$, we consider the tth element y_t of y, because $y_t \in (\alpha_t^k, \overline{y}_t^k] \cap \hat{Y}_t^k$, we have

$$\alpha_t^k < y_t \le \overline{y}_t^k, t = 1, 2, \cdots, p.$$

According to the definition of α_t^k and the above inequality, we also have

$$\gamma^k = \min\{UB^k, \overline{y}_{2p-1}^k\} = \alpha^k \prod_{j=1, j\neq t}^{p} \underline{y}_j^k < y_t \prod_{j=1, j\neq t}^{p} \underline{y}_j^k < \prod_{j=1}^{p} y_j = y_{2p-1}.$$

This means that, for all $y \in Y^{k1}$, there is $\gamma^k < y_{2p-1}$. Therefore, there is no globally optimal solution for the problem (EP). $\square$

To facilitate the description of Theorem 5, we still record the hyper-rectangle reduced by Theorem 4 as $Y^k = \hat{Y}^k \times \tilde{Y}^k = \prod_{j=1}^{p}[\underline{y}_j^k, \overline{y}_j^k] \times \prod_{s=1}^{p-1}[\underline{y}_{p+s}^k, \overline{y}_{p+s}^k] \subseteq Y^0$. It can be seen that the second part of Y^k does not change, and Theorem 5 is given below to reduce $\tilde{Y}^k$.

Theorem 5. *For each $v = 1, 2, \cdots, p-1$, if $\beta_{p-v-1}^k < \overline{y}_{p+v}^k$, the problem (EP) has no globally optimal solution on the hyper-rectangle Y^{k2}, where*

$$Y^{k2} = \hat{Y}^k \times \tilde{Y}^{k2} \subseteq Y^k \;\&\&\; \tilde{Y}_s^{k2} = \begin{cases} \tilde{Y}_s^k, s \neq v, \\ (\beta_{p-v-1}^k, \overline{y}_{p+v}^k] \cap \tilde{Y}_s^k, s = v. \end{cases}$$

Proof of Theorem 5. First, according to the definition of β_v^k, we have

$$\beta_v^k = \frac{\beta_{v-1}^k}{\underline{y}_v^k} = \frac{\beta_{v-2}^k}{\underline{y}_v^k \underline{y}_{v-1}^k} = \cdots = \frac{\beta_1^k}{\underline{y}_v^k \underline{y}_{v-1}^k \cdots \underline{y}_2^k} = \frac{\beta_0^k}{\prod_{l=1}^{v} \underline{y}_l^k} = \frac{\gamma^k}{\prod_{l=1}^{v} \underline{y}_l^k}.$$

Second, we prove that, for any $y \in Y^{k2}$, there is $\gamma^k < y_{2p-1}$. If $v = p-1$, obviously, $y_{p+v} = y_{2p-1} > \beta_0^k = \gamma^k$; Then, Y^{k2} does not contain the globally optimal solution of the problem (EP). If $v \in \{1, 2, \cdots, p-2\}$ exists and $\beta_{p-v-1}^k < \overline{y}_{p+v}^k$ is satisfied, we continue to consider the $(p+v)$th element y_{p+v} of y. If $y_{p+v} \in (\beta_{p-v-1}^k, \overline{y}_{p+v}^k] \cap \tilde{Y}_v^k$, then $\beta_{p-v-1}^k < y_{p+v} \leq \overline{y}_{p+v}^k$. Because $\beta_{p-v-1}^k = \frac{\gamma^k}{\prod_{l=1}^{p-v-1} \underline{y}_l^k}$, which means that, for all $y \in Y^{k2}$, there is

$$\gamma^k = \beta_{p-v-1}^k \prod_{l=1}^{p-v-1} \underline{y}_l^k < y_{p+v} \prod_{l=1}^{p-v-1} y_l^k = y_{p+v-1} \prod_{l=1}^{p-v} y_l^k = \cdots = y_p \prod_{l=1}^{p-1} y_l^k = y_{2p-1}.$$

Therefore, there is no globally optimal solution for the original problem (EP) on the hyper-rectangular Y^{k2}. $\square$

According to Theorems 4 and 5, we can construct the following reduction techniques to reduce the hyper-rectangle Y^k, which makes the compressed hyper-rectangle thinner and removes the part of the hyper-rectangle Y^k that does not contain the globally optimal solution, so that the search-space required for the algorithm to solve the problem (EP) is reduced, thus speeding up the convergence of the algorithm.
$Step\ 4.(rectangle - reducing\ operation)$

(i) For each $t = 1, 2, \cdots, p$, if $\alpha_t^k < \underline{y}_t^k$, let $Y^k = \emptyset$. Otherwise, if $\alpha_t^k < \overline{y}_t^k$, let $\overline{y}_t^k = \alpha_t^k$.

(ii) For each $v = 1, 2, \cdots, p-1$, if $\beta_{p-v-1}^k < \overline{y}_{p+v}^k$, let $\overline{y}_{p+v}^k = \beta_{p-v-1}^k$.

In addition, if the objective function of the problem (LMP) contains an additional linear function, then, to make the above reduction method valid, we need to make an adjustment to UB^k because it is affected by the additional linear term. Assuming that this additional linear term is $e^T x + f$, then, before the algorithm begins, we need to solve a linear programming problem $\xi = \min_{x \in X} e^T x + f$, so that there is $UB^k = UB^k - \xi$. Of course, e is an n-dimensional column vector while f is a constant. Obviously, the adjusted UB^k also satisfies the conditions of the above two theorem.

2.5. Output-Space Branch-and-Bound Reduction Algorithm

In this subsection, we combine the output-space branch-and-bound reduction procedure with the bounding operation, branching operation, and rectangle-reducing operation, and then construct a new deterministic global optimization algorithm for solving the problem (EP), namely the Output-Space Branch-and-Bound Reduction Algorithm ($OSBBRA$).

To describe the algorithm smoothly, we explain the relevant symbols of the algorithm iteration to step k as follows: Y^k is the hyper-rectangle to be subdivided in the current iteration step; Θ is the set of feasible solutions stored in the current iteration step of the problem (EP); Ξ is the set of sub-rectangles remaining after the pruning step; UB^k is the upper bound of the global optimal value of the problem (EP) in the current iteration step; LB^k is the lower bound of the globally optimal value of the problem (EP); and $LB(Y^k)$ and (x, y) represent the optimal value and solution of the subproblem (LRP^k) on the rectangle Y^k, respectively. In addition, any feasible point x of (LMP) must have $(x, \ddot{y})$, with $\ddot{y}^k = (\ddot{y}_1^k, \ddot{y}_2^k, \cdots, \ddot{y}_p^k, \ddot{y}_{p+1}^k, \ddot{y}_{p+2}^k, \cdots, \ddot{y}_{2p-1}^k)^T$, $\ddot{y}_j^k = c_j^T x + d_j, j = 1, 2, \cdots, p$, and $\ddot{y}_{p+s}^k = \ddot{y}_{p+s-1}^k \ddot{y}_{p-s}^k, s = 1, 2, \cdots, p-1$. being a feasible point for (EP). The specific steps of algorithm ($OSBBRA$) are as follows:

Step 0. ($Initialization$) Set the tolerance $\epsilon > 0$. The initial hyper-rectangular Y^0 is constructed by using Equations (2) and (3), whereas solving each feasible solution x of the (LMP) obtained from the linear programming problem (2) corresponds to one feasible solution $(x, \ddot{y})$ of (EP), and then stores all such feasible solutions of (EP) into the set Θ. Solve the initial subproblem LRP^0 on hyper-rectangular Y^0. Then, the optimal value and solution corresponding to the initial subproblem are $L(Y^0)$ and (x^0, y^0), respectively. Let $\Theta = \Theta \cup \{(x^0, \ddot{y}^0)\}$. Thus, $LB^0 = L(Y^0)$ can be used as the initial lower bound of the globally optimal value of the problem (EP). The initial upper bound is $UB^0 = \min\{h(x, y) : (x, y) \in \Theta\}$. The initial best solution to the original problem (EP) is $(x^*, y^*) = \arg UB^0$. If $UB^0 - LB^0 \leq \epsilon$, then stop, and the ϵ-globally optimal solution of the problem (EP) is (x^*, y^*). Otherwise, set $\Xi = \{Y^0\}$, $H = \emptyset$, $\Theta = \emptyset$, the iteration number $k = 1$, and go to Step 2.

Step 1. (*Termination criteria*) If $UB^k - LB^k \leq \epsilon$ or $\Xi = \emptyset$, then the algorithm is terminated, and the ϵ-globally optimal solution (x^*, y^*) and the ϵ-globally optimal value $h(x^*, y^*)$ of the (EP) problem are output immediately, and the ϵ-globally optimal solution x^* and the ϵ-globally optimal value $f(x^*) = h(x^*, y^*)$ of (LMP) are also output immediately. Otherwise, go to Step 2.

Step 2. According to $LB^k = LB(Y^k)$, select the sub-rectangle Y^k from the set Ξ and set $\Xi = \Xi \backslash Y^k$, and then go to Step 3.

Step 3. (*Branching operation*) By using the subdivision and refinement methods in Section 2.3, Y^k is divided into two sub-rectangles: Y^{k1} and Y^{k2} that satisfy $Y^{k1} \cap Y^{k2} = \emptyset$. Then, go to Step 4.

Step 4. ($rectangle - reducing\ operation$) Through the reduction in Section 2.4, we compress the two sub-rectangles Y^{k1} and Y^{k2} obtained in the previous iteration, and the index set of the remaining sub-rectangles after compression is expressed as Γ. Obviously, $|\Gamma| \leq 2$.

