J Glob Optim (2008) 41:593–612
DOI 10.1007/s10898-008-9283-0

# A robust algorithm for generalized geometric programming

**Peiping Shen · Yuan Ma · Yongqiang Chen**

Received: 16 July 2007 / Accepted: 22 January 2008 / Published online: 22 February 2008


**Abstract** Most existing methods of global optimization for generalized geometric programming (GGP) actually compute an approximate optimal solution of a linear or convex relaxation of the original problem. However, these approaches may sometimes provide an infeasible solution, or far from the true optimum. To overcome these limitations, a robust solution algorithm is proposed for global optimization of (GGP) problem. This algorithm guarantees adequately to obtain a robust optimal solution, which is feasible and close to the actual optimal solution, and is also stable under small perturbations of the constraints.

**Keywords** Generalized geometric programming · Robust solution · Global optimization · Essential optimal solution · Monotonic optimization

## 1 Introduction

Generalized geometric programming (GGP) is a special nonlinear programming. Its great impact has been in the areas of

(1) Engineering design [1–4];
(2) Economics and statistics [5–8];
(3) Manufacturing [9,10];
(4) Chemical equilibrium [11,12].

(GGP) problem can be formulated as follows:

$$\begin{aligned} \text{(GGP)}\ \min\ & F_0(y) \\ \text{s.t.}\ & F_m(y) \geq 0, \quad m = 1, \ldots, M, \\ & y \in \Omega_0 = \{y \mid 0 < y_i^l \leq y_i \leq y_i^u < \infty,\ y_i^l \neq y_i^u,\ i = 1, \ldots, n_0\}, \end{aligned}$$

P. Shen (✉) · Y. Ma · Y. Chen
College of Mathematics and Information Science, Henan Normal University, Xinxiang 453007, P.R.China
e-mail: shenpeiping@163.com

where

$$F_m(y) = \sum_{t=1}^{T_m} \delta_{mt} \prod_{i=1}^{n_0} y_i^{\eta_{mti}}, \quad \text{for each } m = 0, 1, \ldots, M,$$

and $\delta_{mt}$ are arbitrary real constant coefficients, $\eta_{mti}$ are arbitrary real constant exponents. In general, formulation (GGP) corresponds to a nonlinear optimization problem with nonconvex objective function and constraints.

Though local optimization approaches for solving (GGP) problem are ubiquitous, the global optimization algorithm based on the characteristics of (GGP) problem is scarce. Maranas and Floudas [11] proposed such a global optimization algorithm based on the exponential variable transformation of (GGP), the convex relaxation and branching and bounding on some rectangle region. Recently, using differential linear relaxation, several authors (for example, Shen and Zhang [13], Wang and Zhang [14]) presented the corresponding branch and bound algorithms for solving (GGP) problem, respectively. However, as has been shown in Ref. [15], these methods mentioned above may sometimes provide an infeasible solution which cannot be accepted as an approximate optimal solution in any reasonable sense. This poses the necessity to re-examine the approximation concept so far commonly used and stresses the importance of robustness for practical implementation of global optimization methods. Motivated by these consideration, a robust approach to (GGP) problem will be proposed below.

The goal of this research is two-fold. First, we present a transformation of the problem based on the characteristics of (GGP) problem. Thus the original problem (GGP) is equivalently reformulated as a monotonic optimization problem (GGP3) in the form studied in recent papers [16,17]. That is to say, in the monotonic optimization problem (GGP3) the objective function is increasing and all the constrained functions can be denoted as the difference of two increasing functions. Second, by using a special procedure of monotonic optimization problem (GGP3) (see Ref. [16,17]), we propose a robust solution algorithm for (GGP) problem based on a robust approach developed in Ref. [15] but more easily implementable. This is because it requires less nonlinear computations, and main computation work is to solve the linear programming. Compared with most existing methods that are based on solving a refined linear or convex relaxation of the original problem (GGP), this robust approach consists in seeking the best nonisolated feasible solution. This solution, i.e., the robust optimal solution which is computed by the proposed approach is adequately guaranteed to be feasible and to be close to the actual optimal solution. Hence, the proposed approach can find a more appropriate approximate optimal solution which is also stable under small perturbations of the constraints. This stresses the importance of robustness for practical implementation of global optimization methods.

The remainder of this paper is organized as follows. The next section converts the (GGP) problem into a monotonic optimization problem. Section 3 introduces the concept of essential optimality (see, e.g., Ref. [15]). In addition, a method for finding such an essential optimal solution is presented in this section. The rectangular branching process, the reducing process and the upper bounding process used in this approach are defined and studied in Sect. 4. Section 5 incorporates this approach into an algorithm for solving (GGP) to be referred to a robust solution algorithm, and shows the convergence property of the algorithm. In Sect. 6, we give the results of solving some numerical examples with the algorithm.

## 2 Equivalent monotonic reformulation

A function $f: R^n \to R$ is said to be increasing if $f(x') \le f(x)$ for all $x', x \in R^n$ satisfying $x' \le x$, i.e. $x'_i \le x_i, \forall i = 1, \ldots, n$. Any function that can be decomposed into the difference of two increasing functions is said to be a d.m. function.

