Journal of Inequalities and Applications
a SpringerOpen Journal

RESEARCH Open Access

Inner approximation algorithm for generalized linear multiplicative programming problems

Yingfeng Zhao[1*] and Juanjuan Yang[1]

*Correspondence:
zhaoyingfeng6886@163.com
[1]School of Mathematical Science, Henan Institute of Science and Technology, Xinxiang, China

Abstract

An efficient inner approximation algorithm is presented for solving the generalized linear multiplicative programming problem with generalized linear multiplicative constraints. The problem is firstly converted into an equivalent generalized geometric programming problem, then some magnifying-shrinking skills and approximation strategies are used to convert the equivalent generalized geometric programming problem into a series of posynomial geometric programming problems that can be solved globally. Finally, we prove the convergence property and some practical application examples in optimal design domain, and arithmetic examples taken from recent literatures and GLOBALLib are carried out to validate the performance of the proposed algorithm.

Keywords: Generalized multiplicative programming; Inner approximation algorithm; Geometric programming

1 Introduction

In this paper, we focus on the following generalized linear multiplicative programming problem:

$$(\text{GLMP}): \begin{cases} \min & \phi_0(y) = \sum_{j=1}^{p_0} c_{0j} \prod_{t=1}^{T_{0j}} (f_{0jt}(y))^{\gamma_{0jt}} \\ s.t. & \phi_i(y) = \sum_{j=1}^{p_i} c_{ij} \prod_{t=1}^{T_{ij}} (f_{ijt}(y))^{\gamma_{ijt}} \le 0, \quad i = 1,2,\ldots,M, \\ & y \in Y^0 = \{0 < \underline{y}_i^0 \le y_i \le \overline{y}_i^0, i = 1,2,\ldots,N\}, \end{cases}$$

where c_{ij}, γ_{ijt}, $i = 0,1,\ldots,M$, $j = 1,2,\ldots,p_i$, $t = 1,2,\ldots,T_{ij}$ are all arbitrary real numbers; p_i, T_{ij}, $i = 0,1,\ldots,M$, $j = 1,2,\ldots,p_i$ are all positive integers and $f_{ijt}(y)$, $i = 0,1,\ldots,M$, $j = 1,2,\ldots,p_i$, $t = 1,2,\ldots,T_{ij}$ are all affine functions defined on R^N such that $f_{ijt}(y) > 0$ for all $y \in Y^0$. Furthermore, we suppose that the interior of the feasible region for (GLMP) is not empty. Problem (GLMP) and its special cases are ubiquitous in optimal design applications, including power control, optimal doping profile, production planning, chemical equilibrium, heat exchanger network, digital circuit gate sizing, VLSI chip design, truss design, and so on [1–8]. And on the other hand, problem (GLMP) which corresponds to a nonlinear optimization problem with generalized linear multiplicative objective and constraint functions includes a large class of mathematical programs such as generalized geometric programming, multiplicative programming, sum of linear ratios problems, quadratic programming et al. [9–12]. Thus in this context, an algorithmic study of problem (GLMP) makes some theoretical and practical significance.

Algorithms for solving the special form of problem (GLMP) emerged endlessly. They are mainly classified as primal-based algorithms that directly solve the primal problem, dual-based algorithms that solve the dual problem, and adapted general nonlinear programming methods [13–15]. Recently, many works aimed at globally solving special forms of (GLMP) are presented, for example, global algorithms for signomial geometric programming problems, branch and bound algorithms for multiplicative programming with linear constraints, branch and reduction methods for quadratic programming problems, and sum of ratios problems are all in this category [16–21]. Despite these various contributions to their special forms, however, optimization algorithms for solving the general case of (GLMP) are still scarce. As far as we know, only [9] consider this general case, but only for (GLMP) with geometric constraints.

In this paper, we present an inner approximation algorithm for solving generalized linear multiplicative programming problem described as (GLMP). The (GLMP) is first converted into a generalized geometric programming problem, then the inner approximation algorithm relying on arithmetic-geometric mean inequality and magnifying-shrinking techniques is established. The algorithm works by solving a series of posynomial geometric programming problems. This strategy can be realized owing to the fact that recently developed solution methods can solve even large-scale posynomial geometric programming problems extremely efficiently and reliably [22]. The convergence property is proved and some examples taken from practical applications and recent literatures are performed to verify the efficiency of the presented algorithm. The experimental results show that the presented algorithm has a better capability to solve the (GLMP).

The remainder of this paper is organized in the following way. In Sect. 2, the equivalent generalized geometric programming problem is established and the inner approximation algorithm for solving (GLMP) is designed by utilizing arithmetic-geometric mean inequality and condensation techniques. The convergence property and error analysis of the algorithm are discussed in Sect. 3. Section 4 computationally investigates the performance of the inner approximation algorithm by solving some selective test examples. Some concluding remarks are proposed in the last section.

2 Equivalent problem and algorithm development

In this section, the original problem (GLMP) is first transformed into an equivalent generalized geometric programming problem (EGGP) through variable substitution. And for convenience, problem (EGGP) will be further converted into generalized geometric programming with standard form described as formulation (Q). Then our focus will be shifted to solving the equivalent problem (Q). By utilizing the arithmetic-geometric mean inequality and condense techniques based on first order Taylor expansion, we can construct a posynomial geometric programming auxiliary problem (AQ) of the reformulated problem (Q) at each iterative point. Based on this, the proposed algorithm will be developed. The proposed algorithm works by solving a sequence of posynomial geometric programming problems.

