An Effective Global Computational Algorithm for a class of Generalized Linear Multiplicative Programs

Bo Zhang[a], YueLin Gao[b,*], Xia Liu[a], XiaoLi Huang[c]

[a] School of Mathematics and Statistics, Ningxia University, Yinchuan, 750021, P.R. China

[b] Ningxia province key laboratory of intelligent information and data processing, North Minzu University, Yinchuan, 750021, P.R. China

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

Abstract

This paper explains a region-division-linearization algorithm for solving a class of generalized linear multiplicative programs (GLMP) with exponent. In this algorithm, the original non-convex problem (GLMP) is transformed into a series of linear programming problems by dividing the outer space of the problem (GLMP) into finite polynomial rectangles. A new two-stage acceleration technique is put in place to improve the computational efficiency of the algorithm, which removes part of the region of the optimal solution without problems (GLMP) in outer space. In addition, the global convergence of the algorithm is discussed, and the computational complexity of the algorithm is investigated. It demonstrates that the algorithm is a completely polynomial time approximation scheme. Finally, the numerical results show that the algorithm is effective and feasible.

Keywords: Global optimization, Generalized linear multiplicative programs, Approximation algorithm, Computational complexity, region-division-linearization

1. Introduction

Consider a class of generalized linear multiplicative programs(GLMP),

$$(\mathrm{LFP}): \begin{cases} \min\ f(x) = \prod_{i=1}^{p} (c_i^T x + d_i)^{\alpha_i}, \\ s.t.\ \ x \in \mathrm{X} = \{x \in \mathbb{R}^n | Ax \leq b, x \geq 0\}. \end{cases}$$

Here, $p \geq 2$, X is a non-empty bounded closed set, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, c_i, $e_i \in \mathbb{R}^n$ and $d_i, f_i \in \mathbb{R}$, $\alpha_i \geq 0, c_i^T x + d_i > 0, i = 1, \ldots, p$.

The problem (GLMP) usually has multiple non-global local optimal solutions and is a class of NP-hard problems [1], which can be widely used in the fields of finance optimization [2, 3], robust optimization [4], microeconomics [5], and multi-objective decision making [6, 7]. In addition, the (GLMP) includes a wide range of mathematical program categories, such as linear multiplicative programming, quadratic programming, bilinear programming, and more. Therefore, for these and various other reasons, (GLMP) has caught

Email addresses: zbsdx121@163.com (Bo Zhang[a]), gaoyuelin@263.net (YueLin Gao[b,*]), lingxiaoyu911@163.com (Xia Liu[a]), hxl1569501@163.com. (XiaoLi Huang[c])

* Author to whom any correspondence should be addressed.

the attention of many experts, scholars and engineering practitioners who have studied this theory, and set off a new wave of global optimization learning. With the increasing dependence of practical problems on modeling optimization, local optimization theory and global optimization algorithms have made remarkable progress. However, the algorithm theory of global optimization algorithm is still quite insufficient relative to local optimization algorithm. There are many methods to study this kind of problems, such as level set algorithm [8], heuristic algorithm [9, 10], branch and bound algorithm [11–13], outer approximation algorithm [14], parametric simplex algorithm [15] and so on, but these methods do not give the computational complexity of the algorithm. Furthermore, Daniele and Marco [16] consider the problem of minimizing the product of two affine functions on a polyhedron set and propose a full polynomial time approximation algorithm. Marco [17] gives an approximate algorithm to solve more general types of global optimization problems, and give a computational complexity analysis, but lack of numerical results of the algorithm. Recently, Shen and Wang [18] also proposed a full polynomial time approximation algorithm for resolving the problem (GLMP) globally, but there is no acceleration technique. Moreover, for a more comprehensive overview of the (GLMP), we encourage readers to mention the more detailed literature [8, 19–21].

In this paper, two approximation algorithms are proposed for the (GLMP) , mainly by establishing a non-uniform mesh to transform the process of solving the original problem into a series of linear solver problem solving process and prove that the proposed algorithm can obtain a global $\epsilon-$approximation solution for the problem. Besides, we put forward a two-stage acceleration technique to speed up Algorithm I, which yields Algorithm II. Then, by discussing the computational complexity of the algorithm, it is shown that the two algorithms are polynomial time approximation algorithms. The problem (GLMP) considered in this paper not only generalizes the model proposed in [16] , but also the results of numerical experiments show that at least Algorithm II outperforms the algorithm in [17, 18] , both in CPU running time and iterations.

The rest of this paper will be paid below. In section 2, we first transform the problem (GLMP) into its equivalent optimization problem (EOP) and give its region-decomposition-linearization technique. Section 3 presents the global $\epsilon-$approximation algorithm for obtaining the problem (GLMP) and the convergence of the proposed algorithm. In section 4, we give the computational complexity of the proposed algorithm and carry out some numerical experiments in section 5 to verify the feasibility and effectiveness of the algorithm. The concluding section is a simple summary.

2. Equivalence problem and its linearization technique

In this section, we will give some properties of the equivalence optimization problem (EOP) of the problem (GLMP) and its objective function, then we will give the linearization technique of the equivalence problem.

2.1. Equivalent problems and their properties

To solve the problem (GLMP), the definition of global $\epsilon-$approximation solution is given below.

Definition 2.1. *Let x^* be a global optimal solution to the problem (GLMP) at a given precision $\epsilon \in (0,1)$. If the $\hat{x} \in X$ satisfies $f(\hat{x}) \leq (1+\epsilon)f(x^*)$, the $\hat{x}$ is referred to as the global approximation of the problem (GLMP).*

To obtain the global $\epsilon-$approximation solution of the problem (GLMP), let $f_i(x) = c_i^T x + d_i$, $l_i = \min\limits_{x\in X} f_i(x)$.

Theorem 2.1. *For each $i = 1, 2, \cdots, p$, let $\tilde{x}^i = arg\min\limits_{x\in X} f_i(x)$, $Q = \bigcup\limits_{i=1}^{p} \tilde{x}^i$, $\check{x} = arg\min\limits_{x\in Q} f(x)$, $\tilde{U} = f(\check{x})$. And then, for each $i \in \{1, 2, \cdots, p\}$, let $M_i = \prod\limits_{j=1, j\neq i}^{p} l_j^{\alpha_j}$, then $f_i(x^*) \leq u_i$ with $u_i = (\frac{\tilde{U}}{M_i})^{\frac{1}{\alpha_i}}$.*

Proof. This is easy to know, for any $i \in \{1, 2, \cdots, p\}$, there are $l_i \leq f_i(x^*)$, thus

$$\prod_{j=1, j\neq i}^{p} l_j^{\alpha_j} (f_i(x^*))^{\alpha_i} \leq \prod_{i=1}^{p} (f_i(x^*))^{\alpha_i} = f(x^*) \leq f(\check{x}) = \tilde{U}.$$

Therefore, $f_i(x^*) \leq (\frac{\tilde{U}}{M_i})^{\frac{1}{\alpha_i}} = u_i$ and then the conclusion holds. □

Next, according to the Theorem 2.1, for any $i = 1, 2, \cdots, p$, $u_i = (\frac{\tilde{U}}{M_i})^{\frac{1}{\alpha_i}}$ provide an upper bound for each $f_i(x^*)$.