In the following we show that any (GGP) problem can be transformed into a monotonic optimization problem with increasing objective function and d.m. constrained functions. To see how such a reformulation is possible, we first consider each constraint of (GGP). Let

$$\eta_m = \max\{|\eta_{mti}| \mid t = 1, \ldots, T_m, i = 1, \ldots, n_0\}, \quad m = 1, \ldots, M,$$

then for any $y \in \Omega_0$, it follows from each constraint of (GGP) that

$$F_m(y) \cdot \prod_{i=1}^{n_0} y_i^{\eta_m} = \sum_{t=1}^{T_m} \delta_{mt} \prod_{i=1}^{n_0} y_i^{\eta_{mti}+\eta_m} \ge 0, \quad m = 1, \ldots, M.$$

By changing the notation, one can thus convert (GGP) into the form

$$\begin{aligned} \text{(GGP1)} \quad \min \ & G_0(y) \\ \text{s.t.} \ & G_m(y) \ge 0, \quad m = 1, \ldots, M, \\ & y \in \Omega_0 = \{y \mid 0 < y_i^l \le y_i \le y_i^u < \infty, \quad i = 1, \ldots, n_0\}, \end{aligned}$$

where

$$G_m(y) = \sum_{t=1}^{T_m} \delta_{mt} \prod_{i=1}^{n_0} y_i^{\gamma_{mti}}, \quad m = 0, 1, \ldots, M$$

with $\gamma_{0ti} = \eta_{0ti}$ and $\gamma_{mti} = \eta_{mti} + \eta_m > 0$, for $m = 1, \ldots, M$.

Moreover, by applying the following exponent transformation

$$y_i = \exp\ z_i, \quad i = 1, \ldots, n_0$$

to the formulation (GGP1), we can obtain the following equivalent programming problem:

$$\begin{aligned} \text{(GGP2)} \quad \min \ & f_0(z) \\ \text{s.t.} \ & f_m(z) \ge 0, \quad m = 1, \ldots, M, \\ & z \in \Omega = \{z \mid z_i^l = \ln y_i^l \le z_i \le \ln y_i^u = z_i^u < \infty, \quad i = 1, \ldots, n_0\}, \end{aligned}$$

where $f_m(z) = \sum_{t=1}^{T_m} \delta_{mt} \exp(\sum_{i=1}^{n_0} \gamma_{mti} z_i), \quad m = 0, 1, \ldots, M.$

Next, we turn to consider the objective function of (GGP2). For convenience, for each $m = 0, 1, \ldots, M$, we assume without loss of generality that $\delta_{mt} > 0$ for $t = 1, \ldots, J_m$, and $\delta_{mt} < 0$ for $t = J_m + 1, \ldots, T_m$. Let

$$I_t^+ = \{i \mid \gamma_{0ti} > 0, i = 1, \ldots, n_0\}, \ I_t^- = \{i \mid \gamma_{0ti} < 0, i = 1, \ldots, n_0\}.$$

In addition, some notations are introduced as follows:

$$l_t = \min_{z\in\Omega} \sum_{i\in I_t^+} \gamma_{0ti} z_i = \sum_{i\in I_t^+} \gamma_{0ti} \ln y_i^l, \quad t = J_0+1, \ldots, T_0,$$

$$u_t = \max_{z\in\Omega} \sum_{i\in I_t^+} \gamma_{0ti} z_i = \sum_{i\in I_t^+} \gamma_{0ti} \ln y_i^u, \quad t = J_0+1, \ldots, T_0,$$

$$L_t = \min_{z\in\Omega} \sum_{i\in I_t^-} \gamma_{0ti} z_i = \sum_{i\in I_t^-} \gamma_{0ti} \ln y_i^u, \quad t = 1, \ldots, J_0,$$

$$U_t = \max_{z\in\Omega} \sum_{i\in I_t^-} \gamma_{0ti} z_i = \sum_{i\in I_t^-} \gamma_{0ti} \ln y_i^l, \quad t = 1, \ldots, J_0.$$

Then, by introducing an additional vector $w = (w_1, w_2, \ldots, w_{T_0})^T \in R^{T_0}$, we can convert the problem (GGP2) into

$$\begin{aligned}
\text{(GGP3)}\ \min\ & \sum_{t=1}^{J_0} \delta_{0t} \exp\left(\sum_{i\in I_t^+} \gamma_{0ti} z_i + w_t\right) + \sum_{t=J_0+1}^{T_0} \delta_{0t} \exp\left(\sum_{i\in I_t^-} \gamma_{0ti} z_i - w_t\right),\\
\text{s.t.}\ & \sum_{t=1}^{J_m} \delta_{mt} \exp\left(\sum_{i=1}^{n_0} \gamma_{mti} z_i\right) + \sum_{t=J_m+1}^{T_m} \delta_{mt} \exp\left(\sum_{i=1}^{n_0} \gamma_{mti} z_i\right) \geq 0, \quad m = 1, \ldots, M,\\
& w_t + \sum_{i\in I_t^-} (-\gamma_{0ti}) z_i \geq 0, \quad t = 1, \ldots, J_0,\\
& \sum_{i\in I_t^+} \gamma_{0ti} z_i + w_t \geq 0, \quad t = J_0+1, \ldots, T_0,\\
& L_t \leq w_t \leq U_t, \quad t = 1, \ldots, J_0,\\
& -u_t \leq w_t \leq -l_t, \quad t = J_0+1, \ldots, T_0,\\
& z \in \Omega.
\end{aligned}$$

Note that the objective function of (GGP3) is increasing and each constrained function is a d.m. function. The key equivalence result for problems (GGP2) and (GGP3) is given by the following Theorem 1.