2.1 Equivalent problem

To solve the problem, we will first transform the (GLMP) into an equivalent problem (EGGP), where the objective and constraint functions are all generalized polynomial functions. To explain how such a reformulation is possible, we first compute $\underline{z}_{ijt} = \min_{y\in Y^0} f_{ijt}(y)$, $\overline{z}_{ijt} = \max_{y\in Y^0} f_{ijt}(y)$, then introduce some auxiliary variables z_{ijt} such that $0 < \underline{z}_{ijt} \leq z_{ijt} \leq \overline{z}_{ijt}$ for each $i = 0, 1, \dots, M$, $j = 1, 2, \dots, p_i$, $t = 1, 2, \dots, T_{ij}$, and define vector z and an initial box Z^0 as follows:

$$\begin{aligned} z = \{ & z_{011}, z_{012}, \dots, z_{01T_{01}}, z_{021}, z_{022}, \dots, z_{02T_{02}}, \dots, z_{0p_01}, z_{0p_02}, \dots, z_{0p_0T_{0p_0}}, \\ & z_{111}, z_{112}, \dots, z_{11T_{11}}, z_{121}, z_{122}, \dots, z_{12T_{12}}, \dots, z_{1p_11}, z_{1p_12}, \dots, z_{1p_1T_{1p_1}}, \dots, \\ & z_{M11}, z_{M12}, \dots, z_{M1T_{M1}}, z_{M21}, z_{M22}, \dots, z_{M2T_{M2}}, \dots, z_{Mp_M1}, z_{Mp_M2}, \dots, \\ & z_{Mp_MT_{Mp_M}} \} \in R^s, \\ Z^0 = & \left\{ z \in R^s \mid 0 < \underline{z}_{ijt} \leq z_{ijt} \leq \overline{z}_{ijt}, i = 0, 1, \dots, M, j = 1, 2, \dots, p_i, t = 1, 2, \dots, T_{ij} \right\}, \end{aligned}$$

where $s = \sum_{i=0}^{m} \sum_{j=1}^{p_i} T_{ij}$.

For convenience in exposition, we reintroduce some new notations as follows:

$$\begin{aligned} T_i^+ &= \left\{ (j,t) \mid c_{ijt}\gamma_{ijt} > 0, j = 1, 2, \dots, p_i, t = 1, 2, \dots, T_{ij} \right\}, \quad i = 0, 1, 2, \dots, M, \\ T_i^- &= \left\{ (j,t) \mid c_{ijt}\gamma_{ijt} < 0, j = 1, 2, \dots, p_i, t = 1, 2, \dots, T_{ij} \right\}, \quad i = 0, 1, 2, \dots, M. \end{aligned}$$

With these new notations, problem (GLMP) can be further equivalently reformulated as the following problem:

$$(\mathrm{EP}): \begin{cases} \min & \sum_{j=1}^{p_0} c_{0j} \prod_{t=1}^{T_{0j}} (z_{0jt})^{\gamma_{0jt}} \\ s.t. & f_{ijt}(y) - z_{ijt} \leq 0, \quad (j,t) \in T_i^+, i = 0, 1, 2, \dots, M, \\ & z_{ijt} - f_{ijt}(y) \leq 0, \quad (j,t) \in T_i^-, i = 0, 1, 2, \dots, M, \\ & \sum_{j=1}^{p_i} c_{ij} \prod_{t=1}^{T_{ij}} (y_{ijt})^{\gamma_{ijt}} \leq 0, \quad i = 1, 2, \dots, M, \\ & y \in Y^0, \qquad z \in Z^0. \end{cases}$$

Upon the monotonicity of the function in problem (EP), it is not too hard to find that problems (GLMP) and (EP) have the same optimal solutions in the sense of the following theorem.

Theorem 1 *y^* is an optimal solution for the* (GLMP) *if and only if* (y^*, z^*) *is an optimal solution of* (EP), *where* $z^*_{ijt} = f_{ijt}(y^*)$, $i = 0, 1, \dots, M$, $j = 1, 2, \dots, p_i$, $t = 1, 2, \dots, T_{ip_j}$.

Proof This theorem is quite easy to verify from the constructing process of problem (EP), thus the proof is omitted here. □

For convenience and without loss of generality, we can reformulate problem (EP) as the following generalized geometric programming problem (EGGP) by performing notation substitution.

$$
(\text{EGGP}): \quad \begin{cases} \min & \psi_0(x) \\ s.t. & \psi_i(x) \le 0, \quad i = 1,2,\dots,m, \\ & x \in X^0, \end{cases}
$$

where $x = (y,z) \in Y^0 \times Z^0 = X^0 \subseteq R^n$, $n = N + s$, $m = M + s$, all of functions $\psi_i(x)$ have the generalized polynomial form, that is to say, it can be described as $\psi_i(x) = \sum_{t=1}^{r_i} \delta_{it} \prod_{j=1}^{n} (x_j)^{\theta_{itj}}$, and thus we only consider how to solve problem (EGGP) from now on.

2.2 Implementable algorithm

In this part, we concentrate on how to design the inner approximation algorithm for solving the (EGGP). For this, we will perform some transformation and condensation strategies so that problem (EGGP) can be converted into a series of posynomial geometric programming problems which can be easily solved by using computer tools (such as CVX, GPLab).