Step 5. (*Bounding operation*) For any $LB(Y^{ki}) < UB^k - \epsilon (i \in \Gamma)$, let $H = H \cup \{Y^{ki} : i \in \Gamma\}$, $\Theta = \Theta \cup \{(x^i, \ddot{y}^i)\} (i \in \Gamma)$. If $H = \emptyset$ and $\Xi \neq \emptyset$, return to Step 2. Else, if $H = \emptyset$ and $\Xi = \emptyset$, return to Step 1. Else, set $\Xi = \Xi \cup H$.

Step 6. (*Updating the upper and lower bound*) Let $U = \min\{UB^k, h(x, y) : (x, y) \in \Theta\}$. If $U \neq UB^k$, update the current best solution to $(x^*, y^*) \in \arg\min\{h(x, y) : (x, y) \in \Theta\}$ and set $UB^k = U$. Let $LB^k = \min\{LB(Y) : Y \in \Xi\}$; Set $k := k + 1$, $H = \emptyset$, $\Theta = \emptyset$, and return to Step 1.

Remark 3. *In Step 5, we save the super-rectangle Y^{ki} of $LB(Y^{ki}) < UB^k - \epsilon$ into Ξ after each compression, which implies the pruning operation of the algorithm.*

Remark 4. *As can be seen from Steps 4–6, the number of elements in Θ does not exceed two in each algorithm loop. At the same time, the phase of updating the upper bound in Step 5 computes at most two function values.*

Remark 5. *The branch search space of our OSBBRA algorithm is p-dimensional. When p is much smaller than the dimension n of decision variables, the convergence rate of the algorithm is relatively faster than that of n-dimensional decision space search.*

3. Analysis of the Computational Complexity of the Algorithm

In this subsection, we deduce the maximum number of iterations of the proposed algorithm by analyzing the computational complexity of the algorithm. For this reason, for the convenience of narration, we first define the longest edge of the first part of the rectangle $Y^k = \hat{Y}^k \times \tilde{Y}^k \subseteq Y^0$, that is, the longest edge of

$$\hat{Y}^k = \prod_{j=1}^{p} [\underline{y}_j^k, \overline{y}_j^k], \tag{18}$$

using

$$\triangle(\hat{Y}^k) = \max\{\overline{y}_j^k - \underline{y}_j^k : j = 1, 2, \cdots, p\}. \tag{19}$$

In addition, we also define

$$h_s = \frac{1}{2} |\overline{y}_{p-s}^k - \underline{y}_{p-s}^k| \cdot |\overline{y}_{p+s-1}^k - \underline{y}_{p+s-1}^k|, s = 1, 2, \cdots, p-1, \tag{20}$$

$$\varsigma = \frac{\sum_{s=2}^{p-1} s \cdot N_{s-1} + 1}{2}, \tag{21}$$

$$\varrho = \max\{1, \prod_{j=1}^{s} |\overline{y}_j^0| : s = 1, 2, \cdots, p-2\}, \tag{22}$$

where $\overline{y}_j^0$ is given by (2).

Lemma 1. *For the given convergence tolerance $\epsilon \geq 0$, if there is a rectangle $Y^k = \hat{Y}^k \times \tilde{Y}^k$ satisfying $\triangle(\hat{Y}^k) \leq \sqrt{\frac{\epsilon}{\varrho\varsigma}}$ when the algorithm is running to the kth cycle, then we have*

$$UB - LB(Y^k) \leq \epsilon,$$

where $LB(Y^k)$ is the optimal value of the problem (LRP^k) and UB denotes the current best upper bound of the equivalent problem (EP).

Proof of Lemma 1. If (x^k, y^k) is assumed to be the optimal solution of the linear relaxation problem (LRP^k), obviously $y_j^k = c_j^T x^k + d_j$, $j = 1, 2, \cdots, p$, in addition, let

$$\widetilde{y}_p^k = y_p^k, \ \widetilde{y}_{p+s}^k = \widetilde{y}_{p+s-1}^k y_{p-s}^k, s = 1, 2, \cdots, p-1,$$

and

$$\widetilde{y}^k = (y_1^k, y_2^k, \cdots, y_j^p, \widetilde{y}_{p+1}^k, \widetilde{y}_{p+2}^k, \cdots, \widetilde{y}_{2p-1}^k)^T,$$

then $(x^k, \widetilde{y}^k)$ is a feasible solution of the equivalent problem (EP^k). By using the definitions of $LB(Y^k)$ and UB, we have

$$\underline{f}^k(x^k, y^k) = LB(Y^k) \le UB \le h(x^k, \widetilde{y}^k) = f(x^k),$$

Therefore, from Equations (1) and (18)–(22), there is the following

$$\begin{aligned}
UB - LB(Y^k) &\le f(x^k) - \underline{f}^k(x^k, y^k), \\
&= h(x^k, \widetilde{y}^k) - \underline{f}^k(x^k, y^k), \\
&= \widetilde{y}_{2p-1}^k - y_{2p-1}^k, \\
&\le \widetilde{y}_{2p-1}^k - \underline{\psi}_{p-1}^k(y_1, y_{2p-2}), \\
&= (\widetilde{y}_{2p-2}^k y_1^k - y_{2p-2}^k y_1^k) + (y_{2p-2}^k y_1^k - \underline{\psi}_{p-1}^k(y_1, y_{2p-2})), \\
&\le y_1^k(\widetilde{y}_{2p-2}^k - y_{2p-2}^k) + \frac{1}{2}|\overline{y}_1^k - \underline{y}_1^k| \cdot |\overline{y}_{2p-2}^k - \underline{y}_{2p-2}^k|, \\
&\le \overline{y}_1^0(\widetilde{y}_{2p-2}^k - y_{2p-2}^k) + h_{p-1}, \\
&\le \overline{y}_1^0\overline{y}_2^0(\widetilde{y}_{2p-3}^k - y_{2p-3}^k) + \overline{y}_1^0 h_{p-2} + h_{p-1}, \\
&\le \cdots, \\
&\le \prod_{j=1}^{p-2}\overline{y}_j^0 h_1 + \prod_{j=1}^{p-3}\overline{y}_j^0 h_2 + \cdots + \prod_{j=1}^{2}\overline{y}_j^0 h_{p-3} + \overline{y}_1^0 h_{p-2} + h_{p-1}, \\
&\le \varrho \sum_{s=1}^{p-1} h_s.
\end{aligned}$$

Then, by using (17), we have

$$\begin{aligned}
UB - LB(Y^k) &\le \varrho \sum_{s=1}^{p-1} h_s, \\
&= \varrho \sum_{s=1}^{p-1}(\frac{1}{2}|\overline{y}_{p-s}^k - \underline{y}_{p-s}^k| \cdot |\overline{y}_{p+s-1}^k - \underline{y}_{p+s-1}^k|), \\
&= \frac{\varrho}{2}[\sum_{s=2}^{p-1}(|\overline{y}_{p-s}^k - \underline{y}_{p-s}^k| \cdot |\overline{y}_{p+s-1}^k - \underline{y}_{p+s-1}^k|) + (|\overline{y}_{p-1}^k - \underline{y}_{p-1}^k| \cdot |\overline{y}_p^k - \underline{y}_p^k|)], \\
&\le \frac{\varrho}{2}[\sum_{s=2}^{p-1}(|\overline{y}_{p-s}^k - \underline{y}_{p-s}^k| \cdot N_{s-1} \cdot \sum_{j=p-s+1}^{p}|\overline{y}_j^k - \underline{y}_j^k|) + (|\overline{y}_{p-1}^k - \underline{y}_{p-1}^k| \cdot |\overline{y}_p^k - \underline{y}_p^k|)], \\
&\le \frac{\varrho}{2}(\triangle(\hat{Y}^k))^2(\sum_{s=2}^{p-1} s \cdot N_{s-1} + 1), \\
&= \varrho\varsigma(\triangle(\hat{Y}^k))^2.
\end{aligned}$$

Thus, according to the above inequality and combined with $\triangle(\hat{Y}^k) \le \sqrt{\frac{\epsilon}{\varrho\varsigma}}$, we can obtain that

$$UB - LB(Y^k) \le \varrho\varsigma(\triangle(\hat{Y}^k))^2 \le \epsilon.$$

Finally, the proof of Lemma 1 is completed. □

On the premise of Lemma 1, if the $\triangle(\hat{Y}^k) \leq \sqrt{\frac{\epsilon}{\varrho\varsigma}}$ is satisfied, then the sub-rectangular Y^k can be deleted. Thus, according to Step 5 of the algorithm, it can be appreciated that when each sub-rectangular Y^k obtained by the refinement of Y^0 satisfies $\triangle(\hat{Y}^k) \leq \sqrt{\frac{\epsilon}{\varrho\varsigma}}$, the algorithm terminates the iteration. Therefore, we can derive the maximum number of iterations of the algorithm through Lemma 1. Theorem 6 gives a specific process.

Theorem 6. *For the given convergence tolerance $\epsilon \geq 0$, the maximum number of iterations required by the algorithm OSBBRA to obtain the $\epsilon-$globally optimal solution of the problem (LMP) is*

$$N = 2^{\sum\limits_{j=1}^{p} \lceil \log_2 \frac{\sqrt{\varrho\varsigma}(\overline{y}_j^0 - \underline{y}_j^0)}{\sqrt{\epsilon}} \rceil} - 1,$$

where ς and ϱ are given by (21) and (22), respectively. In addition, the definition of $\hat{Y}^0 = \prod\limits_{j=1}^{p} \hat{Y}_j^0$ with $\hat{Y}_j^0 = [\underline{y}_j^0, \overline{y}_j^0]$ is given by (2).