On the basis of the above definition of l_i and u_i, define the rectangle H as follows.

$$H = [l_1, u_1] \times [l_2, u_2] \times \cdots \times [l_p, u_p].$$

Moreover, the rectangle H is also called the problem (GLMP) of the outer space. Thus, by introducing variable $y = (y_1, y_2, \cdots, y_p)^T \in H$, the problem (GLMP) is equivalent to the following problem (P1).

$$\text{(P1)} \quad \begin{cases} \min \ \ h(y) = \prod\limits_{i=1}^{p} y_i^{\alpha_i}, \\ s.t. \ \ f_i(x) \leq y_i, i = 1, 2, \cdots, p \\ \quad x \in X, y \in H. \end{cases}$$

Next, the equivalence of problems (GLMP) and (P1) is explained by Theorem 2.1.

Theorem 2.2. *x^* is the global optimal solution of the problem* (GLMP) *if and only if (x^*, y^*) is the optimal solution of the* (P1) *and $y_i^* = f_i(x^*), i = 1, 2, \cdots, p$.*

Proof. Let $y_i^* = f_i(x^*), i = 1, 2, \cdots, p$ if x^* is a global optimal solution of the problem (GLMP). And then it is obvious that (x^*, y^*) is a feasible solution to (P1). Suppose the (x^*, y^*) is not the optimal solution of (P1), then there is at least one feasible solution $(\bar{x}, \bar{y})$ of (P1), which makes

$$f(\bar{x}) = \prod_{i=1}^{p} (f_i(\bar{x}))^{\alpha_i} \leq \prod_{i=1}^{p} \bar{y}_i^{\alpha_i} < \prod_{i=1}^{p} (y_i^*)^{\alpha_i} = \prod_{i=1}^{p} (f_i(x^*))^{\alpha_i} = f(x^*),$$

which contradicts the optimality of the x^*, so the hypothesis does not hold, and then (x^*, y^*) is an optimal solution of (P1).

On the contrary, if (x^*, y^*) is an optimal solution of (P1), if there is a $i \in \{1, 2, \cdots, p\}$ that makes $f_i(x^*) < y_i^*$, let $\tilde{y}_i = f_i(x^*)$, then $(x^*, \tilde{y})$ is a feasible solution of (P1) and

$$\prod_{i=1}^{p} \tilde{y}_i^{\alpha_i} < \prod_{i=1}^{p} (y_i^*)^{\alpha_i},$$

which contradicts the optimality of (x^*, y^*), so $y_i^* = f_i(x^*), i = 1, 2, \cdots, p$. Suppose x^* is not the global optimal solution of the problem (GLMP), then there must be a $\bar{x} \in \mathrm{X}$ that makes $f(\bar{x}) < f(x^*)$. Let $\bar{y}_i = f_i(\bar{x})$, obviously $(\bar{x}, \bar{y})$ a feasible solution to (P1), so we have

$$\prod_{i=1}^{p} (\bar{y}_i)^{\alpha_i} = f(\bar{x}) < f(x^*) = \prod_{i=1}^{p} (y_i^*)^{\alpha_i},$$

which contradicts the optimality of the (x^*, y^*). Therefore, x^* is the global optimal solution of the problem (GLMP), which proves to be completed. □

It is easy to understand from Theorems 2.2 that the problem (GLMP) and (P1) are equivalent and have the same global optimal value.

Then, for a given $y \in H$, define the set

$$D(y) = \{x \in \mathrm{X} | f_i(x) \leq y_i, i = 1, 2, \cdots, p\}$$

and function

$$g(y) = \begin{cases} h(y), & D(y) \neq \emptyset, \\ +\infty, & D(y) = \emptyset. \end{cases}$$

Then the problem (P1) is equivalent to the following equivalent optimization problem.

$$(\mathrm{EOP}) \quad \begin{cases} \min \ \ g(y), \\ s.t. \ \ y \in H. \end{cases}$$

Theorem 2.3. *y^* is the global optimal solution of the problem* (EOP) *if and only if (x^*, y^*) is the optimal solution of the* (P1) *and $y_i^* = f_i(x^*), i = 1, 2, \cdots, p$.*

Proof. Suppose (x^*, y^*) is an optimal solution of (P1), then according to Theorem 2.3, we can know $y_i^* = f_i(x^*), i = 1, 2, \cdots, p$ and $y^* \in H$. In addition, $h(y^*) = g(y^*) = \prod_{i=1}^{p} (y_i^*)^{\alpha_i}$. Suppose that y^* is not the global optimal solution of the problem (EOP), there must be a $\bar{y} \in H$ such that $g(\bar{y}) < g(y^*)$ and $D(\bar{y}) \neq \emptyset$, then there must also be a $\bar{x} \in D(\bar{y})$ such that $f_i(\bar{x}) \leq \bar{y}_i, i = 1, 2, \cdots, p$. Then, $(\bar{x}, \bar{y})$ is a feasible solution of the (P1), there is $h(\bar{y}) = g(\bar{y}) < g(y^*) = h(y^*)$, which contradicts the optimality of (x^*, y^*), so the hypothesis does not hold, so y^* is the global optimal solution of the problem.

On the other hand, if y^* is a global optimal solution of the problem (EOP), then $D(y^*) \neq \emptyset$, there must be a $x^* \in D(y^*)$ such that (x^*, y^*) is a feasible solution of the (P1). Suppose (x^*, y^*) is not the global optimal solution of the problem (P1), Then there must be an optimal solution $(\bar{x}, \bar{y})$ to the problem (P1) such that $h(\bar{y}) < h(y^*), \bar{y}_i = f_i(\bar{x}), i = 1, 2, \cdots, p$, So $D(\bar{y}) \neq \emptyset$ and $g(\bar{y}) = h(\bar{y}) < h(y^*) = g(y^*)$, which contradicts the fact that y^* is the global optimal solution of the problem (EOP), Therefore, (x^*, y^*) is the global optimal solution of (P1), and $y_i^* = f_i(x^*), i = 1, 2, \cdots, p$ can be obtained from Theorem 2.2 and then proved to be over. □

Through Theorem 2.3, the problem (EOP) and (P1) has the same global optimal value, so combined with Theorem 2.2, the problem (EOP) and (GLMP) is also equivalent. Therefore, we can solve the equivalent problem (EOP) instead of addressing the problem (GLMP).

Next, we note the following linear programming problem.

$$(\mathrm{LP}_y) \quad \begin{cases} \min \sum_{i=1}^{p} \dfrac{\alpha_i f_i(x)}{y_i} \\ s.t.\ x \in D(y). \end{cases},$$

If $D(y) \neq \emptyset$, the optimal solution to the problem (LP_y) is recorded as x_y, and let $\tilde{y}_i = f_i(x_y)$, $\rho = \sum_{i=1}^{p} \alpha_i > 0$, then

$$\rho = \sum_{i=1}^{p} \frac{\alpha_i y_i}{y_i} \geq \sum_{i=1}^{p} \frac{\alpha_i f_i(x)}{y_i}, \forall x \in D(y).$$

Furthermore, according to the Jensen inequality, we have

$$\sum_{i=1}^{p} \frac{\alpha_i f_i(x_y)}{y_i} \geq \rho(\prod_{i=1}^{p}(\frac{f_i(x_y)}{y_i})^{\alpha_i})^{\frac{1}{\rho}} = \rho(\frac{g(\tilde{y})}{g(y)})^{\frac{1}{\rho}},$$

then

$$\rho \geq \rho(\frac{g(\tilde{y})}{g(y)})^{\frac{1}{\rho}},\ \ g(\tilde{y}) \leq g(y). \tag{1}$$

Theorem 2.4. *Suppose $x^* \in$ X is the global optimal solution of the original problem* (GLMP)*, let $y_i^* = f_i(x^*), i = 1, 2, \cdots, p$, then $y^* = (y_1^*, y_2^*, \cdots, y_p^*)^T \in H$ and x^* is also the global optimal solution of the problem* (LP_{y^*}).