**Theorem 1** *If* $(z^*, w^*)$ *is a global optimal solution for problem* (GGP3)*, then* $z^*$ *is a global optimal solution for problem* (GGP2)*. Conversely, if* $z^*$ *is a global optimal solution for problem* (GGP2)*, then* $(z^*, w^*)$ *is a global optimal solution for problem* (GGP3)*, where,* $w_t^* = \sum_{i\in I_t^-} \gamma_{0ti} z_i^*$ $(t = 1, \ldots, J_0)$, $w_t^* = -\sum_{i\in I_t^+} \gamma_{0ti} z_i^*$ $(t = J_0+1, \ldots, T_0)$.

*Proof* The proof of this theorem follows easily from the definitions of problems (GGP2) and (GGP3), therefore, it is omitted. □

From Theorem 1, notice that, in order to globally solve problem (GGP2), we may globally solve problem (GGP3) instead. In addition, it is easy to see that the global optimal values of problems (GGP2) and (GGP3) are equal.

In addition, for the sake of simplicity, let $x = (z, w) \in R^{n_0+T_0}$ with $z \in R^{n_0}$, $w \in R^{T_0}$ and let $n = n_0 + T_0$, then, without loss of generality, by changing the notation, the problem (GGP3) can be rewritten as the form

$$\text{(P)}\ \min\{g(x) \mid h(x) \geq 0,\ x \in X_0 = [x^l, x^u]\},$$

where

$$X_0 = \{x \in R^n \mid x_i^l \le x_i \le x_i^u,\ i = 1, \ldots, n\}$$

$$= \left\{ x \in R^n \;\middle|\; \begin{array}{ll} z_i^l \le x_i = z_i \le z_i^u, & i = 1, \ldots, n_0, \\ L_{i-n_0} \le x_i = w_{i-n_0} \le U_{i-n_0}, & i = n_0+1, \ldots, n_0+J_0, \\ -u_{i-n_0-J_0} \le x_i = w_{i-n_0-J_0} \le -l_{i-n_0-J_0}, & \\ & i = n_0+J_0+1, \ldots, n_0+T_0 \end{array} \right\},$$

and $g(x)$ is an increasing function:

$$g(x) = \sum_{t=1}^{J_0} \delta_{0t} \exp\left( \sum_{i \in I_t^+} \gamma_{0ti} x_i + x_{n_0+t} \right) + \sum_{t=J_0+1}^{T_0} \delta_{0t} \exp\left( \sum_{i \in I_t^-} \gamma_{0ti} x_i - x_{n_0+t} \right), \quad (1)$$

while

$$h(x) = \min_{k=1,\ldots,k_0} \{u_k(x) - v_k(x)\},\ k_0 = M + T_0, \quad (2)$$

with $u_k(x)$, $v_k(x)$ being increasing functions such that

$$u_k(x) = \begin{cases} \sum\limits_{t=1}^{J_k} \delta_{kt} \exp\left( \sum\limits_{i=1}^{n_0} \gamma_{kti} x_i \right), & k = 1, \ldots, M, \\ \sum\limits_{i \in I_t^-} (-\gamma_{0ti}) x_i + x_{n_0-M+k},\ t = k-M, & k = M+1, \ldots, M+J_0, \\ \sum\limits_{i \in I_t^+} \gamma_{0ti} x_i + x_{n_0-M+k},\ t = k-M, & k = M+J_0+1, \ldots, k_0, \end{cases} \quad (3)$$

and

$$v_k(x) = \begin{cases} \sum\limits_{t=J_k+1}^{T_k} (-\delta_{kt}) \exp\left( \sum\limits_{i=1}^{n_0} \gamma_{kti} x_i \right), & k = 1, \ldots, M, \\ 0, & k = M+1, \ldots, k_0. \end{cases} \quad (4)$$

Based on the above discussion, here, from now on we assume that the original problem (GGP) has been converted to the problem (P), with $g(x)$ increasing and $h(x)$ defined as in (1)–(4), then a robust algorithm will be considered for the problem (P).

## 3 Essential optimization solution

An isolated optimal solution even if computable is often difficult to implement practically because of its instability under small perturbations of the constraints. Therefore, for solving problem (P) we only consider nonisolated feasible solutions of (P) from a practical point of view. This motivates the following definitions (see [15]).

A nonisolated feasible solution $x^*$ of (P) is called an essential optimal solution if $g(x^*) \le g(x)$ for all nonisolated feasible solutions $x$ of (P), i.e. if

$$g(x^*) = \min\{g(x) \mid x \in X_0^*\},$$

where $X_0^*$ denotes the set of all nonisolated feasible solutions of (P). Assume

$$\{x \in X_0 \mid h(x) > 0\} \neq \emptyset. \tag{5}$$

For $\varepsilon \geq 0$, an $x \in X_0$ satisfying $h(x) \geq \varepsilon$ is then called an $\varepsilon$-essential feasible solution, and a nonisolated feasible solution $\overline{x}$ of (P) is called an essential $\varepsilon$-optimal solution if it satisfies

$$g(\overline{x}) - \varepsilon \leq \inf\{g(x) \mid h(x) \geq \varepsilon, x \in X_0\}. \tag{6}$$

Clearly for $\varepsilon = 0$, a nonisolated feasible solution which is essentially $\varepsilon$-optimal is optimal.