Proof of Theorem 6. It is assumed that $Y = \hat{Y} \times \tilde{Y}$ with $\hat{Y} = \prod\limits_{j=1}^{p} \hat{Y}_j = \prod\limits_{j=1}^{p} [\underline{y}_j, \overline{y}_j]$ is the rectangle that the algorithm is selected from Ξ when it is cycled to a certain Step 3. It is supposed that after k_j iterations, there is a subinterval $\hat{Y}_j^{k_j} = [\underline{y}_j^{k_j}, \overline{y}_j^{k_j}]$ of $\hat{Y}_j^0 = [\underline{y}_j^0, \overline{y}_j^0]$ satisfying

$$\overline{y}_j^{k_j} - \underline{y}_j^{k_j} \leq \sqrt{\frac{\epsilon}{\varrho\varsigma}},\ j = 1, 2, \cdots, p. \tag{23}$$

Let us consider the branching process of Step 4, and then there is

$$\overline{y}_j^{k_j} - \underline{y}_j^{k_j} = \frac{1}{2^{k_j}}(\overline{y}_j^0 - \underline{y}_j^0),\ j = 1, 2, \cdots, p. \tag{24}$$

By combining (23) with (24), we have

$$\frac{1}{2^{k_j}}(\overline{y}_j^0 - \underline{y}_j^0) \leq \sqrt{\frac{\epsilon}{\varrho\varsigma}},\ j = 1, 2, \cdots, p.$$

That is,

$$k_j \geq \log_2 \frac{\sqrt{\varrho\varsigma}(\overline{y}_j^0 - \underline{y}_j^0)}{\sqrt{\epsilon}},\ j = 1, 2, \cdots, p.$$

Let

$$\overline{k}_j = \lceil \log_2 \frac{\sqrt{\varrho\varsigma}(\overline{y}_j^0 - \underline{y}_j^0)}{\sqrt{\epsilon}} \rceil,\ j = 1, 2, \cdots, p.$$

Then, after $K_1 = \sum\limits_{j=1}^{p} \overline{k}_j$ iterations, the algorithm will generate at most $K_1 + 1$ rectangles, denoted by $Y^1, Y^2, \cdots, Y^{K_1+1}$, and they must all satisfy

$$\triangle(\hat{Y}^t) = 2^{K_1 - t}\triangle(\hat{Y}^{K_1}) = 2^{K_1 - t}\triangle(\hat{Y}^{K_1+1}), t = K_1, K_1 - 1, \cdots, 2, 1, \tag{25}$$

where $\triangle(\hat{Y}^{K_1}) = \triangle(\hat{Y}^{K_1+1}) = \max\{\overline{y}_j^{\bar{k}_j} - \underline{y}_j^{\bar{k}_j} : j = 1, 2, \cdots, p\}$ and

$$\hat{Y}^0 = \bigcup_{t=1}^{K_1+1} \hat{Y}^t. \tag{26}$$

Now, put the $K_1 + 1$ rectangles into the set Ξ^{K_1+1}, that is,

$$\Xi^{K_1+1} = \{Y^t : t = 1, 2, \cdots, K_1 + 1\},$$

and, according to (23), we have

$$\triangle(\hat{Y}^{K_1}) = \triangle(\hat{Y}^{K_1+1}) \le \sqrt{\frac{\epsilon}{\varrho\varsigma}}. \tag{27}$$

Here, we let $\overline{\triangle} = \triangle(\hat{Y}^{K_1}) = \triangle(\hat{Y}^{K_1+1})$ in order to facilitate the smooth description of the following. Combined with (27), we can see that

$$\overline{\triangle} \le \sqrt{\frac{\epsilon}{\varrho\varsigma}} \tag{28}$$

is obvious. Then, Y^{K_1} and Y^{K_1+1} will be thrown out of the set Ξ^{K_1+1} after using Lemma 1 and Step 5 of the algorithm, because there is no globally optimal solution of the problem (EP) in Y^{K_1} and Y^{K_1+1}. Furthermore, the remaining rectangles will be placed in the set Ξ^{K_1}, where

$$\Xi^{K_1} = \Xi^{K_1+1} \backslash \{Y^{K_1}, Y^{K_1+1}\} = \{Y^t : t = 1, 2, \cdots, K_1 - 1\}.$$

Of course, the new set Ξ^{K_1} will continue to be considered.

Next, let us focus on Y^{K_1-1}. According to (25) and combined with the branch rule of Section 2.3, Y^{K_1-1} will be immediately divided into two sub-rectangles $Y^{K_1-1,1}$ and $Y^{K_1-1,2}$ satisfying

$$\hat{Y}^{K_1-1} = \hat{Y}^{K_1-1,1} \cup \hat{Y}^{K_1-1,2} \tag{29}$$

and

$$\triangle(\hat{Y}^{K_1-1}) = 2\triangle(\hat{Y}^{K_1-1,1}) = 2\triangle(\hat{Y}^{K_1-1,2}) = 2\overline{\triangle}. \tag{30}$$

Therefore, Y^{K_1-1} is thrown out of the set Ξ^{K_1} by using (28)–(30), and we can know that the algorithm iterates once again. At the same time, the remaining rectangles are put into the set Ξ^{K_1-1}, that is,

$$\Xi^{K_1-1} = \Xi^{K_1} \backslash \{Y^{K_1-1}\} = \Xi^{K_1+1} \backslash \{Y^{K_1-1}, Y^{K_1}, Y^{K_1+1}\} = \{Y^t : t = 1, 2, \cdots, K_1 - 2\}.$$

Of course, Y^{K_1-2} will also be immediately divided into $Y^{K_1-2,1}$ and $Y^{K_1-2,2}$ satisfying

$$\hat{Y}^{K_1-2} = \hat{Y}^{K_1-2,1} \cup \hat{Y}^{K_1-2,2} \tag{31}$$

and

$$\triangle(\hat{Y}^{K_1-2}) = 2\triangle(\hat{Y}^{K_1-2,1}) = 2\triangle(\hat{Y}^{K_1-2,2}) = 2\triangle(\hat{Y}^{K_1-1}) = 2^2\overline{\triangle}. \tag{32}$$

Then, both $Y^{K_1-2,1}$ and $Y^{K_1-2,2}$ must be divided again for once to satisfy (28); that is, for Y^{K_1-2}, the algorithm must iterate $2^2-1=3$ times for Y^{K_1-2} to be thrown out of the set Ξ^{K_1-1}. Then, put the remaining rectangles in the set Ξ^{K_1-2}, that is,

$$\Xi^{K_1-2}=\Xi^{K_1-1}\backslash\{Y^{K_1-2}\}=\Xi^{K_1+1}\backslash\{Y^{K_1-2},Y^{K_1-1},Y^{K_1},Y^{K_1+1}\}=\{Y^t:\ t=1,2,\cdots,K_1-3\}.$$

Similar to (31) and (32), for a rectangular $Y^t(t=1,2,\cdots,K_1-1)$, we also have

$$\hat{Y}^t=\hat{Y}^{t,1}\cup\hat{Y}^{t,2}$$

and

$$\triangle(\hat{Y}^t)=2\triangle(\hat{Y}^{t,1})=2\triangle(\hat{Y}^{t,2})=2\triangle(\hat{Y}^{t+1})=2^2\triangle(\hat{Y}^{t+2})=\cdots=2^{K_1-1-t}\triangle(\hat{Y}^{K_1-1})=2^{K_1-t}\overline{\triangle}. \tag{33}$$

According to (28) and (33), the algorithm must iterate $2^{K_1-t}-1$ at most before Y^t is thrown out of its corresponding set Ξ^{t+1}.

Then, put the remaining rectangles in the set Ξ^t, that is,

$$\Xi^t=\Xi^{t+1}\backslash\{Y^t\}=\Xi^{K_1+1}\backslash\{Y^t,Y^{t+1},\cdots,Y^{K_1-2},Y^{K_1-1},Y^{K_1},Y^{K_1+1}\},\ t=1,2,\cdots,K_1-1. \tag{34}$$

Therefore, when t is taken from K_1-1 to 1, the algorithm iterates

$$K=K_1+\sum_{t=1}^{K_1-1}(2^{K_1-t}-1)=2^{K_1}-1=2^{\sum\limits_{j=1}^{p}\lceil\log_2\frac{\sqrt{\varrho\varsigma}(\overline{y}_j^0-\underline{y}_j^0)}{\sqrt{\epsilon}}\rceil}-1$$

times at most. In addition, according to (26) and (34), we have

$$\begin{aligned}\Xi^1&=\Xi^{K_1+1}\backslash\{Y^1,Y^2,\cdots,Y^{K_1-2},Y^{K_1-1},Y^{K_1},Y^{K_1+1}\},\\&=\Xi^{K_1+1}\backslash\{\hat{Y}^1\times\tilde{Y}^1,\hat{Y}^2\times\tilde{Y}^2,\cdots,\hat{Y}^{K_1}\times\tilde{Y}^{K_1},\hat{Y}^{K_1+1}\times\tilde{Y}^{K_1+1}\},\\&=\emptyset.\end{aligned}$$

then the algorithm will stop running, using Step 5 of the algorithm. □

Remark 6. *Through Theorem 6, when the proposed algorithm OSBBRA finds the $\epsilon-$globally optimal solution of the problem (LMP), we can use*

$$2KT(m+6p-3,n+2p-1)$$

as the upper bound of the running time of the algorithm, where $T(m+6p-3,n+2p-1)$ denotes the time taken to solve a linear programming problem with $n+2p-1$ variables and $m+6p-3$ constraints.

Remark 7. *Theorem 6 is fully capable of ensuring that the algorithm OSBBRA is completed in a finite number of iterations, because of the existence of such the most extreme number of iterations.*

4. Numerical Examples

In this section, we present many random experiments of different scales through the proposed algorithm. We also provide the performance comparison results compared with the previous methods to solve the problem (LMP).