Proof. Firstly, according to Theorems 2.3 and 2.4, we know that y^* is a global optimal solution of the problem (EOP). Then, by using formula (1) and the optimality of the global optimal solution y^* of the (EOP), we can see that the x^* is an optimal solution of the problem (LP_{y^*}). □

Next, the properties of the function $g(y)$ on the H are given by Theorem 2.5.

Theorem 2.5. *For a given precision $\epsilon \in (0, 1)$, let $\delta = (1+\epsilon)^{\frac{1}{\rho}}$, then for any $\bar{y} \in H$, there is*

$$g(\bar{y}) \leq (1+\epsilon) g(y),\ \ \forall\, y \in [\frac{\bar{y}}{\delta}, \bar{y}]. \tag{2}$$

In addition, if $D(\bar{y}) \neq \emptyset$, the optimal solution to the problem $(\mathrm{LP}_{\bar{y}})$ *is recorded as $\bar{x}$, then let $\tilde{y}_i = f_i(\bar{x}) (i = 1, 2, \cdots, p)$, there is also*

$$g(\tilde{y}) \leq g(\bar{y}) \leq (1+\epsilon) g(y),\ \ \forall\, y \in [\frac{\bar{y}}{\delta}, \bar{y}]. \tag{3}$$

Proof. For all $\bar{y} \in H$, according to the definition of $D(y)$ and $\delta = (1+\epsilon)^{\frac{1}{\rho}} > 1$, one can know $D(\frac{\bar{y}}{\delta}) \subseteq D(\bar{y})$.

If $D(\frac{\bar{y}}{\delta}) \neq \emptyset$, for any $y \in [\frac{\bar{y}}{\delta}, \bar{y}]$, we have $D(y) \neq \emptyset$, obviously $g(\tilde{y}) \leq g(\bar{y})$ and $y_i \geq \frac{\bar{y}_i}{\delta}$ for each $i = 1, 2, \cdots, p$. Thus,

$$\prod_{i=1}^{p}(\frac{\bar{y}_i}{\delta})^{\alpha_i} \leq \prod_{i=1}^{p} y_i^{\alpha_i}. \tag{4}$$

Moreover, according to the definition of function $g(y)$, $g(y) = \prod_{i=1}^{p} y_i^{\alpha_i}$, thus

$$g(\frac{\bar{y}}{\delta}) = \prod_{i=1}^{p}(\frac{\bar{y}_i}{\delta})^{\alpha_i} = \frac{1}{\delta^\rho} \prod_{i=1}^{p} \bar{y}_i^{\alpha_i} = \frac{1}{\delta^\rho} g(\bar{y}). \tag{5}$$

And so in combination with the formula (4), (5), we have

$$g(y) \geq g(\frac{\bar{y}}{\delta}) = \frac{1}{\delta^\rho} g(\bar{y}),\ \forall\, y \in [\frac{\bar{y}}{\delta}, \bar{y}]. \tag{6}$$

Further, through the formula (6) and combined with the definition of δ, we can understand that the formula (3) is formed, the formula (2) is of course also true.

If $D(\frac{\bar{y}}{\delta}) = \emptyset$, $D(\bar{y}) \neq \emptyset$, it is clear that the inequality $g(\tilde{y}) \leq g(\bar{y})$ is established.

For all $y \in [\frac{\bar{y}}{\delta}, \bar{y}]$, if $D(y) \neq \emptyset$, we have $y_i \geq \frac{\bar{y}_i}{\delta} (i = 1, 2, \cdots, p)$, and $y \neq \frac{\bar{y}}{\delta}$, so

$$\prod_{i=1}^{p} (\frac{\bar{y}_i}{\delta})^{\alpha_i} \leq g(y) = \prod_{i=1}^{p} y_i^{\alpha_i}. \tag{7}$$

Furthermore,

$$g(\bar{y}) = \prod_{i=1}^{p} \bar{y}_i^{\alpha_i} = \delta^{\rho} \prod_{i=1}^{p} (\frac{\bar{y}_i}{\delta})^{\alpha_i}, \tag{8}$$

By using the definition of δ and formula (7)-(8), one can infer that formula (2)-(3) holds.

If $D(y) = \emptyset$, and therefore $g(y) = +\infty$, then the formula (2)-(3) is obviously hold.

If $D(\bar{y}) = \emptyset$, and the problem ($\mathrm{LP}_{\bar{y}}$) is not solved, and for any $y \in [\frac{\bar{y}}{\delta}, \bar{y}]$, there is $D(y) = \emptyset$, then $g(y) = +\infty$, so the formula (2) is clearly established and the proof of the conclusion is completed. □

Theorem 2.5 shows that for any $\bar{y} \in H$, we can determine whether the $D(\bar{y})$ is not empty by solving the linear programming problem ($\mathrm{LP}_{\bar{y}}$), and then determine whether formula (3) holds.

2.2. Linearization techniques

The objective function of the problem (EOP) is still non-convex compared to the problem (GLMP). But the space H in which the variable y of the objective function is located is p dimensions. Therefore, based on the above discussion, in order to solve the (EOP), for the given $\epsilon \in (0, 1)$, we first split the outer space H on each dimension at a ratio of $\delta = (1 + \epsilon)^{\frac{1}{\rho}}$, thus producing several small rectangles.

To do this, let

$$\gamma_i = arg \max\{\sigma \in \mathbb{N} | l_i \delta^{\sigma} \leq u_i\}, i = 1, 2, \cdots, p, \tag{9}$$

where $\mathbb{N}$ represents a non-negative integer set. Therefore, the number of these small rectangles is finite, and the set of all their vertices is

$$B^{\delta} = \{\nu_1, \nu_2, \cdots, \nu_p | \nu_i \in P_i^{\delta}, i = 1, 2, \cdots, p\}, \tag{10}$$

where $P_i^{\delta} = \{l_i, l_i\delta, \cdots, l_i\delta^{\gamma_i}\}$. Obviously, for each $y \in H$, there must be a vertex $(\nu_1, \nu_2, \cdots, \nu_p) \in B^{\delta}$ making $y_i \in [\nu_i, \delta\nu_i]$, $i = 1, 2, \cdots, p$. Then it can be concluded that the rectangle H can be approximated by the set B^{δ}.

Next, by using the set B^{δ}, the process of solving the problem (EOP) can be transformed into solving a series of subproblems. To this end, for each $\nu \in B^{\delta}$, we need to consider the value of the $g(\nu)$, that is, to determine whether the set $D(\nu)$ is not empty. According to Theorem 2.5, we can determine whether the $D(\nu)$ is not empty by solving the linear programming problem (LP_{ν}). Therefore, for each vertex $\nu \in B^{\delta}$, the following linear programming subproblem needs to be solved here, that is,

$$(\mathrm{LP}_{\nu}) \quad \begin{cases} \min \sum_{i=1}^{p} \frac{\alpha_i f_i(x)}{\nu_i} \ , \\ s.t. \ f_i(x) \leq \nu_i, i = 1, 2, \cdots, p, \\ \quad x \in \mathrm{X}. \end{cases}$$

On the basis of the conclusion of Theorem 2.5, if the problem (LP_ν) can be solved (its solution is recorded as x_ν), then

$$\tilde{v} = (f_1(x_\nu), f_2(x_\nu), \cdots, f_p(x_\nu))^T \in H, \tag{11}$$

thus,

$$g(\tilde{v}) \leq g(\nu) \leq (1+\epsilon)g(y), \ \ \forall\, y \in [\frac{\nu}{\delta}, \nu].$$

3. Analysis of algorithm and its computational complexity

This section brings an approximate algorithm based on linearization-decomposition to solve the problem (EOP). After that, the analysis of its computational complexity is proved accordingly.