The search for an essential $\varepsilon$-optimal solution of (P) can be achieved by the following approach: start from an initial essential feasible solution (the best so far known), find a better essential feasible solution, then reiterate the operation until an evidence is obtained that no better feasible solution than the current best exists. Next, we will show how this approach is formed.

Let $X = [a, b]$ be any subrectangle of $X_0$, and $UB$ be the objective function value of the best so far essential feasible solution $x_0 \in X$ to problem (P)(of course $UB \leq g(b)$). Given an $\varepsilon > 0$, we want to find a nonisolated feasible solution $\overline{x} \in X$ of (P) such that $g(\overline{x}) \leq UB - \varepsilon$, or else establish that none such $\overline{x}$ exists.

Clearly, if $g(a) \geq UB - \varepsilon$, then, since $g$ is increasing, $g(x) \geq UB - \varepsilon$, $\forall x \in X$, so there is no $\overline{x} \in X$ with $g(\overline{x}) < UB - \varepsilon$. If $g(a) < UB - \varepsilon$ and $h(a) > 0$, then $a$ is an essential feasible solution with objective function value less than $UB - \varepsilon$. Therefore, we shall assume without loss generality that

$$h(a) \leq 0, \ g(a) < UB - \varepsilon. \tag{7}$$

Under this assumption, we consider the following auxiliary problem

$$\text{(P1)} \ \max\{h(x) \mid g(x) \leq UB - \varepsilon, x \in X = [a, b]\}.$$

Since the function $h(x)$ is continuous and

$$\{x \in X \mid g(x) \leq UB - \varepsilon\} = \text{cl}\{x \in \text{int}X \mid g(x) < UB - \varepsilon\},$$

it is clear that the problem (P1) is regular (see [15]).

The optimal values of (P) and (P1) are denoted by min(P) and max(P1), respectively. Then the key results for (P) and (P1) are given as follows:

**Theorem 2** *Under assumptions* (5) *and* (7)*:*

(i) *If $h(\tilde{x}) > 0$, where $\tilde{x}$ is a feasible solution of (P1), then $\tilde{x}$ is a nonisolated feasible solution of (P) with $g(\tilde{x}) \leq UB - \varepsilon$. In particular, if $max(P1) = h(x') > 0$, then $x'$ is a nonisolated feasible solution of (P) with $g(x') \leq UB - \varepsilon$.*

(ii) *Let $\hat{x}$ is some nonisolated feasible solution of (P). If $max(P1) < \varepsilon$ and $g(\hat{x}) = UB$, then $\hat{x}$ is an essential $\varepsilon$-optimal solution of (P). If $max(P1) < \varepsilon$ and $UB = g(x^u) + \varepsilon$, then the problem (P) has no nonisolated feasible solution.*

*Proof*

(i) Since $h(a) \leq 0 < h(\tilde{x})$, we have $\tilde{x} \neq a$ and every $x = a + \lambda(\tilde{x} - a)$ with $0 \leq \lambda \leq 1$ satisfies $a \leq x \leq \tilde{x}$. Then, for every $\lambda$ sufficiently close to 1, i.e. every $x$ sufficiently close to $\tilde{x}$, we have $h(x) > 0$, so $x$ is a feasible solution of (P). Furthermore, $g(\tilde{x}) \leq UB - \varepsilon$ because $\tilde{x}$ is feasible to (P1). Therefore, $\tilde{x}$ is a nonisolated feasible solution of (P) satisfying $g(\tilde{x}) \leq UB - \varepsilon$.

(ii) If $\max(\text{P1}) < \varepsilon$ then

$$\sup\{h(x) \mid g(x) \leq UB - \varepsilon,\ x \in X\} < \varepsilon,$$

so for every $x \in X$ satisfying $h(x) \geq \varepsilon$, we must have $g(x) > UB - \varepsilon = g(\hat{x}) - \varepsilon$. Therefore,

$$\inf\{g(x) \mid h(x) \geq \varepsilon,\ x \in X_0\} \geq \inf\{g(x) \mid h(x) \geq \varepsilon,\ x \in X\} \geq g(\hat{x}) - \varepsilon.$$

This means that, $\hat{x}$ is an essential $\varepsilon$-optimal solution of (P).

If $UB = g(x^u) + \varepsilon$, then $\{x \in X_0 \mid h(x) \geq \varepsilon\} = \emptyset$, i.e. the problem (P) has no nonisolated feasible solution, and the proof is complete. □

Since $g(x)$ is continuous and increasing, the problem (P1) is regular (i.e. it has no isolated feasible point). Therefore, solving (P1) is simpler than solving the original problem (P) from a robust optimal point of view. Furthermore, Theorem 2 gives some valuable information, that is, under assumptions (5) and (7), by solving (P1) we can know whether or not an essential feasible solution $x$ of (P) exists such that $g(x) \leq UB - \varepsilon$. Thus, for solving the problem (P) we need consider the problem (P1) in the following.

## 4 Basic operations

In order to solving the problem (P1), a robust approach will be proposed for finding the globally optimal solution of (P1). The main idea of this approach consists of several basic operations: successively refined partitioning of the feasible set; estimation of upper bound for the optimal value of the objective function over each subset generated by the partitions; and the reduction operation by reducing the size of each partition subset without losing any feasible solution currently still of interest. Next, we begin the establishment of the approach with the basic operations needed in a branch-and-bound scheme.