The code of our algorithm was compiled on Matlab (2016a). All calculation processes were carried out on personal PCs with Intel(R) Core(TM)i5-4210M 2.60 GHz power processor 4 GB memory and the operating system used is Microsoft Windows 7. In the process of numerical experiment, the linprog solver of MATLAB(2016a) was used to solve all linear programming problems, while the quadratic convex programming problem in [27] was solved by quadprog solver in MATLAB(2016a). We use the following notation:

- Solution: the optimal solution
- Optimum: the optimal value
- Opt.val: the average of the optimal values for the 10 problems arising from the 10 sets of random coefficients calculated using the commercial software package BARON
- Iter: the number of iterations
- ϵ: tolerance
- Ref: reference
- Time: the CPU running time
- Avg: average performance of an algorithm for a set of random problems
- Std: standard deviation of performance of an algorithm for a set of random problems
- "-": the problem cannot be solved in 2400 s
- "$*$": problems of this size not solved in [41]

4.1. Feasibility Tests

In this subsection, we give several exact examples to illustrate that the algorithm OSBBRA is effective and feasible.

Example 1 [23,25,26,42]

$$\min(x_1+x_2)(x_1-x_2+7) \quad s.t. \begin{cases} 2x_1+x_2\le 14, \\ x_1+x_2\le 10, \\ -4x_1+x_2\le 0, \\ 2x_1+x_2\ge 6, \\ x_1+2x_2\ge 6, \\ x_1-x_2\le 3, \\ x_1+x_2\ge 0, \\ x_1-x_2+7\ge 0, \\ x_1,x_2\ge 0. \end{cases}$$

Example 2 [25,26,39,42]

$$\min f(x) = (0.813396x_1+0.67440x_2+0.305038x_3+0.129742x_4+0.217796)$$
$$\times\ (0.224508x_1+0.063458x_2+0.932230x_3+0.528736x_4+0.091947)$$
$$s.t. \begin{cases} 0.488509x_1+0.063565x_2+0.945686x_3+0.210704x_4\le 3.562809, \\ -0.324014x_1-0.501754x_2-0.719204x_3+0.099562x_4\le -0.052215, \\ 0.445225x_1-0.346896x_2+0.637939x_3-0.257623x_4\le 0.427920, \\ -0.202821x_1+0.647361x_2+0.920135x_3-0.983091x_4\le 0.840950, \\ -0.886420x_1-0.802444x_2-0.305441x_3-0.180123x_4\le -1.353686, \\ -0.515399x_1-0.424820x_2+0.897498x_3+0.187268x_4\le 2.137251, \\ -0.591515x_1+0.060581x_2-0.427365x_3+0.579388x_4\le -0.290987, \\ 0.423524x_1+0.940496x_2-0.437944x_3-0.742941x_4\le 0.373620, \\ x_1\ge 0, x_2\ge 0, x_3\ge 0, x_4\ge 0. \end{cases}$$

Example 3 [23]

$$\min(c_1^T x + d_1)(c_2^T x + d_2) \quad s.t. \ \ Ax = b, x \geq 0. \ .$$

where

$$b = (81, 72, 72, 9, 9, 9, 8, 8)^T, d_1 = 0, d_2 = 0,$$
$$c_1 = (1, 0, \frac{1}{9}, 0, 0, 0, 0, 0, 0, 0, 0)^T, c_2 = (0, 1, \frac{1}{9}, 0, 0, 0, 0, 0, 0, 0, 0)^T.$$
$$A = \begin{pmatrix} 9 & 9 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 1 & 8 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 8 & 8 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 7 & 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 7 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 1 & 7 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

From the literature [23], we know that Example 3 can be transformed into the following forms:

$$(LMP) := \min \ (x_1 + \frac{1}{9}x_3)(x_2 + \frac{1}{9}x_3) \qquad s.t. \begin{cases} 9x_1 + 9x_2 + 2x_3 \leq 81, \\ 8x_1 + x_2 + 8x_3 \leq 72, \\ x_1 + 8x_2 + 8x_3 \leq 72, \\ 7x_1 + x_2 + x_3 \geq 9, \\ x_1 + 7x_2 + x_3 \geq 9, \\ x_1 + x_2 + 7x_3 \geq 9, \\ 0 \leq x_1 \leq 8, \\ 0 \leq x_2 \leq 8, \\ 0 \leq x_3 \leq 9. \end{cases}$$

Then, we obtained the global optimal solution of the problem (LMP): $x = (0.0, 8.0, 1.0)^T$, and got the global optimal solution of Example 3: $x = (0.0, 8.0, 1.0, ...)^T$, where the remaining components were chosen such that $Ax = b$.

Example 4 [26,42]

$$\min x_1 + (x_1 - x_2 + 5)(x_1 + x_2 - 1) \qquad s.t. \begin{cases} -2x_1 - 3x_2 \leq -9, \\ 3x_1 - x_2 \leq 8, \\ -1x_1 + 2x_2 \leq 8, \\ x_1 + 2x_2 \leq 12, \\ x_1 - x_2 + 5 \geq 0, \\ x_1 + x_2 - 1 \geq 0, \\ x_1 \geq 0. \end{cases}$$

Example 5 [26,42]

$$\min(x_1 + x_2)(x_1 - x_2) + (x_1 + x_2 + 1)(x_1 - x_2 + 1) \qquad s.t. \begin{cases} x_1 + 2x_2 \leq 10, \\ x_1 - 3x_2 \leq 20, \\ 0 \leq x_1 \leq 3, \\ 0 \leq x_2 \leq 3. \end{cases}$$

The objective function of Example 5 can be transformed into $2(x_1 - x_2 + 3)(x_1 + x_2 + 1) - 6x_1 - 4x_2 - 5$.
Example 6 [26,42]

$$\min (x_1 + x_2)(x_1 - x_2) + (x_1 + x_2 + 2)(x_1 - x_2 + 2) \qquad s.t. \begin{cases} x_1 + 2x_2 \le 10, \\ x_1 - 3x_2 \le 20, \\ 0 \le x_1 \le 4, \\ 0 \le x_2 \le 4. \end{cases}$$

The objective function of the Example 6 can be transformed into $2(x_1 - x_2 + 4)(x_1 + x_2) - 4x_1 - 8x_2 + 4$.
Example 7

$$\min \ (-x_1 + 2x_2 + 6)(x_1 + x_2 - x_3 + 3)(x_1 + x_2 - 2x_3 + 7) \qquad s.t. \begin{cases} x_2 - 2x_3 \le 1, \\ 3x_1 - 3x_2 + 4x_3 \le 12, \\ 3x_2 - 5x_3 \ge -14, \\ x_1 \ge 0, x_2 \ge 0, x_3 \ge 0. \end{cases}$$

Example 8

$$\begin{aligned} &\min f(x) = (-4x_1 - 2x_4 + 3x_5 + 21)(4x_1 + 2x_2 + 3x_3 - 4x_4 + 4x_5 - 3) \\ &\times (3x_1 + 4x_2 + 2x_3 - 2x_4 + 2x_5 - 7)(-2x_1 + x_2 - 2x_3 + 2x_5 + 11) \\ &s.t. \begin{cases} 4x_1 + 4x_2 + 5x_3 + 3x_4 + x_5 \le 25, \\ -x_1 - 5x_2 + 2x_3 + 3x_4 + x_5 \le 2, \\ x_1 + 2x_2 + x_3 - 2x_4 + 2x_5 \ge 6, \\ 4x_2 + 3x_3 - 8x_4 + 11x_5 \ge 8, \\ x_1 + x_2 + x_3 + x_4 + x_5 \le 6, \\ x_1 \ge 1, x_2 \ge 1, x_3 \ge 1, x_4 \ge 1, x_5 \ge 1. \end{cases} \end{aligned}$$

As shown in Table 1, the algorithm OSBBRA can accurately obtain the globally optimal solution of these eight low-dimensional examples, which shows that the algorithm is effective and feasible. We can see that the algorithm OSBBRA only needs one iteration in solving Example 2 and takes the least time compared with other algorithms. For Example 3, the time and the number of iterations are the most, but only five iterations are required. Examples 1 and 4 need only two iterations to obtain the solution, and the algorithm in this paper only needs very little time to solve them. In the process of solving Example 1, the running speed of this algorithm is much faster than that in Refs. [23,25]. For Examples 5 and 6, because of the special structure of the case, OSBBRA uses more iterations and time to solve such problems than other algorithms, but it can also get the globally optimal solution in less than 0.5 s, which indicates that our relaxation bound method can be further generalized. Examples 7 and 8 are two examples constructed by us, and the results of this algorithm were compared with those of the software package BARON [43]. It can be seen that the calculation results of this algorithm are the same as those of BARON, but the running time of CPU used in our algorithm is less than that of BARON, especially the number of iterations used by BARON in solving Example 8 is much higher than that of algorithm OSBBRA.

Table 1. Comparison of results in Examples 1–8.