3.1. Approximate algorithm

To solve the (EOP), we subdivide the external space H into a finite number of small rectangles with ratio δ, and put all the vertices of these small rectangles into the set B^δ.

Then for each vertex $\nu \in B^\delta$, solve the linear programming problem (LP_ν), and if $D(\nu) \neq \emptyset$, a feasible solution $\tilde{v}$(formula (11)) to the (EOP) can be obtained according to its optimal solution x_ν, which makes

$$g(\tilde{v}) \leq g(\nu) \leq (1+\epsilon)g(y), \ \ \forall\, y \in [\frac{\nu}{\delta}, \nu].$$

If there is a $\tilde{v}$ that satisfies $g(\tilde{v}) \leq (1+\epsilon)g(y^*)$, and then

$$f(x_\nu) = \prod_{i=1}^{p}(f_i(x_\nu))^{\alpha_i} = \prod_{i=1}^{p}\tilde{v}_i^{\alpha_i} = g(\tilde{v}) \leq (1+\epsilon)g(y^*) = (1+\epsilon)f(x^*),$$

then ,x_ν is a global $\epsilon-$approximation solution of the problem (GLMP). The specific algorithm steps are as follows.

Algorithm I

Step 0 (Initialization). Set $\epsilon \in (0,1), \delta = (1+\epsilon)^{\frac{1}{p}}, F = +\infty, k = 0$. By using formulas (9)-(10), the H is subdivided into smaller rectangles, such that the ratio of two consecutive segments is δ in each dimension. Represents the vertex of each small rectangle as $\nu = (\nu_1, \nu_2, \cdots, \nu_p)$, which is stored in the set B^δ.

Step 1. Select a point ν from the B^δ, solve the linear programming problem (LP_ν), let $B^\delta = B^\delta \backslash \nu$.

Step 2. If the problem (LP_ν) is solvable, then $D(\nu) \neq \emptyset$, let $g(\nu) = \prod_{i=1}^{p}(\nu_i)^{\alpha_i}$, if $g(\nu) < F$, let $F = g(\nu)$, $\bar{\nu} = \nu$, $x_{\bar{\nu}} = x_\nu$; If $B^\delta \neq \emptyset$, set $k = k+1$ and go to Step 1, otherwise, the algorithm terminates, let

$$\tilde{v}_i = f_i(x_{\bar{\nu}}), \ i = 1, 2, \cdots, p, \tilde{v} = (\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_p)^T,$$

then $x_{\bar{\nu}}$, $\tilde{v}$ is a global $\epsilon-$approximation solution to the problem (GLMP) and the (EOP), respectively.

Theorem 3.1. *For a given precision $\epsilon \in (0,1)$, let $\delta = (1+\epsilon)^{\frac{1}{p}}$, $\bar{\nu} = arg\min\{g(\nu)|\nu \in B^\delta\}$, $x_{\bar{\nu}}$ is the optimal solution to the linear programming problem* ($LP_{\bar{\nu}}$). *Then, Algorithm I will get a global $\epsilon-$approximation solution $x_{\bar{\nu}}$ of the problem* (GLMP), *i.e.*

$$f(x_{\bar{\nu}}) \leq (1+\epsilon)f(x^*),$$

where, x^* is the global optimal solution to the original problem (GLMP).

Proof. Let

$$y_i^* = f_i(x^*),\ i = 1, 2, \cdots, p.$$

According to the Theorem 2.1, we have

$$l_i \leq y_i^* \leq u_i,\ i = 1, 2, \cdots, p. \tag{12}$$

Then formula (12) implies that $y^* = (y_1^*, y_2^*, \cdots, y_p^*)^T \in H$, so there must be a $\nu^* \in B^\delta$ which makes

$$\frac{\nu_i^*}{\delta} \leq y_i^* \leq \nu_i^*,\ i = 1, 2, \cdots, p.$$

So, using Theorem 2.5 on the small rectangle $[\frac{\nu^*}{\delta}, \nu^*]$, there is

$$f(x^*) = \prod_{i=1}^{p}(y_i^*)^{\alpha_i} = g(y^*) \geq \prod_{i=1}^{p}(\frac{\nu_i^*}{\delta})^{\alpha_i} = (\frac{1}{\delta})^{\sum_{i=1}^{p}\alpha_i}\prod_{i=1}^{p}(\nu_i^*)^{\alpha_i} = \frac{1}{\delta^\rho}g(\nu^*).$$

Thus,

$$\delta^\rho f(x^*) = \delta^\rho g(y^*) \geq g(\nu^*). \tag{13}$$

Noting that $\bar{\nu} = arg\min\{g(\nu)|\nu \in B^\delta\}$, we can know

$$g(\nu^*) \geq g(\bar{\nu}). \tag{14}$$

Since $x_{\bar{\nu}}$ is the optimal solution to the linear programming problem ($LP_{\bar{\nu}}$), let

$$\tilde{\nu}_i = f_i(x_{\bar{\nu}}),\ i = 1, 2, \cdots, p.$$

Apparently, $\tilde{\nu} = (\tilde{\nu}_1, \tilde{\nu}_2, \cdots, \tilde{\nu}_p) \in H$. So, by taking advantage of the formula (3) in Theorem 2.5, we have

$$g(\bar{\nu}) \geq g(\tilde{\nu}) = \prod_{i=1}^{p}(\tilde{\nu}_i)^{\alpha_i} = \prod_{i=1}^{p}(f_i(x_{\bar{\nu}}))^{\alpha_i} = f(x_{\bar{\nu}}). \tag{15}$$

Therefore, by integrating the formula (13)-(15) and combining the $\delta = (1+\epsilon)^{\frac{1}{\rho}}$, we can obtain

$$f(x_{\bar{\nu}}) \leq (1+\epsilon)f(x^*)$$

and this proof is completed. □

Remark 3.2. According to Theorem 3.1, if $y^* \in B^\delta$, then from Theorem 2.5, the optimal solution x_{y^*} of the linear programming problem (LP_{y^*}) is exactly the global optimal solution of the original problem (GLMP).

Through the Theorem 3.1, we can see that for a given precision $\epsilon \in (0, 1)$, Algorithm I will obtain a global $\epsilon-$approximation solution to the problem (GLMP). Moreover, the Remark 3.2 also shows that if $y^* \in B^\delta$, then Algorithm I will find a global optimal solution of the problem (GLMP) exactly.

3.2. Accelerating techniques

Algorithm I shows that, for any $\nu \in B^\delta$, it is required to solve the linear programming problem (LP_ν) , in order to verify that the $D(\nu)$ is non-empty. Hence, the computational cost of Algorithm I depends on the number of points within the set B^δ, respectively. Then, the proposal of the acceleration technique will

discard some points that are not necessary to consider the set B^δ, and only consider the region that contains the global optimal solution of the problem (EOP). The detailed process is paid below.