### 4.1 Rectangular partition

To present partition operation, at each iteration of a branch-and-bound algorithm to be presented, assume without loss of generality that a subrectangle of $X_0$ to be subdivided is $X = (X_j)_{n\times 1} = [a, b]$ with $X_j = [a_j, b_j]$. The branching rule is as follows:

(i) Choose a longest edge among the edges of the rectangle $X = X_1 \times X_2 \times \cdots \times X_n$.
(ii) Let $X_j$ denote the rectangle containing the chosen longest edge.
(iii) Let $\omega$ denote the midpoint of this edge.
(iv) $X$ is subdivided into two subrectangles $X^1$ and $X^2$ of equal volume, where $X^1 = X_1 \times X_2 \times \cdots \times \overline{X_j} \times \cdots \times X_n$, $X^2 = X_1 \times X_2 \times \cdots \times \overline{\overline{X_j}} \times \cdots \times X_n$, and $\overline{X_j} = [a_j, \omega]$, $\overline{\overline{X_j}} = [\omega, b_j]$.

It follows easily that the branching process is exhaustive, i.e., if $X^q$ denotes a nested sequence of rectangles (i.e. $X^{q+1} \subset X^q$, for all $q$) formed by the branching process, then for some unique point $x \in R^n$,

$$\bigcap_q X^q = \{x\}.$$

### 4.2 Upper bound

For each rectangle $X \subseteq X_0$, we intend to compute an upper bound $V[\mathrm{P1}(X)]$ of the optimal value of (P1) over $X$. Therefore, We will first generalize an equivalent problem of (P1).

By introducing a new variable $x_{n+1}$, the problem (P1) is equivalent to the following problem:

$$\text{(P2)}\ \max\{x_{n+1} \mid x_{n+1} \le u_k(x) - v_k(x)\ (k = 1, \ldots, k_0),\ g(x) \le UB - \varepsilon, x \in X\}.$$

Our main method for computing an upper bound $V[\mathrm{P1}(X)]$ over $X \subseteq X_0$ is to solve the linear relaxation programming of (P2) by using the following linearization technique.

The linear relaxation of the problem (P2) can be realized by underestimating every function $v_k(x)$ and $g(x)$, and by upper-estimating every function $u_k(x)$, for each $k = 1, \ldots, M$. All the details of this linearization technique for generating the linear relaxation will be given in the following Theorem 3.

For any $X = (x_i)_{n\times 1} = [a, b] \subseteq X_0$ with $x_i = [a_i, b_i]$ and $\forall x \in X$, for simplicity, we denote

$$X_{0t} = \begin{cases} \sum\limits_{i\in I_t^+} \gamma_{0ti} x_i, & t = 1, \ldots, J_0, \\ \sum\limits_{i\in I_t^-} \gamma_{0ti} x_i, & t = J_0 + 1, \ldots, T_0, \end{cases}$$

$$X_{0t}^l = \begin{cases} \sum\limits_{i\in I_t^+} \gamma_{0ti} a_i, & t = 1, \ldots, J_0, \\ \sum\limits_{i\in I_t^-} \gamma_{0ti} b_i, & t = J_0 + 1, \ldots, T_0, \end{cases}$$

$$X_{0t}^u = \begin{cases} \sum\limits_{i\in I_t^+} \gamma_{0ti} b_i, & t = 1, \ldots, J_0, \\ \sum\limits_{i\in I_t^-} \gamma_{0ti} a_i, & t = J_0 + 1, \ldots, T_0, \end{cases}$$

$$X_{kt} = \sum_{i=1}^{n} \gamma_{kti} x_i, \quad t = 1, \ldots, T_k,$$

$$X_{kt}^l = \sum_{i=1}^{n} \gamma_{kti} a_i, \quad t = 1, \ldots, T_k,$$

$$X_{kt}^u = \sum_{i=1}^{n} \gamma_{kti} b_i, \quad t = 1, \ldots, T_k,$$

where $k = 1, \ldots, M$. In addition, let

$$\begin{aligned} A_{kt} &= \frac{\exp(X_{kt}^u) - \exp(X_{kt}^l)}{X_{kt}^u - X_{kt}^l}, \\ \varphi_{kt}(x) &= \exp(X_{kt}), \\ \overline{\varphi}_{kt}(x) &= \exp(X_{kt}^l) + A_{kt}(X_{kt} - X_{kt}^l), \\ \underline{\varphi}_{kt}(x) &= A_{kt}(1 + X_{kt} - \ln A_{kt}), \end{aligned}$$

where $k = 0, 1, \ldots, M,\ t = 1, \ldots, T_k$.