	Ref.	Solution	Optimum	Iter	Time	ϵ
1	[23]	(1.9999998, 7.9999988)	10.0000090	41	0.02	10^{-5}
	[25]	(2.0, 8.0)	10.0	48	5.0780	10^{-3}
	[26]	(2.0, 8.0)	10.0	1	0.0128	10^{-6}
	[42]	(2.0, 8.0)	10.0	1	0.046	10^{-4}
	OSBBRA	(2.0000, 8.0000)	10.0000	2	0.0182	10^{-6}
2	[25]	(1.3148, 0.1396, 0.0, 0.4233)	0.8902	1	0.1880	10^{-3}
	[26]	(1.3148, 0.1396, 0.0, 0.4233)	0.8902	1	0.0601	10^{-6}
	[39]	(1.3148, 0.1396, 0.0000, 0.4233)	0.890190	3	0.047	0.05
	[42]	(1.3148, 0.1396, 0.0000, 0.4233)	0.8902	1	0.093	10^{-4}
	OSBBRA	(1.3148, 0.1396, 0.0000, 0.4233)	0.8902	1	0.0226	10^{-6}
3	[23]	(8.0, 0.0, 1.0, ...)	0.901235	3	0.00	10^{-3}
	[27]	(0.0, 8.0, 1.0, ...)	0.901235	3	0.0469	–
	OSBBRA	(0.0, 8.0, 1.0, ...)	0.901235	5	0.0743	10^{-3}
4	[26]	(0.0, 4.0)	3	1	0.0693	10^{-6}
	[42]	(0, 4)	3	1	0.062	10^{-4}
	OSBBRA	(0.0000, 4.0000)	3.0000	2	0.0218	10^{-6}
5	[26]	(1.0, 3.0)	−13	1	0.0868	10^{-6}
	[42]	(1, 3)	−13	1	0.047	10^{-4}
	OSBBRA	(1.0000, 3.0000)	−13.0000	16	0.1845	10^{-6}
6	[26]	(1.0, 4.0)	−22	1	0.0849	10^{-6}
	[42]	(1, 4)	−22	1	0.046	10^{-4}
	OSBBRA	(1.0000, 4.0000)	−22.0000	19	0.2143	10^{-6}
7	*BARON*	(0.0000, 0.0000, 2.8000)	1.6800	1	0.3824	10^{-6}
	OSBBRA	(0.0000, 0.0000, 2.7999)	1.6800	9	0.1377	10^{-6}
8	*BARON*	(1.0000, 1.9999, 1.0000, 1.0000, 1.0000)	9503.9999	155	1.4662	10^{-6}
	OSBBRA	(1.0000, 2.0000, 1.0000, 1.0000, 1.0000)	9503.9999	2	0.0691	10^{-6}

The above eight small-scale examples only illustrate the validity and feasibility of the algorithm OSBBRA, but we cannot know the other performance of the algorithm in solving the problem (LMP). Therefore, in the next subsection, we describe the other features of the algorithm by performing a series of random tests. The experimental results of random tests are presented in Tables 2–9, and the information disclosed, in these eight tables, is analyzed to illustrate the performance and applicable conditions of the algorithm.

4.2. Testing of Random Problems

To test the other features of the algorithm, we used three random problem generation schemes to generate random (LMP):

$$(LMP1)\ :\ \min\prod_{j=1}^{p} c_j^T x \quad s.t.\begin{cases} \sum_{i=1}^{n} A_{si}x_i \le b_s,\ s=1,2,\cdots,m, \\ 0\le x_i\le 1,\ i=1,2,\cdots,n. \end{cases}$$

$$(LMP2)\ :\ \min\prod_{j=1}^{2}(c_j^T x+1) \quad s.t.\begin{cases} \sum_{i=1}^{n} A_{si}x_i \le b_s,\ s=1,2,\cdots,m, \\ x_i\ge 0,\ i=1,2,\cdots,n. \end{cases}$$

In (LMP1) and (LMP2), A_{si} is randomly generated in the interval [−1,1], and the value of the right b_s is generated by $\sum_{i=1}^{n} A_{si} + 2\pi$, where π is randomly generated in the interval [0,1]. This is consistent with the methods in Refs. [19,41].

$$(LMP3) \ : \ \min \prod_{j=1}^{p} c_j^T x \quad s.t. \begin{cases} \sum_{i=1}^{n} A_{si} x_i \geq b_s, \ s = 1, 2, \cdots, m, \\ x_i \geq 0, \ i = 1, 2, \cdots, n. \end{cases}$$

The A_{si}, b_s and c_j of (LMP3) were randomly generated in the interval [0, 100]. This agrees that the random number is generated in Ref. [19].

In addition, for all problems, we solves 10 different random instances for each size, and give the statistical information of the results. In addition, the tolerance was set to 10^{-6} for all random problems.

Remark 8. *Each random example is based on the size of (p, m, n) a set of random coefficients that randomly generate the problem in a given interval, and then solved by the algorithms of OSBBRA, BARON, Reference [27], Reference [38], Reference [19], Reference [41] respectively, to obtain the corresponding calculation results.*

4.2.1. Testing of Random Problem (LMP1)

For the problem (LMP1), we compared the algorithm OSBBRA with the algorithm in Ref. [27] and the Primal algorithm in Ref. [38], respectively, and used the optimal value obtained by commercial software package BARON as a standard to evaluate the quality of the optimal solution obtained by these three algorithms. For each group (p, m, n), 10 groups of examples were randomly generated to obtain the average value of the calculated results. For the measurement of the quality of the optimal solution, we used the following formula:

$$Optimum.ratio = |\frac{f(x^*) - Opt.val}{Opt.val}|$$

where x^* is the final optimal solution of the three algorithms and $f(x^*)$ is the optimal value corresponding to the three algorithms. To see the optimal value quality clearly, we used $Optimum.ratio \times 10^5$, and the corresponding calculation results are listed in Table 2.

As can be seen in Table 2, in terms of the quality of the obtained optimal solution, the three algorithms are arranged in the order of optimal to lowest order: OSBBRA, Ref. [38], and Ref. [27]. This means that the optimal value obtained by our algorithm is the most accurate. For the CPU running time of the algorithm, the following cases are discussed:

(i) For $(p, m, n) = (2, 20, 200)$, $(2, 30, 300)$, $(2, 40, 400)$, the time spent is arranged as OSBBRA, Ref. [38], Ref. [27] in order from less to more. Furthermore, the time occupied by OSBBRA is the least in the three algorithms.

(ii) In the case of $(p, m, n) = (2, 10, 100)$ and $p = 3, 4$, the time occupied by OSBBRA is the most in the three algorithms, followed by the algorithm in Ref. [27], and the time of the algorithm in the Ref. [38] is the least.

It can be seen that, in Case (ii), our algorithm OSBBRA takes the most time, but, in Case (i), OSBBRA takes less time than the other two algorithms. This is because, in the first case, $p \ll n$ can better reflect the advantages of our algorithm, which can also be seen in the higher scale experimental results in Table 3. This feature of algorithm OSBBRA is also reflected in Tables 4 and 5. Of course, the data in Tables 8 and 9 also imply this reason, which we mention again below.

Table 2. The average result of 10 low-dimensional random problems (LMP1).

(p, m, n)	$Opt.val$	$Optimum$			$Optimum.ratio \times 10^5$			$Time$		
		$OSBBRA$	**Ref.** [27]	**Ref.** [38]	$OSBBRA$	**Ref.** [27]	**Ref.** [38]	$OSBBRA$	**Ref.** [27]	**Ref.** [38]
(2, 10, 100)	29.9097	29.9097	29.9187	29.9157	0.067	33.6192	2.2411	0.5537	0.4737	0.1615
(2, 20, 200)	133.3865	133.3866	133.434	133.4183	0.0023	34.3035	2.2944	0.5746	1.2424	0.6026
(2, 30, 300)	182.721	182.721	182.7537	182.7428	0.0072	24.6207	1.6412	1.8197	3.194	1.9535
(2, 40, 400)	351.5904	351.5926	351.754	351.6994	0.6039	43.3172	2.8874	2.2338	6.4919	4.5876
(3, 10, 100)	96.6445	96.6445	96.6546	96.6514	0.0643	14.0031	9.5493	0.6325	0.5851	0.286
(3, 20, 200)	1375.6595	1375.6434	1375.9541	1375.8571	1.17	43.1338	2.8891	2.2412	1.4385	0.7983
(3, 30, 300)	6510.4561	6510.0954	6514.145	6512.9254	5.54	51.3809	34.4346	3.796	3.6375	2.3271
(3, 40, 400)	10,302.6134	10,303.1708	10,306.0578	10,304.9265	5.41	27.5294	18.5314	13.1311	8.0374	5.6722
(4, 10, 100)	521.4775	521.4759	521.7151	521.6364	0.293	48.6827	32.5493	2.6662	1.1085	0.4045
(4, 20, 200)	22,512.6836	22,511.8327	22,527.3196	22,522.4657	3.78	65.0123	43.4513	19.8316	2.4383	0.9227
(4, 30, 300)	248,728.2167	248,734.8081	248,898.3767	248,841.8108	2.65	68.4121	45.6699	25.0813	8.8808	5.8772
(4, 40, 400)	170,323.0981	170,338.2399	170,446.6353	170,405.9477	8.89	72.5311	48.6425	27.3393	12.6321	7.5582

Table 3. The average result of 10 high-dimensional random problems (LMP1).

(p, m, n)	*Optimum*			*Time*		
	OSBBRA	**Ref.** [27]	**Ref.** [38]	***OSBBRA***	**Ref.** [27]	**Ref.** [38]
$(5, 50, 1000)$	1.5534×10^8	1.5541×10^8	1.5539×10^8	941.3086	873.6958	723.1655
$(5, 60, 2000)$	6.5945×10^8	6.5985×10^8	6.5972×10^8	449.8872	1529.7757	917.3199
$(5, 70, 3000)$	8.0127×10^9	8.0144×10^9	8.0138×10^9	782.0679	1639.8610	1323.6961
$(5, 80, 4000)$	4.0686×10^9	–	4.0692×10^9	1007.6914	–	1796.8545
$(6, 50, 1000)$	2.5156×10^9	2.5166×10^9	2.5163×10^9	1202.9066	1332.1953	1081.1864
$(6, 60, 2000)$	1.8276×10^{10}	1.8281×10^{10}	1.8279×10^{10}	1101.4307	1542.3941	1484.3904
$(6, 70, 3000)$	2.0195×10^{11}	2.0208×10^{11}	2.0204×10^{11}	1360.1356	1927.7173	1619.3256
$(6, 80, 4000)$	3.5453×10^{11}	–	–	1404.9818	–	–
$(7, 50, 1000)$	2.0887×10^{11}	2.0892×10^{11}	2.0890×10^{11}	1500.4977	1661.2653	1321.2203
$(7, 60, 2000)$	1.4108×10^{13}	1.4111×10^{13}	1.4110×10^{13}	1690.9317	2169.4610	1923.1707
$(7, 70, 3000)$	5.0654×10^{13}	–	5.0655×10^{13}	1893.3633	–	2164.7689
$(7, 80, 4000)$	2.6370×10^{14}	–	–	2116.0214	–	–