If $\bar{v}$ is the best known solution to the problem (EOP), $x_{\bar{v}}$ is the optimal solution to the linear programming problem ($LP_{\bar{v}}$), for each $i = 1, 2, \cdots, p$, let $\tilde{v}_i = f_i(x_{\bar{v}})$, $\tilde{v} = (\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_p)^T$, obviously $g(\tilde{v}) \leq g(\bar{v})$, then $\tilde{v}$ may be a better solution than $\bar{v}$. Well, using $\tilde{v}$ may be able to remove more vertexes from the B^δ that do not need to be explored. To give the acceleration technique for Algorithm I, we first need to specify a necessary condition that the points in each sub-rectangle $H^k \subseteq H^0 = H (k \geq 1)$ containing the global optimal solution of the problem (EOP) must be satisfied, that is,

$$\prod_{i=1}^{p} l_i^{\alpha_i} \leq g(y^*) \leq g(y) \leq g(\tilde{v}), \ \forall \ y \in H^k, \tag{16}$$

where $H^k = [l, u^k]$, $u^k = (u_1^k, u_2^k, \cdots, u_p^k)^T$, $u_i^k \leq u_i^{k-1} \leq u_i, i = 1, 2, \cdots, p$. Similarly, if $\delta = (1+\epsilon)^{\frac{1}{\rho}}$ are used to segment rectangles H^k on each dimension, this will produce a limited number of small rectangles. For this purpose, let

$$\gamma_i^k = arg\max\{\sigma \in \mathbb{N} | l_i \delta^\sigma \leq u_i^k\}, i = 1, 2, \cdots, p.$$

Then, a set of vertices of a finite number of small rectangles will also be generated on a rectangular H^k, that is,

$$B_k^\delta = \{\nu_1, \nu_2, \cdots, \nu_p | \nu_i \in P_{ki}^\delta, i = 1, 2, \cdots, p\},$$

where $P_{ki}^\delta = \{l_i, l_i\delta, \cdots, l_i\delta^{\gamma_i^k}\}$. Clearly, $B_k^\delta \subseteq B_0^\delta = B^\delta$ and $B_k^\delta \subset H^k \subseteq H_0 = H$.

Based on the above discussion, we will give propositions 3.3 and 3.4 to clarify the acceleration techniques of the algorithm.

Proposition 3.3. *The global optimal solution of the problem* (EOP) *can not be obtained on the set* $\bar{B}_{ki}^\delta$ *if a* $i \in \{1, 2, \cdots, p\}$ *makes* $(\frac{g(\tilde{v})}{M_i})^{\frac{1}{\alpha_i}} < l_i\delta^{\gamma_i^k}$, *of which*

$$\bar{B}_{ki}^\delta = \{\nu \in B_k^\delta | (\frac{g(\tilde{v})}{M_i})^{\frac{1}{\alpha_i}} < \nu_i\}, i \in \{1, 2, \cdots, p\}.$$

Proof. If $\nu \in \bar{B}_{ki}^\delta$, then there must be $(\frac{g(\tilde{v})}{M_i})^{\frac{1}{\alpha_i}} < \nu_i \leq l_i\delta^{\gamma_i^k}$, and thus there is

$$g(\tilde{v}) = ((\frac{g(\tilde{v})}{M_i})^{\frac{1}{\alpha_i}})^{\alpha_i} M_i < (\nu_i)^{\alpha_i} M_i = (\nu_i)^{\alpha_i} \prod_{j=1, j\neq i}^{p} l_j^{\alpha_j} \leq \prod_{j=1}^{p} (\nu_j)^{\alpha_j} = g(\nu),$$

which contradicts the inequality chain (16), so the conclusion is valid. □

With this proposition 3.3, we generate a new rectangle H^{k+1} and vertex set B_{k+1}^δ, i.e. for each $i = 1, 2, \cdots, p$, let

$$u_i^{k+1} = \begin{cases} (\frac{g(\tilde{v})}{M_i})^{\frac{1}{\alpha_i}}, & (\frac{g(\tilde{v})}{M_i})^{\frac{1}{\alpha_i}} < l_i\delta^{\gamma_i^k}, \\ u_i^k, & (\frac{g(\tilde{v})}{M_i})^{\frac{1}{\alpha_i}} \geq l_i\delta^{\gamma_i^k}, \end{cases} \tag{17}$$

as well as

$$\gamma_i^{k+1} = \begin{cases} arg\max\{\sigma \in N | l_i\delta^\sigma \leq u_i^{k+1}\}, & (\frac{g(\tilde{v})}{M_i})^{\frac{1}{\alpha_i}} < l_i\delta^{\gamma_i^k}, \\ \gamma_i^k, & (\frac{g(\tilde{v})}{M_i})^{\frac{1}{\alpha_i}} \geq l_i\delta^{\gamma_i^k}. \end{cases} \tag{18}$$

where N represents a non-negative integer set. Well, $u^{k+1} = [l, u^{k+1}]$ with $u^{k+1} = (u_1^{k+1}, u_2^{k+1}, \cdots, u_p^{k+1})$.

Moreover, the above rules may produce a small rectangular vertex set B_{k+1}^{δ} with relatively few new elements, but there is still $\tilde{v} \in B_{k+1}^{\delta}$, so we then give proposition 3.4 to delete the other unconsidered elements in the B_{k+1}^{δ}.

Proposition 3.4. *If $\bar{v}$ is the best known solution to the problem* (EOP)*, $x_{\bar{v}}$ is the optimal solution to the linear planning problem* ($\mathrm{LP}_{\bar{v}}$)*, for each $i = 1, 2, \cdots, p$, let $\tilde{v}_i = f_i(x_{\bar{v}})$, $\tilde{v} = (\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_p)^T$, define the set*

$$\bar{B}_{k+1}^{\delta} = \{\nu \in B_{k+1}^{\delta} | \tilde{v}_i \leq \nu_i, i = 1, 2, \cdots, p\},$$

then, for any $\nu \in \bar{B}_{k+1}^{\delta}$, the (EOP) *can not get a better solution than $\tilde{v}$.*

Proof. Since $x_{\bar{v}}$ is the optimal solution to a linear programming problem ($\mathrm{LP}_{\bar{v}}$), then there is at least one point $x_{\bar{v}}$ in the set $D(\tilde{v})$, so $D(\tilde{v}) \neq \emptyset$. For arbitrary $\nu \in \bar{B}_{k+1}^{\delta}$, Obviously $D(\tilde{v}) \subseteq D(\nu)$, thus $D(\nu) \neq \emptyset$. According to the definition of the function $g(y)$, for each $\nu \in \bar{B}_{k+1}^{\delta}$, the objective function value of the (EOP) meet

$$g(\nu) = \prod_{i=1}^{p} (\nu_i)^{\alpha_i} \geq \prod_{i=1}^{p} (\tilde{v}_i)^{\alpha_i} = g(\tilde{v})$$

and this conclusion is proved. □

Next, for a given $\epsilon \in (0, 1)$, $\delta = (1 + \epsilon)^{\frac{1}{p}}$, make use of proposition 3.4, let

$$\tau_i^{k+1} = arg \min\{\sigma \in \mathbb{N} | \tilde{v}_i \leq l_i \delta^{\sigma} \leq u_i^{k+1}\}. \tag{19}$$

Through the expression of γ_i^{k+1} in (3.9), the set $\bar{B}_{k+1}^{\delta}$ is defined as follows.

$$\bar{B}_{k+1}^{\delta} = \{l_i \delta^{\sigma_1}, l_i \delta^{\sigma_2}, \cdots, l_i \delta^{\sigma_p} | \sigma_i \in \{\tau_i^{k+1}, \tau_i^{k+1} + 1, \cdots, \gamma_i^{k+1}\}, i = 1, 2, \cdots, p\}. \tag{20}$$

Therefore, for the convenience of narration, let $S_{k+1}^{\delta} = B_{k+1}^{\delta} \setminus \bar{B}_{k+1}^{\delta}$. This means that in order to obtain the global $\epsilon-$approximation solution of the problem (EOP), it is only necessary to calculate up to $|S_{k+1}^{\delta}|$ linear programming subproblems (LP_{ν}) to determine whether the $D(\nu)$ is not empty, which determines the function value $g(\nu)$ at each vertex $\nu \in S_{k+1}^{\delta}$. Then, by using the set S_{k+1}^{δ}, the computational efficiency of Algorithm I will be improved, leading to the following algorithm.