**Theorem 3** *Consider the functions $\varphi_{kt}(x)$, $\overline{\varphi}_{kt}(x)$ and $\underline{\varphi}_{kt}(x)$, for any $x \in X$, where $k = 0, 1, \ldots, M$ and $t = 1, \ldots, T_k$. Then the following two statements are valid.*

(i) *The function* $\overline{\varphi}_{kt}(x)$ *is the (affine) concave envelope of the function* $\varphi_{kt}(x)$ *over* $X$, *and the function* $\underline{\varphi}_{kt}(x)$ *is a supporting hyperplane of* $\varphi_{kt}(x)$, *which is parallel with* $\overline{\varphi}_{kt}(x)$. *Moreover, the functions* $\varphi_{kt}(x)$, $\overline{\varphi}_{kt}(x)$ *and* $\underline{\varphi}_{kt}(x)$ *satisfy*

$$\underline{\varphi}_{kt}(x) \le \varphi_{kt}(x) \le \overline{\varphi}_{kt}(x), \forall x \in X.$$

(ii) *The differences of* $\overline{\varphi}_{kt}(x)$ *and* $\varphi_{kt}(x)$, $\varphi_{kt}(x)$ *and* $\underline{\varphi}_{kt}(x)$ *satisfy* $max_{x\in X}\triangle^1_{kt}(x) = max_{x\in X}\triangle^2_{kt}(x) = exp(X^l_{kt})(1 - Z_{kt} + Z_{kt} ln Z_{kt}) \to 0$ *as* $\omega_{kt} \to 0$, *where*

$$\triangle^1_{kt}(x) = \overline{\varphi}_{kt}(x) - \varphi_{kt}(x),\ \triangle^2_{kt}(x) = \varphi_{kt}(x) - \underline{\varphi}_{kt}(x),$$
$$\omega_{kt} = X^u_{kt} - X^l_{kt},\ z_{kt} = \tfrac{\exp(\omega_{kt})-1}{\omega_{kt}}.$$

*Proof* The proof is similar to Theorem 1 in Ref. [13], it is omitted here. □

*Remark* From Theorem 3, we can follow that the functions $\overline{\varphi}_{kt}(x)$ and $\underline{\varphi}_{kt}(x)$ enough approximate the function $\varphi_{kt}(x)$ as $\omega_{kt} \to 0$, respectively.

Next, we will give the relaxation linear functions of $u_k(x)$, $v_k(x)$ and $g(x)$ over $X$. From Theorem 3, it is obvious that for all $x \in X$ we have

$$u_k(x) \le \overline{u}_k(x) = \sum_{t=1}^{J_k} \delta_{kt}\overline{\varphi}_{kt}(x),$$
$$v_k(x) \ge \underline{v}_k(x) = \sum_{t=J_k+1}^{T_k} (-\delta_{kt})\underline{\varphi}_{kt}(x),$$

where $k = 1, \ldots, M$ and

$$\underline{g}(x) \le g(x) \le \overline{g}(x),$$

where

$$\underline{g}(x) = \sum_{t=1}^{J_0} \delta_{0t}\underline{\varphi}_{0t}(x) + \sum_{t=J_0+1}^{T_0} \delta_{0t}\overline{\varphi}_{0t}(x),$$
$$\overline{g}(x) = \sum_{t=1}^{J_0} \delta_{0t}\overline{\varphi}_{0t}(x) + \sum_{t=J_0+1}^{T_0} \delta_{0t}\underline{\varphi}_{0t}(x).$$

Consequently, we obtain the following linear program (LP2) in $(x, x_{n+1})$ as a linear relaxation of (P2) over the partition set $X$:

$$\begin{aligned}
\text{(LP2)}\ \max\ & x_{n+1} \\
\text{s.t.}\ & x_{n+1} \le \overline{u}_k(x) - \underline{v}_k(x), \quad k = 1, \ldots, M, \\
& x_{n+1} \le u_k(x) - v_k(x), \quad k = M+1, \ldots, k_0, \\
& \underline{g}(x) \le UB - \varepsilon, \\
& x \in X.
\end{aligned}$$

An important property of (LP2)$(X)$ is that its optimal value $V$[LP2$(X)$] satisfies:

$$\max\{h(x) \mid g(x) \le UB - \varepsilon, x \in X\} \le V[\text{LP2}(X)],$$

thus the optimal value of (LP2) provides a valid upper bound for the globally optimal value of (P1) over $X$. However, for a better performance of the upper bound procedure, we can use any tight bound available. For instance, the following procedure may give a better bound.

**Theorem 4**

(i) *If $h(a) \geq 0$ and $g(a) \leq UB - \varepsilon$ then the point $a$ is a nonisolated feasible solution with $g(a) \leq UB - \varepsilon$.*

(ii) *If $g(b) > UB - \varepsilon$ and $z(X) = a + \theta(b-a)$, where $\theta$ satisfies $g(a+\theta(b-a)) = UB - \varepsilon$, and $\rho^i = b + (z_i(X) - b_i)e^i,\ i = 1, \ldots, n$, then an upper bound of $h(x)$ over all $x \in [a, b]$ satisfying $g(x) \leq UB - \varepsilon$ is*

$$\beta(X) = \max_{i=1,\ldots,n} \min_{k=1,\ldots,k_0} \{u_k(\rho^i) - v_k(a)\}. \tag{8}$$

*Proof* (i) Obvious. (ii) Let

$$X^i = [a, \rho^i] = \{x \mid a \leq x \leq \rho^i\} = \{x \in [a, b] \mid a_i \leq x_i \leq z_i(X)\}.$$