For the problem (LMP1), we also performed higher-dimensional numerical experiments and record the results in Table 3. In the high dimensional example, we did not use the software package BARON to calculate, because BARON takes a long time to solve the higher-dimensional (LMP1) problem, and, in Table 2, we only use the results of BARON to evaluate the optimal worth quality of the three algorithms. In Table 3, the optimal quality of our algorithm is still the best, followed by Refs. [27,38]. Especially when solving the 4000-dimensional problem, the algorithm in Ref. [27] cannot find the optimal solution in 2400 s, and, when $p = 6, 7$, the algorithm in Ref. [38] also fails to solve the optimal solution of the problem within 2400 s. In addition, the results in Tables 2 and 3 also show that the computing power of the algorithm in Ref. [38] is better than that in Ref. [27]. The results in Table 3 show that, when $(p, m, n) = (5, 50, 1000)$, the OSBBRA takes less time than the other two algorithms, whereas, in the case of $(p, m, n) = (6, 50, 1000)$, $(7, 50, 1000)$, the OSBBRA takes less time only in the case of other larger-scale problems in [27]. In addition, although the time spent on these three algorithms increases with the size of the problem, OSBBRA always takes less than 2400 s. This shows that our algorithm is more suitable to solve large-scale optimization problems (LMP) than other algorithms under certain circumstances.

The phenomenon reflected in the results of Tables 2 and 3 is mainly due to the characteristics of each algorithm itself. Through the understanding of Ref. [27], each iteration of its algorithm to solve the problem (LMP) requires solving a quadratic programming problem, which does take less time to solve small-scale problems in a certain range than the algorithm OSBBRA, but it takes longer than OSBBRA to solve some large-scale problems. Although the original algorithm in Ref. [38] needs to solve only two linear programming problems in each iteration, and indeed has less time to solve small-scale problems than the other two algorithms, it is a tangent plane algorithm, thus increasing the number of constraints for each iteration. As the scale of the problem increases to a certain extent, its advantages will disappear because the increased constraints will gradually increase the storage space of the computer as it progresses and eventually slow down the speed of the computer. At the same time, the algorithms in both Ref. [27,38] need to store a large number of vertices, which will also affect the running time of the computer. The algorithm OSBBRA needs to store the hyper-rectangle whose length is $2p - 1$, and at most two hyper-rectangles are added in each iterative step. Through the pruning operation and rectangle reduction technology of the branch-and-bound algorithm, the hyper-rectangle without the globally optimal solution will be reduced, so that the computer storage space will be saved, and the influence of the storage space on the performance of the algorithm will be reduced as much as possible. Therefore, the computational performance of the algorithm OSBBRA is affected by the length $2p - 1$ of the hyper-rectangle.

Next, we performed additional numerical experiments on the problem (LMP1) with OSBBRA and recorded the results in Tables 4 and 5. The main purpose of Tables 4 and 5 is to explore the case that the number of computer-stored consumption-matrixes varies by the size of (p, m, n) in the search for optimal solutions. Meanwhile, the average consumption time and the number of iterations in this process are also recorded in the table.

Table 4. The results of random calculation for (LMP1).

(m, n)		$p = 2$	$p = 3$	$p = 4$	$p = 5$	$p = 6$	$p = 7$
(10, 20)	Ave.Time	0.1452	0.3205	0.6654	1.2609	1.7287	19.2006
	Ave.Iter	4.6	10.7	10.8	45.7	43.8	458.8
	Ave.Node	1.4	2.9	2.2	11.8	12.8	139.2
(20, 40)	Ave.Time	0.1514	0.3286	1.1668	1.3929	6.6506	23.8151
	Ave.Iter	3.9	9.7	25.1	45.7	77.0	496.6
	Ave.Node	1.3	2.8	5.1	13.9	25.6	131.7
(30, 60)	Ave.Time	0.1720	0.4581	1.8145	7.6766	11.1125	64.3865
	Ave.Iter	5.1	11.5	32.6	184.6	101.9	1013.5
	Ave.Node	1.2	2.3	8.6	31.7	28.7	157.7
(40, 80)	Ave.Time	0.2441	0.7060	2.2780	10.4108	16.3945	86.1229
	Ave.Iter	5.2	12.9	46.4	117.6	133.1	1177.3
	Ave.Node	1.1	3.1	6.2	21.0	40.0	166.9
(50, 100)	Ave.Time	0.4178	1.3477	2.5277	15.5902	22.0331	127.1028
	Ave.Iter	6.1	15.3	42.6	167	271.2	1398
	Ave.Node	1.4	3.2	15.4	27.2	55.8	186.9
(60, 120)	Ave.Time	0.6179	2.7911	6.5797	18.3711	25.2491	155.5248
	Ave.Iter	6.8	20.8	76.5	143.4	387.1	640.2
	Ave.Node	1.5	4.2	10.8	23.5	85.1	109.2
(70, 140)	Ave.Time	1.8234	3.8380	8.1493	38.7537	63.8334	185.5591
	Ave.Iter	9.0	24.0	62.2	212.5	766.5	1197.4
	Ave.Node	1.6	5.3	13.4	30.4	69.4	196.0
(80, 160)	Ave.tTime	1.9886	6.7688	14.7688	81.1816	98.5674	278.3208
	Ave.Iter	6.5	27.5	43.8	310.2	315.3611	891.1
	Ave.Node	1.5	4.1	11.4	39.4	72.4	122.3
(90, 180)	Ave.Time	2.4104	8.8114	19.2281	93.8722	110.7434	293.8722
	Ave.Iter	6.9	26.6	37.2	298.4	444.2	816.7
	Ave.Node	1.2	4.2	16.6	32.4	91.2	131.6
(100, 200)	Ave.Time	4.1027	14.2772	46.8494	114.7671	120.0034	321.4063
	Ave.Iter	8.4	31.0	79.2	230.6	657.8	469.3
	Ave.Node	1.7	5.2	15.8	26.1	116.9	97.0

Table 5. The results of random calculation for (LMP1).

(p, m, n)	$Ave.Iter$	$Ave.Time$	$Ave.Node$
(10, 10, 2)	10.9	0.3576	1.3
(20, 10, 2)	2349	48.5707	1891.8
(30, 10, 2)	11,369.5	240.4569	8072.5
(40, 10, 2)	5060.4	120.8821	4937.7
(2, 10, 1000)	15.5	2.6293	4.0
(2, 10, 2000)	28.5	14.0012	75.9
(3, 10, 1000)	101.8	19.3235	25.3
(3, 10, 2000)	185.4	90.3898	37.0
(4, 10, 1000)	757.6	156.5649	134.7
(4, 10, 2000)	1352.1	995.4707	257.3

The data of two groups of numerical experiments are separately calculated by the algorithm OSBBRA, which is used to observe how the size of p and n affects the performance (Ave.Time and Ave.Iter) of the algorithm.The first group is to fix p to 2, 3, 4, 5, 6, and 7 in turn, take $n = 2m$, take m as from 10 to 100 by steps of 10, and the calculated results are shown in Table 4. The results in Table 4 show that, when p is fixed, the average CPU running time Ave.Time increases as the dimension n of the decision variable becomes larger, and the average maximum number of nodes Ave.Node and the average number of iterations Ave.Iter of the algorithm stored in the branching and delimiting tree either increases or decreases. For fixed (m, n), as p increases, the size of the Ave.Time, Ave.Iter, and Ave.Node are increased; in particular, when p transitions from 6 to 7, the rise speed of these three increases sharply, which indicates that the effect of the size of p on the performance of the algorithm is sharp. The results of Tables 2–4 show that p and n have an effect on the algorithm calculation. In more extreme cases, we did another set of experiments. As can be seen from the first four rows of data in Table 5, when fixed $(m, n) = (10, 2)$ and p in order of 10, 20, 30, and 40, t Ave.Time, Ave.Iter, and Ave.Node are increasing and increasing rapidly, at which point p is larger than n. From the last six rows of data in Table 5, we can also see that when p is much less than n, the algorithm can obtain the globally optimal solution of the problem (LMP) in a short time, and the performance of the algorithm is very sensitive to the size of p, which is mainly because our branch operation is to branch the p-dimensional space $\hat{Y}^k$ (it is the same as the definition in Section 2.3).

4.2.2. Testing of Random Problem (LMP2)

Through the previous experiments, we know the computational effect of OSBBRA, which is influenced by (p, m, n). Next, we conducted numerical experiments on the special scheme (LMP2), and we fixed $p = 2$ to observe the effect of (m, n) on the algorithm. For the random generation scheme (LMP2), the algorithm OSBBRA was compared with the calculated results in Ref. [19,41], and the related data are recorded in Table 6. According to the method in Ref. [19], the normalized CPU running time and the number of iterations of the problem (LMP2) with respect to the 10×20 were obtained using the formula

$$\frac{Time(Iter)\ on\ m \times\ n\ problem}{Time\ (Iter)\ on\ 10 \times 20\ problem},$$

respectively, and the data are recorded in Table 7.

Table 6. Computational results on (LMP2) (p = 2) and comparison with results reported in [19,41].