To this end, we first denote all generalized polynomial functions in (EGGP) as

$$
\psi_i(x) = \psi_i^+(x) - \psi_i^-(x) \triangleq \sum_{j \in J_i^+} \delta_{ij} \prod_{j=1}^{n} (x_j)^{\theta_{ijt}} - \sum_{j \in J_i^-} \delta_{ij} \prod_{j=1}^{n} (x_j)^{\theta_{ijt}},
$$

where

$$
J_i^+ = \{j = 1,2,\dots,r_i \mid \delta_{ij} > 0\}, \qquad J_i^- = \{j = 1,2,\dots,r_i \mid \delta_{ij} < 0\}, \quad i = 0,1,2,\dots,m.
$$

Note that the objective function can be rewritten as

$$
\psi_0(x) = \frac{\sum_{t=1}^{r_0} \delta_{0t} \prod_{j=1}^{n} (x_j)^{\theta_{0tj}}}{\prod_{j=1}^{n} (x_j)^{-\eta_{0j}}} = \frac{\sum_{t \in T_0^+} \delta_{0t} \prod_{j=1}^{n} (x_j)^{\theta_{0tj}}}{\prod_{j=1}^{n} (x_j)^{-\eta_{0j}}} + \frac{\sum_{t \in T_0^-} \delta_{0t} \prod_{j=1}^{n} (x_j)^{\theta_{0tj}}}{\prod_{j=1}^{n} (x_j)^{-\eta_{0j}}},
$$

where $\eta_{0j} = \min\{0, \theta_{0tj} \mid t = 1,2,\dots,r_0\}$, $j = 1,2,\dots,n$. If we denote

$$
\psi_0^l = \frac{\sum_{t \in T_0^+} \delta_{0t} \prod_{j=1}^{n} (\underline{x}_j^0)^{\theta_{0tj}}}{\prod_{j=1}^{n} (\overline{x}_j^0)^{-\eta_{0j}}} + \frac{\sum_{t \in T_0^-} \delta_{0t} \prod_{j=1}^{n} (\overline{x}_j^0)^{\theta_{0tj}}}{\prod_{j=1}^{n} (\underline{x}_j^0)^{-\eta_{0j}}}
$$

and

$$
\psi_0^u = \frac{\sum_{t \in T_0^+} \delta_{0t} \prod_{j=1}^{n} (\overline{x}_j^0)^{\theta_{0tj}}}{\prod_{j=1}^{n} (\underline{x}_j^0)^{-\eta_{0j}}} + \frac{\sum_{t \in T_0^-} \delta_{0t} \prod_{j=1}^{n} (\underline{x}_j^0)^{\theta_{0tj}}}{\prod_{j=1}^{n} (\overline{x}_j^0)^{-\eta_{0j}}},
$$

then we have

$$
\psi_0^l \le \psi_0(x) \le \psi_0^u.
$$

This will imply from ψ_0^l that there exists a constant $\tau = -\psi_0^l + \epsilon$ with sufficiently small value $\epsilon > 0$ such that $\psi_0(x) + \tau > 0$, $\forall x \in X^0$. The reason for constructing the constant τ

is that it will force the succedaneous objective function $\psi_0(x)+\tau>0$, and it is convenient for reformulating the following equivalent optimization problem:

$$(\mathrm{Q}):\begin{cases}\min & x_0\\ s.t. & \frac{\psi_0^+(x)+\tau}{\psi_0^-(x)+x_0}\leq 1,\\ & \frac{\psi_i^+(x)}{\psi_i^-(x)}\leq 1,\quad i=1,2,\ldots,m,\\ & x\in X^0,\qquad x_0\in[\psi_0^l,\psi_0^u].\end{cases}$$

In this representation, the objective function of problem (Q) is a positive linear function, and the constraints involve a special structure in the form of a ratio between two posynomials. Given that constraints in the form of a ratio between posynomials are not allowable in standard geometric programming [22], we attempt to approximate every posynomial denominator in constraints with monomial functions. This can be realized by utilizing the following arithmetic-geometric mean inequality:

$$\Phi(x)=\sum_{i=1}^{l}v_i(x)\geq\hat{\Phi}(x)=\prod_{i=1}^{l}\left(\frac{v_i(x)}{\lambda_i(y)}\right)^{\lambda_i(y)},$$

where $v_i(x)$, $i=1,2,\ldots,l$, are monomial terms, and the parameter $\lambda_i(y)$ is obtained by computing $\lambda_i(y)=\frac{v_i(y)}{\Phi(y)}$ so that $\hat{\Phi}(x)$ is a best local monomial approximation of $\Phi(x)$ near each fixed point y [22]. Based on this, the unallowable constraints of posynomial ratios form $\frac{\Psi(x)}{\Phi(x)}\leq 1$ can be approximated with $\frac{\Psi(x)}{\hat{\Phi}(x)}\leq 1$. Applying this skill into all inapposite constraints of problem (Q), we can obtain the following auxiliary problem (AQ) which can be efficiently solved globally [22]:

$$(\mathrm{AQ}):\begin{cases}\min & x_0\\ s.t. & \frac{\psi_0^+(x)+\tau}{\tilde{\psi}_0(x,x_0)}\leq 1,\\ & \frac{\psi_i^+(x)}{\tilde{\psi}_i(x)}\leq 1,\quad i=1,2,\ldots,m,\\ & x\in X^0,\qquad x_0\in[\psi_0^l,\psi_0^u],\end{cases}$$

where $\tilde{\psi}_0(x,x_0)$ equals $\psi_0^-(x)+x_0$ if $\psi_0^-(x)+x_0$ is monomial, and $\tilde{\psi}_0(x,x_0)$ is the monomial approximation of $\psi_0^-(x)+x_0$ if $\psi_0^-(x)+x_0$ is posynomial; $\tilde{\psi}_i(x)$ equals $\psi_i^-(x)$ if $\psi_i^-(x)$ is monomial, and $\tilde{\psi}_i(x)$ is the monomial approximation of $\psi_i^-(x)$ if $\psi_i^-(x)$ is posynomial.