Algorithm II

Step 0 (Initialization). Set $\epsilon \in (0, 1), \delta = (1 + \epsilon)^{\frac{1}{p}}$. By using formulas (9)-(10), the $H^0 = H$ is subdivided into smaller rectangles, such that the ratio of two consecutive segments is δ in each dimension. Represents the vertex of each small rectangle as $\nu = (\nu_1, \nu_2, \cdots, \nu_p)$, which is stored in the set B^{δ}. Let $F = +\infty$, $T = \emptyset$, $B_0^{\delta} = B^{\delta}$, $\Xi^0 = B_0^{\delta}$, $k = 0$.

Step 1. Select a point $\nu = (\nu_1, \nu_2, \cdots, \nu_p)^T$ from the Ξ^k, solve the linear programming problem (LP_{ν}), let $T = T \cup \nu$.

Step 2. If the problem (LP_{ν}) is solvable, then $D(\nu) \neq \emptyset$, let $g(\nu) = \prod_{i=1}^{p} (\nu_i)^{\alpha_i}$, if $g(\nu) < F$, let $\bar{v} = \nu$, $x_{\bar{v}} = x_{\nu}$, $\tilde{v} = (\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_p)^T = (f_1(x_{\bar{v}}), f_2(x_{\bar{v}}), \cdots, f_p(x_{\bar{v}}))^T$, $F = g(\tilde{v})$; Using rules (17)-(18) to produce H^{k+1} and B_{k+1}^{δ}, and through formula (19)-(20) to obtain set $\bar{B}_{k+1}^{\delta}$, let $S_{k+1}^{\delta} = B_{k+1}^{\delta} \setminus \bar{B}_{k+1}^{\delta}$, $\Xi^k = S_{k+1}^{\delta} \setminus T$. If $\Xi^k \neq \emptyset$, set $k = k + 1$ and go to Step 1, otherwise, the algorithm terminates, let

$$\tilde{v}_i = f_i(x_{\bar{v}}),\ i = 1, 2, \cdots, p, \tilde{v} = (\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_p)^T,$$

then $x_{\tilde{v}}$, $\tilde{v}$ is a global $\epsilon-$approximation solution to the problem (GLMP) and the (EOP), respectively.

Note that the Algorithm II simply removes the set of vertices that do not contain a global optimal solution and, therefore, is similar to Theorem 3.1, the Algorithm II will also return a global $\epsilon-$approximation solution of the problem (GLMP) and (EOP) as well.

3.3. Analysis of computational complexity of the algorithm

We first give the following Lemma 3.1 to discuss the computational complexity of the two algorithms.

Lemma 3.1. *[22] Let λ be the maximum of the absolute values of all the elements A, b, c_i, d_i in problem* (GLMP)*, then each component x_j^0 of any pole x^0 of the X can be expressed as $x_j^0 = \frac{p_j}{q}$, where $0 \leq p_j \leq (n\lambda)^n$, $0 < q \leq (n\lambda)^n$, $j = 1, 2, \cdots, n$.*

Because for each $i = 1, 2, \cdots, p$, the solution $\tilde{x}^i$ to the linear programming problem $l_i = \min\limits_{x \in X} f_i(x)$ is the pole of X, thus by Lemma 3.1 ,we have $\tilde{x}_j^i = \frac{p_j^i}{q^i}$, where, $0 \leq p_j^i \leq (n\lambda)^n$, $0 < q^i \leq (n\lambda)^n$, $j = 1, 2, \cdots, n$. Thus, $l_i = \sum\limits_{j=1}^{n} c_{ij} \frac{p_j^i}{q^i} + d_i$, $i = 1, 2, \cdots, p$. Moreover, let

$$\tilde{q} = \max\{\frac{1}{q^i} |\ i = 1, 2, \cdots, p\},\ \omega = \min\{l_i |\ i = 1, 2, \cdots, p\}, \tag{21}$$

$$\tilde{U} = f(\check{x}) = \min_{1 \leq i \leq p} f(\tilde{x}^i), \tag{22}$$

for the sake of the following smooth description of Theorem 3.5, here $\check{x}$ be defined in Theorem 2.1.

Theorem 3.5. *For a given $p \geq 2$, in order to obtain a global $\epsilon-$approximation solution to the problem* (GLMP)*. The upper limit of the time required for the proposed Algorithm I is*

$$O\left(\left(\frac{2\tilde{\alpha}\rho^2}{\epsilon}[(n+1)\ln(n\lambda) - \ln\omega] + 1\right)^p \cdot T(m+p, n)\right),$$

where $\tilde{\alpha} = \max\{\frac{1}{\alpha_i} | i = 1, 2, \cdots, p\}$, $\rho = \sum\limits_{i=1}^{p} \alpha_i$, $T(m+p, n)$ represents the upper limit of the time used to solve a linear programming problem with $m + p$ linear constraints and n variables at a time.

Proof. From the formula(9)-(10) we can see that the maximum number of midpoint of the set B^δ is

$$\prod_{i=1}^{p} (\log_\delta \frac{u_i}{l_i} + 1). \tag{23}$$

Using the definition of $\tilde{q}, \omega$ in the formula (21) and the Lemma 3.1, we have

$$\omega \leq l_i \leq \tilde{q} n\lambda (n\lambda)^n + \lambda \leq 2\tilde{q}(n\lambda)^{n+1},\ i = 1, 2, \cdots, p. \tag{24}$$

Furthermore, we also have

$$\tilde{U} = \prod_{i=1}^{p} (c_i^T \check{x} + d_i)^{\alpha_i} \leq \prod_{i=1}^{p} \left(2\tilde{q}(n\lambda)^{n+1}\right)^{\alpha_i} = \left(2\tilde{q}(n\lambda)^{n+1}\right)^{\sum\limits_{i=1}^{p} \alpha_i}$$

by using the formula (22) and the above inequality (24). Of course, according to the definition of M_i and u_i in Theorem 2.1, and in conjunction with $\rho = \sum_{i=1}^{p} \alpha_i$, there will be

$$u_i = \left(\frac{\hat{U}}{M_i}\right)^{\frac{1}{\alpha_i}} \leq \left(2\tilde{q}(n\lambda)^{n+1}\right)\left(\frac{2\tilde{q}(n\lambda)^{n+1}}{\omega}\right)^{\frac{\rho}{\alpha_i}-1}. \tag{25}$$

By means of the above formulas (24) and (25), we can have

$$\frac{u_i}{l_i} \leq \left(\frac{2\tilde{q}(n\lambda)^{n+1}}{\omega}\right)^{\frac{\rho}{\alpha_i}},$$

thus,

$$\begin{aligned} \ln\frac{u_i}{l_i} &\leq \frac{\rho}{\alpha_i}[\ln 2\tilde{q} + (n+1)\ln(n\lambda) - \ln\omega] \\ &\leq \rho\tilde{\alpha}[\ln 2\tilde{q} + (n+1)\ln(n\lambda) - \ln\omega]. \end{aligned} \tag{26}$$