□

The function $g(x)$ is increasing and from assumption (7) it follows that $g(a) \leq UB - \varepsilon < g(b)$, therefore $0 \leq \theta < 1$ and $g(z(X)) = UB - \varepsilon$. From the definitions of $z(X)$ and $g(x)$ it is clear that $g(x') > g(z(X)) = UB - \varepsilon$ for all $x' = a + \xi(b-a)$ with $\xi > \theta$. Since for each $x > z(X)$ there exists $x' = a + \xi(b-a)$ with $\xi > \theta$, such that $x \geq x'$, it follows that $g(x) \geq g(x') > UB - \varepsilon$. Let $G = \{x \in [a, b] \mid g(x) \leq UB - \varepsilon\}$, $K = \{x \mid z(X) < x \leq b\}$, and $K_i = \{x \in [a, b] \mid x_i > z_i(X)\}$. Then

$$G \subset [a, b] \setminus K = [a, b] \setminus \cap_{i=1}^n K_i = \cup_{i=1}^n \{x \in [a, b] \mid a_i \leq x_i \leq z_i(X)\} = \cup_{i=1}^n X^i.$$

Since $\beta(X^i) \geq \max\{h(x) \mid x \in X^i\}$, it follows that

$$\begin{aligned} \beta(X) &= \max\{\beta(X^i) \mid i = 1, \ldots, n\} \\ &\geq \max\{h(x) \mid x \in \cup_{i=1}^n X^i\} \\ &\geq \max\{h(x) \mid g(x) \leq UB - \varepsilon, x \in [a, b]\}. \end{aligned}$$

□

Based on the above discussion, for any rectangle $X \subset X_0$, in order to obtain an upper bound $V[\mathrm{P1}(X)]$ of the optimal value of the problem:

$$(\mathrm{P1})\ \max\{h(x) \mid g(x) \leq UB - \varepsilon, x \in X\},$$

we may compute $V[\mathrm{P1}(X)]$ such that

$$V[\mathrm{P1}(X)] = \min\{V[\mathrm{LP2}(X)], \beta(X)\}, \tag{9}$$

where $V[\mathrm{LP2}(X)]$ is the optimal value of (LP2)$(X)$. Clearly, $V[\mathrm{P1}(X)]$ satisfies:

$$\max\{h(x) \mid g(x) \leq UB - \varepsilon, x \in X\} \leq V[\mathrm{P1}(X)] \leq \beta(X). \tag{10}$$

More generally, we shall show in the next section that any upper bound $V[\mathrm{P1}(X)]$ satisfying (10) ensures convergence of the algorithm.

### 4.3 Reduction operation

According to the above discussion, the upper bound of the optimal value of (P1) can be calculated by solving the linear relaxation problem (LP2) of (P2) and $\beta(X)$ (see (8)) inside some rectangle defined by

$$X = (X_i)_{n\times 1} = [a, b] \subseteq X_0 \text{ with } X_i = [a_i, b_i].$$

Clearly, the smaller this rectangle $X$, the tighter the upper bound $V[\mathrm{P1}(X)]$ of (P1), and therefore the closer the feasible solution of (P) will be to the essential $\varepsilon$-optimal solution of (P). To show this, the next results give a reduction operation to reduce the size of this partitioned rectangle without losing any feasible solution currently still of interest.

The reduction operation is based on special cuts that exploit the monotonic structure of the problem. At a given stage of the branch-and-bound algorithm for (P1), let $X = [a, b] \subset X_0$ be a rectangle generated during the partitioning procedure and still of interest. The search for a nonisolated feasible solution of (P) in $[a, b]$ such that $g(x) \leq UB - \varepsilon$ can then be restricted to the set $M \cap [a, b]$, where

$$M := \{x \mid g(x) \leq UB - \varepsilon, h(x) \geq 0\}. \tag{11}$$

Since $h(x) = \min\limits_{k=1,\ldots,k_0} \{u_k(x) - v_k(x)\}$ with $u_k(x)$, $v_k(x)$ being increasing functions (see (2)–(4)), we can also write

$$M = \{x \mid g(x) \leq UB - \varepsilon, u_k(x) - v_k(x) \geq 0, k = 1, \ldots, k_0\}.$$

The reduction operation aims at replacing the rectangle $[a, b]$ with a smaller rectangle $[a', b'] \subset [a, b]$ without losing any point $x \in M \cap [a, b]$, i.e. such that $M \cap [a', b'] = M \cap [a, b]$. The rectangle $[a, b]$ satisfying this condition is denoted by $red[a, b]$. For this purpose, let us first give the following rules (A) and (B) used to construct $\alpha_k^i$ and $\beta_k^i$ $(i = 1, \ldots, n)$, and let $e^i$ denote the $i$th unit vector, i.e. a vector with 1 at the $i$th position and 0 everywhere else, then $red[a, b]$ is derived in Theorem 5 below.

**Rule (A)** *For each $k = 1, \ldots, k_0$, if the function $u_k(b - \alpha(b_i - a_i)e^i)$ is not a constant in single variable $\alpha$ and there exists some $\overline{\alpha}_k^i \in (0, 1)$ such that $\overline{\alpha}_k^i$ is a root of the equation*

$$u_k(b - \alpha(b_i - a_i)e^i) = v_k(a),$$

*then let $\alpha_k^i = \overline{\alpha}_k^i$, otherwise, let $\alpha_k^i = 1$.*

**Rule (B)** *For each $k = 1, \ldots, k_0, k_0 + 1$, if there exists some $\overline{\beta}_k^i \in (0, 1)$ such that either $\overline{\beta}_k^i$ is a root of the equation*

$$v_k(a' + \beta(b_i - a_i')e^i) = u_k(b), \quad for \quad k \in \{1, \ldots, k_0\},$$