Based on the discussion above, now we can summarize the proposed algorithm for solving the (GLMP) as follows:

Step 1. (Initialization) Reformulate the initial problem as the equivalent form described in problem (Q), then choose a feasible point $x^{(0)}$ and $x_0^{(0)}$ (if necessary) as the starting point, give out the solution accuracy $\vartheta\geq 0$, and set iteration counter $k:=0$.

Step 2. (Inner approximation) At the k_{th} iteration, replace each constraint with its inner approximation by computing the value of $\lambda_i(y)$ at $(x_0^{(k-1)},x^{(k-1)})$, if necessary.

Step 3. (Posynomial condensation) Construct the auxiliary problem (AQ) and solve it to obtain $(x_0^{(k)},x^{(k)})$.

Step 4. (Termination) If $\|x_0^k-x_0^{k-1}\|\leq\vartheta$, then the algorithm can be terminated. Otherwise, set $k:=k+1$ and return to Step 2.

Remark 1 When performing the algorithm described above, one should choose a feasible interior point as the starting point. However, in the practical implementation, we often select an arbitrary point as the starting point when it is difficult to find a feasible interior point for some large-scale (GLMP) problems. This is mainly because the tool (GGPLab) we used for solving (AQ) can quickly produce a feasible interior point of problem (Q) [22].

3 Convergence property analysis

In this section, we will briefly take into account the convergence properties of the above algorithm and evaluate the errors in objective and constraint functions produced by monomial approximation.

Theorem 2 *The proposed algorithm either terminates within finite iterations with an KKT point for problem* (GLMP) *to be found, or the limit of any convergent sequence is a KKT point of the* (GLMP).

Proof First, according to the construction process of monomial approximation, we can easily verify that

$$\frac{\psi_0^+(x)+\tau}{\psi_0^-(x)+x_0} \le \frac{\psi_0^+(x)+\tau}{\tilde{\psi}_0(x,x_0)}, \qquad \frac{\psi_i^+(x)}{\psi_i^-(x)} \le \frac{\psi_i^+(x)}{\tilde{\psi}_i(x)}, \quad i=1,2,\dots,m, \tag{1}$$

and

$$\frac{\psi_0^+(x^k)+\tau}{\psi_0^-(x^k)+x_0^k} = \frac{\psi_0^+(x^k)+\tau}{\tilde{\psi}_0(x^k,x_0^k)}, \qquad \frac{\psi_i^+(x^k)}{\psi_i^-(x^k)} = \frac{\psi_i^+(x^k)}{\tilde{\psi}_i(x^k)}, \quad i=1,2,\dots,m. \tag{2}$$

Second, we can also prove that

$$\begin{aligned} &\triangledown\left(\frac{\psi_0^+(x^k)+\tau}{\psi_0^-(x^k)+x_0^k}\right) = \triangledown\left(\frac{\psi_0^+(x^k)+\tau}{\tilde{\psi}_0(x^k,x_0^k)}\right), \\ &\triangledown\left(\frac{\psi_i^+(x^k)}{\psi_i^-(x^k)}\right) = \triangledown\left(\frac{\psi_i^+(x^k)}{\tilde{\psi}_i(x^k)}\right), \quad i=1,2,\dots,m. \end{aligned} \tag{3}$$

Finally, we know the interior of the feasible region is not empty and all constraints in problem (AQ) are geometric-convex. This will suggest that the feasible region of problem (AQ) satisfies Slater's constraint qualification condition. Thus based on (1)–(3) and according to Theorem 1 in [23], we conclude that the sequent solutions of problem (AQ) converge to the KKT point for problem (Q), thus for problem (GLMP). □

Remark 2 Although the above algorithm can only obtain a KKT point for problem (Q), according to the special structure of the objective function of problem (Q) and the distinctive characteristics described in [23], we find that the KKT point found by the proposed algorithm is always a global optimal solution for problem (Q).

Remark 3 Suppose (x^*, x_0^*) is the final solution obtained by the presented algorithm, we can evaluate the errors in objective and constraint functions produced by monomial approximation by the following formulas:

$$\begin{aligned}\Theta_0 &= \left|\left(\psi_0^+(x^*) + \tau - \psi_0^-(x^*) - x_0^*\right) - \left(\psi_0^+(x^*) + \tau - \tilde{\psi}_0(x^*, x_0^*)\right)\right| \\ &= \left|\psi_0^-(x^*) + x_0^* - \tilde{\psi}_0(x^*, x_0^*)\right|, \\ \Theta_i &= \left|\left(\psi_i^+(x^*) - \psi_i^-(x^*)\right) - \left(\psi_i^+(x^*) - \tilde{\psi}_i(x^*)\right)\right| \\ &= \left|\psi_i^-(x^*) - \tilde{\psi}_i(x^*)\right|, \quad i = 1, 2, \ldots, m.\end{aligned}$$

4 Computational experiments

To test the proposed algorithm in terms of efficiency and solution quality, we performed some computational examples on a personal computer with Intel Xeon(R) CPU 2.40 Ghz and 4 GB memory. The code base is written in matlab 2014a and interfaces GGPLab for the standard geometric programming problems.

We consider some instances of problem (MIQQP) from some recent literature [9, 24–27] and MINLPLib [28]. Among them, Examples 1, 3, and 4 are three practical applications of (GLMP). Examples 2, 5, 6, 7, 8, and 9 are taken from recent literature for comparison analysis. Example 10 is an example for testing the influence of the numerical experiments for different initial points. Examples 11–13 are three examples from GLOBALLib [29], a collection of nonlinear programming models. The last example is a generalized linear multiplicative programming problem with randomized objective and constraint functions.