Using $\epsilon \in (0,1)$, $\delta = (1+\epsilon)^{\frac{1}{\rho}}$ in Algorithm I and $\frac{\epsilon}{2} < \ln(1+\epsilon) < \epsilon$, then there will be

$$\log_\delta \frac{u_i}{l_i} = \rho \log_{(1+\epsilon)} \frac{u_i}{l_i} = \rho\frac{\ln\frac{u_i}{l_i}}{\ln(1+\epsilon)} < \frac{2\rho\ln\frac{u_i}{l_i}}{\epsilon}. \tag{27}$$

Then, the upper limit of the number of internal points (expressed in $|B^\delta|$) of B^δ is

$$|B^\delta| \leq \left(\frac{2\tilde{\alpha}\rho^2}{\epsilon}[\ln 2\tilde{q} + (n+1)\ln(n\lambda) - \ln\omega] + 1\right)^p \tag{28}$$

in the utilized formula (23), (26)-(27). From the above formula (28), we can see that the running time of Algorithm I is at most

$$O\left(\left(\frac{2\tilde{\alpha}\rho^2}{\epsilon}[(n+1)\ln(n\lambda) - \ln\omega] + 1\right)^p \cdot T(m+p,n)\right)$$

when the global $\epsilon-$approximation solution is obtained, and then the proof of the conclusion is completed. □

Remark 3.6. By propositions 3.3 and 3.4, we can see that Algorithm II simply removes the vertices of a small rectangle that is not necessary to consider on the basis of Algorithm I, and acts as an acceleration Algorithm I. Then the upper bound of the CPU running time required by algorithm II is the same as that of Algorithm I in the most extreme cases (where acceleration techniques always fail). Therefore, Algorithm II is likewise a polynomial time approximation algorithm.

4. Numerical experiments

This section will set the performance of the algorithm through several test problems. All of our testing procedures were performed via MATLAB (2012a) on computers with Intel(R) Core(TM)i5-2320 3.00 GHz power processor 4.00 GB memory and Microsoft Win7 operating system.

Problem 1 [17, 18]

$$\begin{aligned}\min \quad & (0.813396x_1 + 0.67440x_2 + 0.305038x_3 + 0.129742x_4 + 0.217796) \\ & \times (0.224508x_1 + 0.063458x_2 + 0.932230x_3 + 0.528736x_4 + 0.091947)\end{aligned}$$

$$s.t. \begin{cases} 0.488509x_1 + 0.063565x_2 + 0.945686x_3 + 0.210704x_4 \leq 3.562809, \\ -0.324014x_1 - 0.501754x_2 - 0.719204x_3 + 0.099562x_4 \leq -0.052215, \\ 0.445225x_1 - 0.346896x_2 + 0.637939x_3 - 0.257623x_4 \leq 0.427920, \\ -0.202821x_1 + 0.647361x_2 + 0.920135x_3 - 0.983091x_4 \leq 0.840950, \\ -0.886420x_1 - 0.802444x_2 - 0.305441x_3 - 0.180123x_4 \leq -1.353686, \\ -0.515399x_1 - 0.424820x_2 + 0.897498x_3 + 0.187268x_4 \leq 2.137251, \\ -0.591515x_1 + 0.060581x_2 - 0.427365x_3 + 0.579388x_4 \leq -0.290987, \\ 0.423524x_1 + 0.940496x_2 - 0.437944x_3 - 0.742941x_4 \leq 0.373620, \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0. \end{cases}$$

Problem 2 [17, 18]

$$\min \quad (3x_1 - 2x_2 - 2)^{\frac{2}{3}}(x_1 + 2x_2 + 2)^{\frac{2}{5}} \; s.t. \begin{cases} 2x_1 - x_2 \geq 2, \;\; x_1 - 2x_2 \leq 2, \\ x_1 + x_2 \leq 5, \;\; 3 \leq x_1 \leq 5, 1 \leq x_2 \leq 3. \end{cases}$$

Problem 3 [20]

$$\min \quad (x_1 + x_2)(x_1 - x_2 + 7) \quad s.t. \begin{cases} 2x_1 + x_2 \leq 14, \\ x_1 + x_2 \leq 10, \\ -4x_1 + x_2 \leq 0, \\ 2x_1 + x_2 \geq 6, \\ x_1 + 2x_2 \geq 6, \\ x_1 - x_2 \leq 3, \\ 1.99 \leq x_1 \leq 2.01, \\ 7.99 \leq x_2 \leq 8.01. \end{cases}$$

Table 1: Comparison of results in Problem 1-3.

Problem	Reference	Solution	Optimum	Iter	Time	ϵ
1	[17]	(1.3148,0.1396,0.0000,0.4233)	0.890190	404	9.606	0.05
	[18]	(1.3148,0.1396,0.0000,0.4233)	0.890190	3	0.047	0.05
	Algorithm I/II	(1.3148,0.1396,0.0000,0.4233)	0.890190	1	0.0149	0.05
2	[17]	(3.000,2.000)	5.014514	69	2.4960	0.15009
	[18]	(3.000,2.000)	5.009309	4	0.0800	0.15009
	Algorithm I	(3.000,2.000)	5.014514	6	0.1024	0.15009
	Algorithm II	(3.000,2.000)	5.014514	4	0.0657	0.15009
3	[20]	(2,8)	10	2	0.015	0.01
	Algorithm I/II	(2,8)	10	1	0.0241	0.01

Problem 4

$$\min \ \prod_{i=1}^{p}(c_i^T x_j + d_i)^{\alpha_i},\ s.t.\ Ax \leq b,\ x \geq 0,$$

where $p \geq 2$, $c_i \in \mathbb{R}^n$, $\alpha_i \in \mathbb{R}$ are pseudo-random numbers in [0,1], $d_i = 1$, constraint matrix elements a_{ij} are generated in [-1,1] via $a_{ij} = 2 * Q - 1$, where Q are pseudo-random numbers in [0,1], and the right-hand side values are generated via $b_i = \sum_{j=1}^{n} a_{ij} + 2\pi$, where π are pseudo-random numbers in [0,1].