$LMP2$	Ref. [19]		Ref. [41]		$OSBBRA$	
(m,n)	$Avg(Std)Time$	$Avg(Std)Iter$	$Avg(Std)Time$	$Avg(Std)Iter$	$Avg(Std)Time$	$Avg(Std)Iter$
(10, 20)	0.1 (0.1)	6.2 (4.3)	0.6062 (0.0695)	14.2 (1.5492)	0.2083 (0.3861)	2.6 (6.2561)
(20, 20)	0.2 (0.1)	7.0 (2.8)	0.8368 (0.0756)	17.4 (1.7127)	0.2814 (0.5504)	4.8 (6.9793)
(22, 20)	0.2 (0.1)	8.8 (4.2)	0.9460 (0.1235)	18.5 (1.9003)	0.3231 (0.9257)	6.0 (12.2564)
(20, 30)	0.3 (0.1)	8.0 (3.6)	1.0781 (0.0674)	19.9 (0.5676)	0.3302 (0.4899)	6.4 (7.4951)
(35, 50)	1.0 (0.4)	11.0 (3.5)	1.8415 (0.1338)	21.2 (0.4316)	0.4267 (0.8646)	8.1 (11.6772)
(45, 60)	1.2 (0.3)	13.3 (4.9)	2.4338 (0.1016)	23.0 (0.6667)	0.4867 (0.8930)	8.7 (14.2688)
(45, 100)	3.9 (1.2)	15.2 (6.0)	5.1287 (0.0935)	35.7 (1.1595)	0.6049 (0.9664)	11.9 (12.3809)
(60, 100)	5.6 (1.2)	14.8 (3.8)	6.8143 (0.1713)	36.1 (0.7379)	0.7955 (1.2783)	9.7 (12.9822)
(70, 100)	6.5 (3.0)	17.5 (7.2)	8.1967 (0.2121)	36.6 (1.2649)	0.8152 (1.3057)	8.3 (11.6638)
(70, 120)	9.0 (1.8)	17.2 (4.8)	9.5642 (0.2975)	39.1 (1.6633)	0.9693 (1.3529)	10.1 (14.6462)
(100, 100)	7.6 (1.0)	13.3 (4.3)	13.0578 (0.3543)	37.5 (2.1731)	1.1889 (1.2506)	11.1 (9.0549)
(102, 150)	15.9 (2.9)	24.8 (7.0)	$*$	$*$	1.7051 (0.9492)	12.6 (8.9361)
(102, 190)	21.4 (3.5)	28.4 (7.5)	$*$	$*$	1.8014 (1.7103)	8.4 (8.0443)
(72, 199)	18.3 (6.2)	25.5 (8.3)	$*$	$*$	1.5827 (2.1399)	9.7 (9.6171)
(110, 199)	22.7 (3.0)	21.7 (5.7)	$*$	$*$	2.9039 (4.1332)	9.7 (16.5476)

Table 7. Computational results of Normalized values on (LMP2).

$(LMP2)$	$Normalized\ Values(p=2)$					
(m,n)	Ref. [19]		Ref. [41]		$OSBBRA$	
	$(Avg)Time$	$(Avg)Iter$	$(Avg)Time$	$(Avg)Iter$	$(Avg)Time$	$(Avg)Iter$
(10, 20)	1	1	1	1	1	1
(20, 20)	1	1	1	1	1	2
(20, 30)	2	1	2	1	2	2
(45, 60)	8	2	4	2	2	3
(70, 100)	46	3	14	3	4	3
(100, 100)	54	2	22	2	6	4
(102, 150)	114	4	$*$	$*$	8	5
(110, 199)	162	4	$*$	$*$	14	4

First, the results of Table 6 show that the stability of our algorithm is not as good as the other two. Table 7 shows that our algorithm has the best performance in the average case in terms of time, but, in terms of the number of iterations, the performance of our algorithm OSBBRA in the average case is slightly worse than that of the other two algorithms. When $(m,n) = (110,199)$, the matrix size of the solved problem is 109 times larger than that of the basic problem, and the time used by the algorithm OSBBRA is only 14 times larger than that of the basic problem. Whereas the algorithm in Ref. [19] used 162 times more time than the basic problem. In the case of $(m,n) = (100,100)$, we solved the problem with only six times the time of the fundamental problem, which is 50 times larger than the fundamental problem in terms of the size of the constraint matrix, whereas the algorithm of Ref. [41] takes 22 times the time, and the algorithm of Ref. [41] takes 54 times the time.

To sum up, the algorithm OSBBRA has less computational time increase than the algorithms in Ref. [19,41], in this particular case of fixed $p = 2$. In the next subsection, we take the case of $p = 2$ as the basic problem to test the the growth of computing time requirements of algorithm OSBBRA compared to the Ref. [19].

4.2.3. Testing of Random Problem (LMP3)

The results in Section 4.2.2 show that, in the case of $p = 2$, our algorithm OSBBRA takes far less time to solve a relatively large-scale problem than the algorithm in Ref. [19], but less stable. In this subsection, we use a stochastic scheme (LMP3) with a wider range of values of random coefficients to verify the growth of computing time requirements (measured by r_p, and the definition of r_p is given below) of our algorithm based on the premise of $p = 2$. In addition, To better contrast with Ref. [19] and reduce the problem of unnecessary computation, we simply extracted the data that can be compared in Ref. [19] and used our algorithm to calculate, and record the experimental results in Table 8, while recording the values of r_p in Table 9. For $p = 2, 3, 4$ and 5, r_p was obtained by calculating the formula

$$r_p = \frac{AvgTime\ for\ p = i}{AvgTime\ for\ p = 2}.$$

Table 8. Computational results on (LMP3) and comparison with results reported in [19].

p	(m,n)	**Ref.** [19]		*OSBBRA*	
		$Avg(Std)Time$	$Avg(Std)Iter$	$Avg(Std)Time$	$Avg(Std)Iter$
2	(20, 30)	0.3(0.1)	9.0(3.1)	0.1(0.1)	2.3(5.5)
	(100, 100)	5.8(2.0)	17.5(8.5)	0.4(1.2)	2.5(7.5)
	(120, 120)	8.9(2.7)	15.8(6.8)	0.9(1.8)	3.0(7.4)
	(200, 200)	50.1(13.0)	25.8(6.2)	8.8(22.6)	8.3(18.6)
3	(20, 30)	0.8(0.3)	39.4(20.2)	0.2(0.9)	2.3(6.6)
	(100, 100)	25.6(9.6)	90.6(24.4)	1.2(3.4)	7.8(23.2)
	(120, 120)	35.3(10.0)	82.1(40.9)	2.1(6.5)	6.3(21.3)
	(200, 200)	149.0(60.2)	87.3(46.9)	18.8(92.4)	9.9(43.1)
4	(20, 30)	2.6(0.8)	158.2(64.8)	0.2(1.2)	7(56.9)
	(100, 100)	61.0(21.1)	243.8(117.8)	5.7(50.0)	17.9(160.3)
	(120, 120)	94.2(23.3)	271.4(70.2)	17.2(94.5)	35.3(186.4)
	(200, 200)	396.3(189.4)	301.4(171.7)	127.0(812.3)	47.6(301.1)
5	(20, 30)	6.0(2.0)	370.8(108.2)	0.4(3.3)	13.9(122.3)
	(100, 100)	197.9(38.4)	830.8(148.3)	16.5(152.3)	56.7(528.4)
	(120, 120)	245.4(97.0)	686.0(285.2)	21.5(196.0)	46.4(430.7)
	(200, 200)	1381.1(860.1)	1047.5(693.1)	122.8(1137.1)	52.4(491.5)

Table 9. Computational results on (LMP3) and comparison with results reported in [19].

(LMP3)	**Normalized AvgTime**							
(m,n)	**Ref.** [19]				*OSBBRA*			
	r_2	r_3	r_4	r_5	r_2	r_3	r_4	r_5
(20,30)	1	3	10	23	1	2	2	4
(100,100)	1	4	10	34	1	3	14	41
(120,120)	1	4	11	28	1	2	19	24
(200,200)	1	3	8	28	1	2	14	14

The calculated results in Table 8 show that the stability of OSBBRA is still poor, which also has an effect on the value of r_p. Our algorithm is particularly sensitive to the values of p, as can be seen from the experimental data and analysis in Tables 4 and 5, and in the case of $p = 2, 3$ or $(m, n) = (20, 30)$ the growth of computing time requirements is slower. In the case of $(m, n) = (100, 100), (120, 120), (200, 200)$ and $p = 4, 5$, the growth of computing time requirements increases. However, when $(m, n) = (200, 200)$, the growth of computing time requirements for $p = 5$ is similar to that for $p = 4$, but the random coefficient values are diverse, and the stability of our algorithm is slightly poor; it is also acceptable to produce such a result. Furthermore, the results in Table 9 show that the computational requirements of the algorithm OSBBRA increase more rapidly than those of [19] only in $(p, m, n) = (4, 100, 100)$, $(4, 120, 120)$, $(4, 200, 200)$, and $(5, 100, 100)$. When n is relatively large compared to p, the value of r_p is relatively small, which is because our algorithm has a distinct advantage in solving this case. The results of Tables 4 and 5 also show that, in the case of $p \ll n$, the growth of computing time requirements of the algorithm grows slowly, which also implies in another aspect the reason the value of r_p in Table 9 is relatively small, which also means that our algorithm has obvious advantages in solving this case.

From the experimental results in Sections 4.2.1–4.2.3, we can summarize that our algorithm compared with other algorithms is characterized by the high accuracy of the calculated optimal value, slightly poor stability, and the computational effect of solving large-scale problems with high dimensions in the case of $p \ll n$ is more advantageous than other algorithms. In fact, in practical problems, the numerical size of p in problem (LMP) is generally not more than 10, and the dimension n of decision variable is much larger than p. In the process of branching, the number of vertices of the divided $p-$dimensional rectangle is 2^p, which is very small compared with the $n-$dimensional hyper-rectangle with the partition number of 2^n, which is the main reason our algorithm performs better than those in Ref. [19,27,38,41] in solving this kind of large-scale problems. We can also see from Theorem 5 in Ref. [44] that p and n affect the convergence rate of the output space algorithm.