Example 1 (see [24])

$$\begin{cases} \min & x_1 + x_2 + x_3 \\ \text{s.t.} & 833.33252 x_1^{-1} x_4 x_6^{-1} + 100 x_6^{-1} \le 1, \\ & 1250 x_2^{-1} x_5 x_7^{-1} + x_4 x_7^{-1} - 1250 x_2^{-1} x_4 x_7^{-1} \le 1, \\ & 1{,}250{,}000 x_3^{-1} x_8^{-1} + x_5 x_8^{-1} - 2500 x_3^{-1} x_5 x_8^{-1} \le 1, \\ & 0.0025 x_4 + 0.0025 x_6 \le 1, \\ & -0.0025 x_4 + 0.0025 x_5 + 0.0025 x_7 \le 1, \\ & 0.01 x_8 - 0.01 x_5 \le 1, \\ & 100 \le x_1 \le 10{,}000, \\ & 1000 \le x_2, x_3 \le 10{,}000, \\ & 10 \le x_i \le 1000, \quad i = 4, 5, \ldots, 8. \end{cases}$$

This special instance of (GLMP) is first proposed to deal with the optimal design of heat exchanger networks [30]. When performing the algorithm for solving this instance, we choose $(500, 500, 4200, 500, 400, 340, 300, 600)$ as the starting point, the termination error was set to be $\vartheta = 1 \times 10^{-6}$. The proposed algorithm terminates after 3.74 seconds (CPU time) with solution $(579.326059, 1359.9445, 5109.977472, 182.019317, 295.600901, 217.980682, 286.418416, 395.600901)$ and optimal value 6944.248031 to be found, and the number of iterations is 21. While the method of Tsai and Lin [24] takes nearly one hour and forty minutes for solving this example, and they obtain a solution $(578.973143, 1359.572730, 5110.701048, 181.9898, 295.5719, 218.0101, 286.4179, 395.5719)$ with the optimal value 7049.24682.

Example 2 (see [9])

$$\begin{cases} \min & (x_1+x_2+1)^{1.1}(x_1+x_2+2)^{-1.1}(x_1+x_2+3)^{1.2}(x_1+x_2+4)^{-1.2} \\ & -(x_1+x_2+6)^{1.1}(x_1+x_2+5)^{-1.1}(x_1+x_2+8)^{1.2}(x_1+x_2+7)^{-1.2} \\ \text{s.t.} & x_1^{-1}x_2^{0.5}+x_1x_2 \le 4, \\ & 1 \le x_1 \le 2, \qquad 1 \le x_2 \le 2. \end{cases}$$

In this example, both the objective function and the constraint function are generalized linear multiplicative functions. This example is taken from Jiao, Liu, and Zhao [9]. For solving this problem with the branch and bound algorithm, quite a lot of CPU times need to be consumed; however, we only expend less than two seconds for solving it to global optimality. In the iteration process, we select $(1.5, 1.5)$ as the starting point, the termination error was also set to be $\vartheta = 1 \times 10^{-6}$.

Example 3 (see [25])

$$\begin{cases} \min & 0.5(x_1-10)x_2^{-1}-x_1 \\ \text{s.t.} & x_2x_3^{-1}+x_1+0.5x_1x_3 \le 100, \\ & 1 \le x_i \le 100, \quad i=1,2,3. \end{cases}$$

This example is a signomial geometric programming problem (special case of (GLMP)) which is used to optimize the design of a membrane separation process [25]. Lin and Tsai solved it with a range reduction method and obtained an optimal solution with optimal value −83.249728. For obtaining this solution, the range reduction method spend about 22 second (CPU time). Here, our algorithm terminated after 11 iterations and obtained the optimal solution $(87.614446, 8.754375, 1.413643, 19.311410)$ with optimal value −85.68859, the algorithm implementation took about 0.942 seconds. In the algorithm implementation, we choose the initial upper bound $(100, 100, 100)$ as the starting point, the termination error was set to be $\vartheta = 1 \times 10^{-6}$.

Example 4 (see [24])

$$\begin{cases} \min & -x_1+0.4x_1^{0.67}x_7^{-0.67}-x_2+0.4x_2^{0.67}x_8^{-0.67}+10 \\ \text{s.t.} & 0.0588x_5x_7+0.1x_1 \le 1, \\ & 4x_3x_5^{-1}+2x_3^{-0.71}x_5^{-1}+0.0588x_3^{-1.3}x_7 \le 1, \\ & 0.0558x_6x_8+0.1x_1+0.1x_2 \le 1, \\ & 4x_4x_6^{-1}+2x_4^{-0.71}x_6^{-1}+0.0588x_4^{-1.3}x_8 \le 1, \\ & 0.1 \le x_i \le 10, \quad i=1,2,\ldots,8. \end{cases}$$

This example is a mathematical model born from optimal design of a reactor. For solving it, we select $(7, 7, 7, 7, 7, 7, 7, 7)$ as the starting point, the termination error was set to be $\vartheta = 1 \times 10^{-6}$. The proposed algorithm terminates after 7.123 seconds (CPU time) with solution $(6.350802, 2.365111, 0.670723, 0.597563, 5.951950, 5.537204, 1.042703, 0.415594)$ and optimal value 3.908619 to be found, and the number of iterations is 44. While Tsai and

Lin [24] spent nearly 56 minutes and 312 seconds for solving this example and obtained a solution (6.473164, 2.238234, 0.664955, 0.591012, 5.930263, 5.523595, 1.011611, 0.397171) with the optimal value 3.95109.