Table 2: Comparison of numerical results by using Problem 4

(p, m, n)	Algorithm I		Algorithm II	
	Avg(Std).Time	Avg(Std).Iter	Avg(Std).Time	Avg(Std).Iter
(2,10,20)	2.9068(2.8062)	75.8(84.7700)	1.9686(1.9352)	22.5(27.3395)
(2,20,20)	2.3784(3.1017)	52.6(76.8936)	1.7129(2.1472)	23.4(35.1801)
(2,22,20)	0.8663(0.9232)	18.1(25.0257)	0.6568(0.6239)	8(10.0199)
(2,20,30)	6.2414(6.3274)	165.2(164.0334)	3.4923(3.7124)	49(61.764)
(2,35,50)	3.9868(4.4041)	66.4(78.2102)	3.3046(3.9017)	32.3(38.6886)
(2,45,60)	5.8908(5.4016)	129.1(125.2481)	3.7409(3.3526)	40.5(38.1084)
(2,45,100)	6.6579(5.9685)	125.3(123.7061)	4.2665(3.6485)	40.2(40.1343)
(2,60,100)	7.8626(6.3057)	96.6(99.4818)	4.5517(2.8324)	26(19.8343)
(2,70,100)	9.1245(8.1057)	96.3(104.6633)	5.0942(3.3528)	23.6(18.9430)
(2,70,120)	11.2742(13.2311)	148(215.2185)	6.0341(5.5968)	35(37.3256)
(2,100,10)	0.1877(0.1300)	2.4(2.9732)	0.1542(0.0663)	1.3(0.6403)
(2,100,50)	0.9029(0.5392)	8.9(7.0632)	0.6542(0.3654)	3.9(2.7730)
(2,100,100)	9.8811(8.0793)	68.6(55.5287)	6.9462(6.3403)	24.1(26.6813)
(2,100,150)	15.4331(10.2573)	97.4(75.1720)	9.8838(6.2545)	30.8(22.1260)
(2,100,200)	27.1157(25.3267)	124.4(130.8076)	18.9561(16.8612)	49.2(47.3810)
(2,100,250)	64.8144(72.0125)	285.1(353.7955)	40.3711(41.0487)	91(103.9576)
(2,100,300)	87.5572(100.4846)	331(398.8197)	55.5067(64.5147)	117.2(153.2434)
(2,100,400)	132.4251(176.2381)	363.9(581.9823)	87.0321(97.4482)	130.7(169.6585)
(2,100,500)	158.4767(183.7060)	338.3(493.9785)	111.0958(106.7086)	133.8(145.3470)
(2,100,700)	331.3275(351.8534)	414.2(546.8741)	272.7311(264.9189)	227.1(257.8927)
(2,100,1000)	1020.9318(880.7910)	1063.6(1019.7921)	778.8913(638.1782)	522 (460.2479)
(3,100,10)	4.4724(7.4341)	59.7(117.2502)	3.6522(5.7934)	35.2(55.7867)
(3,100,50)	90.8139(74.9843)	1062.4(982.8277)	57.8342(55.5978)	473.3(564.6533)
(4,100,10)	75.4301(189.1250)	1509.3(4122.7180)	52.9122(122.0553)	868.1(2203.3283)

The numerical results in Table 1 show that algorithms I and II can effectively solve the three test problems known in the literature and get an approximate solution, so both algorithms are feasible.

Further, we do the corresponding random numerical experiments through the problem 4, which is utilized to explore the performance of the two algorithms. We determine the convergence accuracy of the algorithm to 0.05. For each set of fixed parameters (p, m, n), we run the two algorithms 10 times for numerical comparison, and the numerical results are given in Table 2. In Table 2, Avg(Std).Time and Avg(Std).Iter

represent the average (standard deviation) of the CPU running time and the average(standard deviation) of iterations, respectively, after the algorithm has run 10 times. Table 2 shows that the computation effect of algorithm II is better than that of algorithm I, mainly because our acceleration technique plays a significant role by deleting the vertices of small rectangles that do not need to consider. Hence, we believe that this acceleration technique may be generalized on other approximation algorithms such as [17, 18, 20].

Moreover, under the condition that the fixed parameters (p, m) are invariant, the CPU running time of the two algorithms will increase with the scale n of the problem 4. Under the condition that the pre-fixed parameters (m, n) are invariant, the CPU running time and iterations of the two algorithms will grow with the number p of linear functions in the objective function of the problem 4.

5. Conclusions

In this paper, we mainly propose two polynomial time approximation algorithms that can be utilized to solve the problem (GLMP) globally, where algorithm II is obtained by accelerating algorithm I by the proposed acceleration technique. The numerical results show that both algorithms are effective and feasible, but the overall calculation effect of algorithm II is better than that of algorithm I, which shows that our acceleration technique is efficient and can be extended to some approximate algorithms such as [17, 18, 20].

Acknowledgements

This research is supported by the National Natural Science Foundation of China under Grant (11961001), the Construction Project of first-class subjects in Ningxia higher Education(NXYLXK2017B09) and the major proprietary funded project of North Minzu University (ZDZX201901).

Author contributions

Bo Zhang and YueLin Gao conceived of and designed the study. Bo Zhang performed the experiments. Bo Zhang wrote the paper. YueLin Gao, XiaoLi Huang and Xia Liu reviewed and edited the manuscript. All authors read and approved the manuscript.

Financial disclosure

None reported.

Conflict of interest

The authors declare no potential conflict of interests.

References

[1] Matsui T. NP-Hardness of linear multiplicative programming and related problems. Journal of Global Optimization 1996, 9(2): 113-119.

[2] Maranas C, Androulakis I, Floudas C, et al. Solving long-term financial planning problems via global optimization. Journal of Economic Dynamics and Control, 1997, 21(8-9): 1405-1425.

[3] Konno H. Kuno T. Yajima Y. Global optimization of a generalized convex multiplicative function. Journal of Global Optimization, 1994, 41(1): 47-62.

[4] Mulvey J, Vanderbei R, Zenios S. Robust Optimization of Large-Scale Systems. Operations Research, 1995, 43(2): 264-281.

[5] Nicholas R, Layard P, Walters A. Microeconomic Theory. Economica, 1980, 47(186): 211.

[6] Benson H. Vector maximization with two objective functions. Journal of Optimization Theory and Applications, 1979, 28(2): 253-257.

[7] Dennis D. Analyzing Public Inputs to Multiple Objective Decisions on National Forests Using Conjoint Analysis. Forest ence, 1998, 44(3): 421-429.

[8] Liu S, Zhao Y. An efficient algorithm for globally solving generalized linear multiplicative programming. Journal of Computational and Applied Mathematics, 2016, 296: 840-847.

[9] Liu X, Umegaki T, Yamamoto Y. Heuristic methods for linear multiplicative programming. Journal of Global Optimization, 1999, 15(4): 433-447.

[10] Benson H, Boger G. Multiplicative programming problems: analysis and efficient point search heuristic. Journal of Optimization Theory and Applications, 1997, 94(2): 487-510.

[11] Shen P, Bai X, Li W. A new accelerating method for globally solving a class of nonconvex programming problems. Nonlinear Analysis, 2009, 71: 2866-2876.

[12] Chen Y, Jiao H. A nonisolated optimal solution of general linear multiplicative programming problems. Computers & Operations Research, 2009, 36(9): 2573-2579.

[13] Wang C, Bai Y, Shen P. A practicable branch-and-bound algorithm for globally solving linear multiplicative programming. Mathematische Operationsforschung Und Statistik, 2017, 66(3): 397-405.

[14] Gao Y, Xu C, Yang Y. An outcome-space finite algorithm for solving linear multiplicative programming. Applied Mathematics and Computation, 2006, 179(2): 494-505.

[15] Konno H, Kuno T. Linear multiplicative programming. Mathematical Programming, 1992, 56(1-3): 51-64.

[16] Depetrini D, Locatelli M. A FPTAS for a class of linear multiplicative problems. Computational Optimization and Applications, 2007, 44(2): 275-288.

[17] Locatelli M. Approximation algorithm for a class of global optimization problems. journal of global optimization, 2013, 55(1): 13-25.

[18] Shen P, Wang L. A Fully Polynomial Time Approximation Algorithm for Generalized Linear Multiplicative Programming. Mathematica Applicata, 2018, 31(1): 208-213.

[19] Zhang B, Gao Y, Liu X, Huang X. Output-Space Branch-and-Bound Reduction Algorithm for a Class of Linear Multiplicative Programs. Mathematics, 2020, 8: 315.

[20] Shen P , Huang B , Wang L . Range division and linearization algorithm for a class of linear ratios optimization problems. Journal of Computational and Applied Mathematics, 2019, 350: 324-342.

[21] Wang C, Liu S. A new linearization method for generalized linear multiplicative programming. Computers and Operations Research, 2011, 38(7): 1008-1013.

[22] Peiping S, Xiaoke Z, A polynomial time approximation algorithm for linear fractionl programs. Mathematica Applicata. 2011, 26(2): 355-359.