5. Conclusions

In this paper, an output-space branch-and-bound reduction algorithm (OSBBRA) is proposed to solve the problem (LMP). Based on a new bilinear function relaxation technique, the linear relaxation problem of the equivalent problem (EP) is constructed, and other related parts of the algorithm OSBBRA (Bounding operation, Branching operation and rectangle-reducing operation) are given. In Section 4, the feasibility, effectiveness, and other performance metrics of the algorithm are fully illustrated by a large number of numerical experiments, and it is pointed out that the algorithm is more effective in solving high-dimensional problems under the condition of $p \ll n$. The method in this paper can also be directly extended and used to solve the linear maximum multiplicative programming problem. In a broader sense, it can also be indirectly promoted, and we will also consider this issue in future academic research.

Author Contributions: B.Z. and Y.G. conceived of and designed the study. B.Z. and X.L. performed the experiments. B.Z. wrote the paper. Y.G. and X.H. reviewed and edited the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China under Grant (11961001), the Construction Project of first-class subjects in Ningxia higher Education (NXY LXK2017B09), and the major proprietary funded project of North Minzu University (ZDZX201901).

Acknowledgments: The authors are grateful to the responsible editor and the anonymous references for their valuable comments and suggestions, which have greatly improved the earlier version of this paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LMP	linear multiplicative programming
EP	equivalent nonlinear programming
LRP	linear relaxation programming
T	transpose of a vector or matrix

References

1. Maranas, C.D.; Androulakis, I.P.; Floudas, C.A.; Berger, A.J.; Mulvey, J.M. Solving long-term financial planning problems via global optimization. *J. Econ. Dyn. Control* **1997**, *21*, 1405–1425. [CrossRef]
2. Konno, H.; Shirakawa, H.; Yamazaki, H. A mean-absolute deviation-skewness portfolio optimization model. *Ann. Oper. Res.* **1993**, *45*, 205–220. [CrossRef]
3. Nicholas, R.; Layard, P.R.G.; Walters, A.A. Microeconomic Theory. *Economica* **1980**, *47*, 211.
4. Mulvey, J.M.; Vanderbei, R.J.; Zenios, S.A. Robust Optimization of Large-Scale Systems. *Oper. Res.* **1995**, *43*, 264–281. [CrossRef]
5. Bennett, K.P. Global tree optimization: A non-greedy decision tree algorithm. *Comput. Sci. Stat.* **1994**, *26*, 156.
6. Benson, H.P. Vector maximization with two objective functions. *J. Optim. Theory Appl.* **1979**, *28*, 253–257. [CrossRef]
7. Dennis, D.F. Analyzing Public Inputs to Multiple Objective Decisions on National Forests Using Conjoint Analysis. *For. Sci.* **1998**, *44*, 421–429.
8. Dorneich, M.C.; Sahinidis, N.V. Global optimization algorithms for chip layout and compaction. *Eng. Optim.* **1995**, *25*, 131–154. [CrossRef]
9. Kuno, T. Globally determining a minimum-area rectangle enclosing the projection of a higher-dimensional set. *Oper. Res. Lett.* **1993**, *13*, 295–303. [CrossRef]
10. Mititelu, Ş.; Treanţă, S. Efficiency conditions in vector control problems governed by multiple integrals. *J. Appl. Math. Comput.* **2018**, *57*, 647–665. [CrossRef]
11. Treanţă, S. On Locally and Globally Optimal Solutions in Scalar Variational Control Problems. *Mathematics* **2019**, *7*, 829. [CrossRef]
12. Treanţă, S. Multiobjective fractional variational problem on higher-order jet bundles. *Commun. Math. Stat.* **2016**, *4*, 323–340. [CrossRef]
13. Treanţă, S. On a new class of vector variational control problems. *Numer. Funct. Anal. Optim.* **2018**, *39*, 1594–1603. [CrossRef]
14. Saghand, P.G.; Charkhgard, H.; Kwon, C. A branch-and-bound algorithm for a class of mixed integer linear maximum multiplicative programs: A bi-objective optimization approach. *Comput. Oper. Res.* **2019**, *101*, 263–274. [CrossRef]
15. Grötschel, M.; Lovász, L.; Schrijver, A. *Geometric Algorithms and Combinatorial Optimization*; Springe: Berlin, Germany, 1988.
16. Charkhgard, H.; Savelsbergh, M.; Talebian, M. A linear programming based algorithm to solve a class of optimization problems with a multi-linear objective function and affine constraints. *Comput. Oper. Res.* **2018**, *89*, 17–30. [CrossRef]
17. Matsui, T. NP-Hardness of linear multiplicative programming and related problems. *J. Glob. Optim.* **1996**, *9*, 113–119. [CrossRef]
18. Kuno, T. A finite branch-and-bound algorithm for linear multiplicative programming. *Appl. Math. Comput.* **2001**, *20*, 119–135.
19. Ryoo, H.S.; Sahinidis, N.V. Global optimization of multiplicative programs. *J. Glob. Optim.* **2003**, *26*, 387–418. [CrossRef]
20. Kuno, T. Solving a class of multiplicative programs with 0-1 knapsack constraints. *J. Optim. Theory Appl.* **1999**, *103*, 121–135. [CrossRef]
21. Benson, H.P. An outcome space branch and bound-outer approximation algorithm for convex multiplicative programming. *J. Glob. Optim.* **1999**, *15*, 315–342. [CrossRef]
22. Jiao, H. A branch and bound algorithm for globally solving a class of nonconvex programming problems. *Nonlinear Anal. Theory Methods Appl.* **2009**, *70*, 1113–1123. [CrossRef]
23. Chen, Y.; Jiao, H. A nonisolated optimal solution of general linear multiplicative programming problems. *Comput. Oper. Res.* **2009**, *36*, 2573–2579. [CrossRef]

24. Shen, P.; Bai, X.; Li, W. A new accelerating method for globally solving a class of nonconvex programming problems. *Nonlinear Anal. Theory Methods Appl.* **2009**, *71*, 2866–2876. [CrossRef]
25. Wang, C.F.; Liu, S.Y.; Shen, P.P. Global minimization of a generalized linear multiplicative programming. *Appl. Math. Model.* **2012**, *36*, 2446–2451. [CrossRef]
26. Wang, C.F.; Bai, Y.Q.; Shen, P.P. A practicable branch-and-bound algorithm for globally solving linear multiplicative programming. *Optimization* **2017**, *66*, 397–405. [CrossRef]
27. Gao, Y.; Xu, C.; Yang, Y. An outcome-space finite algorithm for solving linear multiplicative programming. *Appl. Math. Comput.* **2006**, *179*, 494–505. [CrossRef]
28. Kuno, T.; Yajima, Y.; Konno, H. An outer approximation method for minimizing the product of several convex functions on a convex set. *J. Glob. Optim.* **1993**, *3*, 325–335. [CrossRef]
29. Pardalos, P.M. Polynomial time algorithms for some classes of constrained quadratic problems. *Optimization* **1990**, *21*, 843–853. [CrossRef]
30. Liu, X.J.; Umegaki, T.; Yamamoto, Y. Heuristic methods for linear multiplicative programming. *J. Glob. Optim.* **1999**, *15*, 433–447. [CrossRef]
31. Benson, H.P.; Boger, G.M. Multiplicative programming problems: Analysis and efficient point search heuristic. *J. Optim. Theory Appl.* **1997**, *94*, 487–510. [CrossRef]
32. Benson, H.P.; Boger, G.M. Outcome-space cutting-plane algorithm for linear multiplicative programming. *J. Optim. Theory Appl.* **2000**, *104*, 301–332. [CrossRef]
33. Konno, H.; Kuno, T.; Yajima, Y. Global minimization of a generalized convex multiplicative function. *J. Glob. Optim.* **1994**, *4*, 47–62. [CrossRef]
34. Konno, H.; Yajima, Y.; Matsui, T. Parametric simplex algorithms for solving a special class of nonconvex minimization problems. *J. Glob. Optim.* **1991**, *1*, 65–81. [CrossRef]
35. Van Thoai, N. A global optimization approach for solving the convex multiplicative programming problem. *J. Glob. Optim.* **1991**, *1*, 341–357. [CrossRef]
36. Youness, E.A. Level set algorithm for solving convex multiplicative programming problems. *Appl. Math. Comput.* **2005**, *167*, 1412–1417. [CrossRef]
37. Liu, S.; Zhao, Y. An efficient algorithm for globally solving generalized linear multiplicative programming. *J. Comput. Appl. Math.* **2016**, *296*, 840–847. [CrossRef]
38. Shao, L.; Ehrgott, M. Primal and dual multi-objective linear programming algorithms for linear multiplicative programmes. *Optimization* **2016**, *65*, 415–431. [CrossRef]
39. Peiping, S.; Lufan, W. A Fully Polynomial Time Approximation Algorithm for Generalized Linear Multiplicative Programming. *Math. Appl.* **2018**, *31*, 208–213.
40. Benson, H.P. Decomposition branch-and-bound based algorithm for linear programs with additional multiplicative constraints. *J. Optim. Theory Appl.* **2005**, *126*, 41–61. [CrossRef]
41. Wang, C.F.; Liu, S.Y. A new linearization method for generalized linear multiplicative programming. *Comput. Oper. Res.* **2011**, *38*, 1008–1013. [CrossRef]
42. Shen, P.; Huang, B. Global algorithm for solving linear multiplicative programming problems. *Optim. Lett.* **2019**, *2019*, 1–18. [CrossRef]
43. Sahinidis, N. BARON User Manual v.19.7.13 [EB/OL]. 2019. Available online: http://minlp.com (accessed on 7 November 2019).
44. Liu, X.; Gao, Y.L.; Zhang, B.; Tian, F.P. A New Global Optimization Algorithm for a Class of Linear Fractional Programming. *Mathematics* **2019**, *7*, 867. [CrossRef]