Example 5 (see [9])

$$\begin{cases} \min & 3.7x_1^{0.85} + 1.985x_1 + 700.3x_2^{-0.75} \\ \text{s.t.} & 0.7673x_2^{0.05} - 0.05x_1 \le 1, \\ & 0.1 \le x_1 \le 5, \qquad 380 \le x_2 \le 450. \end{cases}$$

Example 6 (see [9])

$$\begin{cases} \min & -x_1 + 0.4x_1^{0.67}x_3^{0.67} \\ \text{s.t.} & 0.05882x_3x_4 + 0.1x_1 \le 1, \\ & 4x_2x_4^{-1} + 2x_2^{-0.71}x_4^{-1} + 0.05882x_2^{-1.3}x_3 \le 1, \\ & 0.1 \le x_i \le 10, \quad i = 1,2,3,4. \end{cases}$$

Example 7 (see [27])

$$\begin{cases} \min & 5.3578x_3^2 + 0.8357x_1x_5 + 37.2392x_1 \\ s.t. & 0.00002584x_3x_5 - 0.00006663x_2x_5 - 0.0000734x_1x_4 \le 1, \\ & 0.000853007x_2x_5 + 0.00009395x_1x_4 - 0.00033085x_3x_5 \le 1, \\ & 1330.3294x_2^{-1}x_5^{-1} - 0.42x_1x_5^{-1} - 0.30586x_2^{-1}x_3^2x_5^{-1} \le 1, \\ & 0.00024186x_2x_5 + 0.00010159x_1x_2 + 0.00007379x_3^2 \le 1, \\ & 2275.1327x_3^{-1}x_5^{-1} - 0.2668x_1x_5^{-1} - 0.40584x_4x_5^{-1} \le 1, \\ & 0.00029955x_3x_5 + 0.00007992x_1x_3 + 0.00012157x_3x_4 \le 1, \\ & 78 \le x_1 \le 102, \qquad 33 \le x_2 \le 45, \qquad 27 \le x_i \le 45, \quad i = 3,4,5. \end{cases}$$

Example 8 (see [26])

$$\begin{cases} \min & x_1(-4x_1 + x_2 + 2) - 5x_2^2 \\ \text{s.t.} & x_1 - x_2 \ge 0, \\ & (x_1 + x_2)(x_1 - x_2) \le 3, \\ & x_1x_2 \le 2, \\ & 0 \le x_1, x_2 \le 3. \end{cases}$$

Example 9 (see [9, 27])

$$\begin{cases} \min & x_1 \\ \text{s.t.} & x_1(1 - x_1) + x_2(8 - x_2) \le 16, \\ & x_1(x_1 - 6) + x_2(x_2 - 6) \le -14, \\ & 1 \le x_1, x_2 \le 5.5. \end{cases}$$

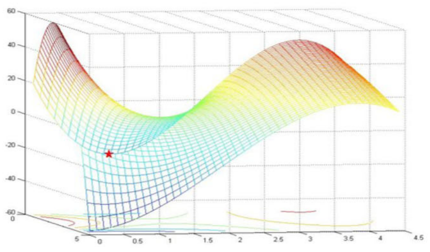
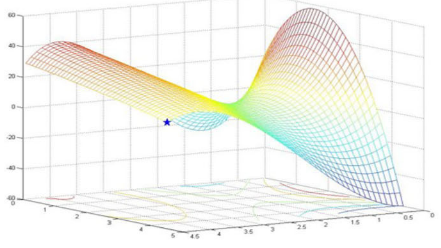
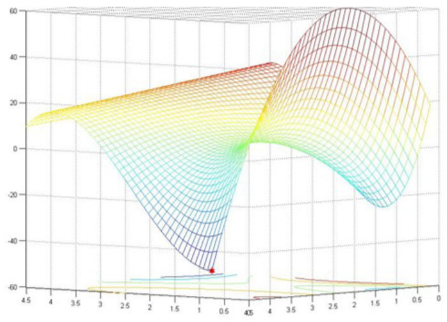



Figure 1 Two local solutions for Example 11 obtained by the proposed algorithm

Figure 2 Global optimal solution for Example 11



Example 10 (see Figs. 1–2)

$$\begin{cases} \min & (x_1-1)(x_1-2)(x_2-7)(x_1-5)-(x_2-1)(x_2-3)(x_1-4)^2 \\ s.t. & 0.1 \le x_1 \le 4.5, \qquad 0.1 \le x_2 \le 4.5. \end{cases}$$

When solving this example, by selecting $x^0 = (0.1, 0.4)$ and $y^0 = (2, 2)$ as starting points and applying the algorithm presented above, we obtained two different solutions $x_{\mathrm{opt}} = (4.5, 4.5)$ and $y_{\mathrm{opt}} = (1.175957, 0.1)$ with optimal objective values 9.625 and -24.641098, respectively. However, both of these two solutions are not the global optimal solution for Example 11. Actually, the only global optimal solution for Example 11 is $(0.1, 4.5)$ with optimal value -58.905. Thus solutions $x_{\mathrm{opt}} = (4.5, 4.5)$ and $y_{\mathrm{opt}} = (1.1759570.1)$ just are two local solutions. The distribution of these three solutions for Example 11 are drawn in Figs. 1–2.

Example 11 (st-qpk1)

$$\begin{cases} \min & 2x_1 - 2x_1^2 + 2x_1x_2 + 3x_2 - 2x_2^2 \\ \text{s.t.} & -x_1 + x_2 \le 1, \\ & x_1 - x_2 \le 1, \\ & -x_1 + 2x_2 \le 3, \\ & 2x_1 - x_2 \le 3, \\ & 0 \le x_1, x_2. \end{cases}$$

Example 12 (ex8-1-7)

$$\begin{cases} \min & (x_1-1)^2+(x_1-x_2)^2+(x_2-x_3)^3+(x_3-x_4)^4+(x_4-x_5)^4 \\ \text{s.t.} & x_2^2+x_3^3+x_1 \le 6.24264068711929, \\ & -x_3^2-x_2^2-x_1 \le -6.24264068711929, \\ & -x_3^2+x_2+x_4 \le 0.82842712474629, \\ & x_1x_5=2, \\ & -5 \le x_1,x_2,x_3,x_4,x_5 \le 5. \end{cases}$$

Example 13 (ex4-1-9)

$$\begin{cases} \min & -x_1-x_2 \\ \text{s.t.} & 8x_1^3-2x_1^4-8x_1^2+x_2 \le 2, \\ & 32x_1^3-4x_1^4-88x_1^2+96x_1+x_2 \le 36, \\ & 0 \le x_1 \le 3, \qquad 0 \le x_2 \le 4. \end{cases}$$

Example 14 (Small random test)

$$\begin{cases} \min & (c^1x+m_1)^{\alpha_1}(c^2x+m_2)^{\alpha_2}-(d^1x+r_1)^{\beta_1}(d^2x+r_2)^{\beta_2} \\ \text{s.t.} & (a^1x+s_1)^{\gamma_1}(a^2x+s_2)^{\gamma_2}-(b^1x+t_1)^{\theta_1}(b^2x+t_2)^{\theta_2} \le 10+s_1^{\gamma_1}s_2^{\gamma_2}-t_1^{\theta_1}t_2^{\theta_2}, \\ & 0 \le x \le 1, \end{cases}$$

where c^1, c^2, d^1, d^2, a^1, a^2, b^1, b^2 are n-dimensional row vectors randomly generated in $[0,1]$, m_1, m_2, r_1, r_2, s_1, s_2, t_1, t_2 are all random real numbers between 0.001 and 1.001, α_1, α_2, β_1, β_2, γ_1, γ_2, θ_1, θ_2 are real numbers randomly generated in $[0,1]$, and we choose n-dimensional vector $(0.5,0.5,\dots,0.5)$ as the starting point in each instance. The computational results of this problem are listed in Table 3.

Actually, the examples we chose in this section can be classified into four groups: Examples 1, 3, and 4 are taken from applications in optimal design; Examples 2, 5, 6, 7, 9 are numerical tests selected from some recent literature; Example 8 is computed to illustrate that the proposed algorithm can find just a local solution; and Example 14 is an example randomly generated with a relative large scale. Computational results are demonstrated in Tables 1–4 and Figs. 1–2. The computational results listed in the tables and figures show that our algorithm can perfectly solve problem (GLMP), and for most cases it can even attain a global optimal solution.

5 Concluding remarks

In this paper, an inner approximation algorithm is presented for solving the generalized linear multiplicative programming problem. Local convergence property is proved and some numerical examples taken from application domain and recent literature are performed to verify the efficiency of the algorithm and quality of the solutions obtained. Results of the numerical tests show that this algorithm can effectively solve most generalized linear multiplicative problems to global optimality although it just has local convergence property.

Table 1 Results of Examples 1–9 obtained by utilizing the presented method

Example	Start point	Iterations	Error in objective	Error constraint
1	$(500,500,4200,500,400,340,300,600)$	21	0	2.2204×10^{-16}
2	$(1.5,1.5)$	5	0	8.8818×10^{-16}
3	$(100,100,100)$	11	2.5070×10^{-16}	0
4	$(7,7,7,7,7,7,7,7)$	44	0	0
5	$(3,400)$	15	0	0
6	$(0.7,0.7,0.7,0.7)$	8	0	0
7	$(100,40,30,30,30)$	5	0	2.2204×10^{-16}
8	$(1.5,1)$	4	9.0949×10^{-13}	0
9	$(2,1.5)$	6	3.5527×10^{-15}	7.1054×10^{-15}

Table 2 Results of the numerical comparison of Examples 5–9

Example	Methods	Optimal value	Optimal solution	CPU time
5	[9]	11.9541	$(11.9604,0.8105,442.344)$	0.416
	Ours	11.3497	$(11.9604,0.681143,436.918047)$	0.13252
6	[9]	−5.7416	$(8.1244,0.6027,0.5660,5.6352)$	42.3259
	Ours	−9.2952	$(9.6867,0.5585,0.1000,5.3252)$	0.8273
7	[27]	10,127.13	$(78,32.999,29.995,45,36.7753)$	1
	Ours	10,122.49325	$(78,33,29.9957,45,36.775327)$	0.331298
8	[26]	−15.0	$(2,1)$	120.580
	Ours	−15.0	$(2,1)$	0.3556
9	[9]	1.177081	$(1.77091,2.17715)$	0.2260
	[27]	1.1771243	$(1.17712,2.17712)$	0.26069
	Ours	1.177124	$(1.177124,2.177124)$	0.18726

Table 3 Results of numerical experiments (Examples 11–13)

Example	Best solution	Our solution	Best value	Our value
11 (st-qpk1)	–	$(1,0)$	–	0
12 (ex8-1-7)	$(1.116635,1.220441,1.53779,$ $1.97277,1.7911)$	$(1.116635,1.220441,1.53779,$ $1.97277,1.7911)$	0.0293	0.0291
13 (ex4-1-9)	$(2.32952,3.1785)$	$(2.32952,3.1785)$	−5.508	−5.511

Table 4 Computational results of random Example 14

Dimension	Iterations	CPU time	Error in objective	Error in constraint
$n=5$	23	9.082938	0	4.4409×10^{-16}
$n=10$	20	14.92016	3.5527×10^{-15}	2.6645×10^{-15}
$n=20$	17	36.85216	0	6.2172×10^{-15}
$n=30$	53	239.0432	3.5527×10^{-15}	5.3291×10^{-15}
$n=50$	25	257.7263	0.7698×10^{-15}	1.5395×10^{-14}
$n=70$	35	740.6696	8.8818×10^{-16}	3.5527×10^{-15}
$n=80$	56	1583.152	1.7764×10^{-15}	1.7764×10^{-15}
$n=100$	69	2043.238	0	3.1086×10^{-15}

Acknowledgements
Thanks for all the referenced authors.

Funding
This paper is supported by the Science and Technology Key Project of Education Department of Henan Province (14A110024).

Competing interests
The authors declare that there is no conflict of interest regarding the publication of this paper.

Authors' contributions
Both authors contributed equally to the manuscript, and they read and approved the final manuscript.

Publisher's Note

Received: 7 July 2018 Accepted: 13 December 2018 Published online: 20 December 2018

References

1. Seido, A.A., Nowak, B., Chu, C.: Fitted Elmore delay: a simple and accurate interconnect delay model. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **12**(7), 691–696 (2004)
2. Abuo-El-Ata, M., Fergany, H., El-Wakeel, M.: Probabilistic multi-item inventory model with varying order cost under two restrictions: a geometric programming approach. Int. J. Prod. Econ. **83**(3), 223–231 (2003)
3. Boche, H., Stanczak, S.: Optimal QoS tradeoff and power control in CDMA systems. In: Proceedings of the 23rd IEEE Iinfocom, pp. 477–486 (2004)
4. Boyd, S., Kim, S.J., Patil, D., Horowitz, M.: Digital circuit optimization via geometric programming. Oper. Res. **53**(6), 899–932 (2005)
5. Chiang, M.: Balancing transport and physical layers in wireless multihop networks: jointly optimal congestion control and power control. IEEE J. Sel. Areas Commun. **23**(1), 104–116 (2005)
6. Dorneich, M.C., Sahinidis, N.V.: Global optimization algorithms for chip design and compaction. Eng. Optim. **25**, 131–154 (1995)
7. Ciric, A.R., Floudas, C.A.: A retrofit approach for heat exchanger networks. Comput. Chem. Eng. **13**(6), 703–715 (1989)
8. Greenberg, H.: Mathematical programming models for environmental quality control. Oper. Res. **43**(4), 578–622 (1995)
9. Jiao, H.W., Liu, S.Y., Zhao, Y.F.: Effective algorithm for solving the generalized linear multiplicative problem with generalized polynomial constraints. Appl. Math. Model. **39**, 7568–7582 (2015)
10. Zhou, X.G., Wu, K.: A method of acceleration for a class of multiplicative programming problems with exponent. J. Comput. Appl. Math. **223**, 975–982 (2009)
11. Wang, C.F., Liu, S.Y., Shen, P.P.: Global minimization of a generalized linear multiplicative programming. Appl. Math. Model. **36**, 2446–2451 (2012)
12. Shen, P.P., Li, X.A.: Branch-reduction-bound algorithm for generalized geometric programming. J. Glob. Optim. **56**, 1123–1142 (2013)
13. Ecker, J.G.: Geometric programming: methods, computations and applications. SIAM Rev. **22**(3), 338–362 (1980)
14. Kortanek, K.O., Xu, X.J., Ye, Y.Y.: An infeasible interior-point algorithm for solving primal and dual geometric programs. Math. Program. **76**, 155–181 (1996)
15. Passy, U.: Generalized weighted mean programming. SIAM J. Appl. Math. **20**, 763–778 (1971)
16. Xu, G.X.: Global optimization of signomial geometric programming problems. Eur. J. Oper. Res. **233**, 500–510 (2014)
17. Jiao, H.W., Liu, S.Y.: A practicable branch and bound algorithm for sum of linear ratios problem. Eur. J. Oper. Res. **243**, 723–730 (2015)
18. Shen, P.P., Wang, C.F.: Global optimization for sum of linear ratios problem with coefficients. Appl. Math. Comput. **176**, 219–229 (2006)
19. Wang, Y.J., Shen, P.P., Liang, Z.A.: A branch-and-bound algorithm to globally solve the sum of several linear ratios. Appl. Math. Comput. **168**, 89–101 (2005)
20. Jiao, H.W.: A branch and bound algorithm for globally solving a class of nonconvex programming problems. Nonlinear Anal. **70**, 1113–1123 (2009)
21. Phuong, N.T.H., Tuy, H.: A unified monotonic approach to generalized linear fractional programming. J. Glob. Optim. **26**, 229–259 (2003)
22. Boyd, S., Kim, S.J., Vandenberghe, L., Hassibi, A.: A tutorial on geometric programming. Optim. Eng. **8**, 67–127 (2007)
23. Marks, B.R., Wright, G.P.: A general inner approximation algorithm for nonconvex mathematical programs. Oper. Res. **26**(4), 681–683 (1978)
24. Lin, M.H., Tsai, J.F.: Range reduction techniques for improving computational efficiency in global optimization of signomial geometric programming problems. Eur. J. Oper. Res. **216**(1), 17–25 (2012)
25. Dembo, R.S., Avriel, M.: Optimal design of a membrane separation process using signomial programming. Math. Program. **15**(1), 12–25 (1978)
26. Shen, P.P., Duan, Y.P., Ma, Y.: A robust solution approach for nonconvex quadratic programs with additional multiplicative constraints. Appl. Math. Comput. **201**, 514–526 (2008)
27. Shen, P.P., Zhang, K.C.: Global optimization of signomial geometric programming using linear relaxation. Appl. Math. Comput. **150**(1), 99–114 (2004)
28. MINLP, http://www.minlplib.org
29. GLOBALLib, http://www.gamsworld.org/global/globallib.htm (2016). Vision: 334
30. Avriel, M., Williams, A.C.: An extension of geometric programming with applications in engineering optimization. J. Eng. Math. **5**(3), 187–194 (1971)