*with the function $v_k(a' + \beta(b_i - a_i')e^i)$ not being a constant in single variable $\beta$, or $\overline{\beta}_k^i$ satisfies*

$$g(a' + \overline{\beta}_k^i(b_i - a_i')e^i) = UB - \varepsilon, \quad for\ k = k_0 + 1,$$

*and the function $g(a' + \beta(b_i - a_i')e^i)$ is not a constant in single variable $\beta$, then let $\beta_k^i = \overline{\beta}_k^i$, otherwise, let $\beta_k^i = 1$.*

**Theorem 5**

(i) *If* $g(a) > UB - \varepsilon$, *or* $\min_{k=1,\ldots,k_0}\{u_k(b) - v_k(a)\} < 0$, *then* $M \cap [a,b] = \emptyset$, *i.e.* $red[a,b] = \emptyset$.

(ii) *If* $g(a) \le UB - \varepsilon$, *and* $\min_{k=1,\ldots,k_0}\{u_k(b) - v_k(a)\} \ge 0$, *then* $red[a,b] = [a',b']$ *with*

$$a' = b - \sum_{i=1}^{n} \min_{k=1,\ldots,k_0}\{\alpha_k^i\} \cdot (b_i - a_i)e^i$$

*and*

$$b' = a' + \sum_{i=1}^{n} \min_{k=1,\ldots,k_0+1}\{\beta_k^i\} \cdot (b_i - a_i)e^i,$$

*where* $\alpha_k^i$ *and* $\beta_k^i$ *are defined in Rule (A) and Rule (B), for each* $i = 1,\ldots,n$.

*Proof*

(i) If $g(a) > UB - \varepsilon$, then $g(x) \ge g(a) > UB - \varepsilon$ for every $x \in [a,b]$. If $\min_{k=1,\ldots,k_0}\{u_k(b) - v_k(a)\} < 0$, then

$$\min_{k=1,\ldots,k_0}\{u_k(x) - v_k(x)\} \le \min_{k=1,\ldots,k_0}\{u_k(b) - v_k(a)\} < 0,$$

for every $x \in [a,b]$. In both cases, $M \cap [a,b] = \emptyset$.

(ii) Given any $x = (x_1,\ldots,x_i,\ldots,x_n)^T \in [a,b]$ satisfying $u_k(x) \ge v_k(x), k = 1,\ldots,k_0$ and $g(x) \le UB - \varepsilon$. Let

$$\alpha_{k'}^i \triangleq \min_{k=1,\ldots,k_0}\{\alpha_k^i\}, \quad \beta_{k''}^i \triangleq \min_{k=1,\ldots,k_0+1}\{\beta_k^i\}, \quad i = 1,\ldots,n. \tag{12}$$

We first show that $x \ge a'$ below. Suppose that $x \not\ge a'$, then there exists some $i$ such that

$$x_i < a_i' = b_i - \alpha_{k'}^i(b_i - a_i), \text{ i.e. } x_i = b_i - \alpha(b_i - a_i) \text{ with } \alpha > \alpha_{k'}^i. \tag{13}$$

By the definition of $\alpha_{k'}^i$, we consider the following two cases.

**Case 1** If $\alpha_{k'}^i = 1$, then, from (13) we have $x_i < a_i' = b_i - \alpha_{k'}^i(b_i - a_i) = a_i$, conflicting with $x \in [a,b]$.

**Case 2** If $0 < \alpha_{k'}^i < 1$, from Rule (A) and the definition of $\alpha_{k'}^i$, we can imply that

$$u_{k'}(b - \alpha_{k'}^i(b_i - a_i)e^i) = v_{k'}(a). \tag{14}$$

In addition, by using Rule (A) and the definition of $u_k(x)$, we have $u_{k'}(b - \alpha(b_i - a_i)e^i)$ is strictly decreasing in single variable $\alpha$. Then, it follows from (13) and (14) that

$$u_{k'}(b - (b_i - x_i)e^i) = u_{k'}(b - \alpha(b_i - a_i)e^i) < u_{k'}(b - \alpha_{k'}^i(b_i - a_i)e^i) = v_{k'}(a).$$

Hence

$$u_{k'}(x) \le u_{k'}(b - (b_i - x_i)e^i) < v_{k'}(a) \le v_{k'}(x) \text{ with } x_i = b_i - \alpha(b_i - a_i),$$

conflicting with $u_{k'}(x) - v_{k'}(x) \ge 0$.

Based on the above results, in either case we have $x \ge a'$, i.e. $x \in [a',b]$.

Similarly, we also can show that $x \le b'$. Suppose that $x \not\le b'$, then there exists some $i$ such that

$$x_i > b_i' = a_i' + \beta_{k''}^i(b_i - a_i'), \text{ i.e. } x_i = a_i' + \beta(b_i - a_i') \text{ with } \beta > \beta_{k''}^i. \tag{15}$$

By the definition of $\beta^i_{k''}$, we consider two cases as follows.

**Case 1** If $\beta^i_{k''} = 1$, then from (15), $x_i > b'_i = a'_i + \beta^i_{k''}(b_i - a'_i) = b_i$, conflicting with $x \in [a, b]$.

**Case 2** If $0 < \beta^i_{k''} < 1$, then, from Rule (B) and the definition of $\beta^i_{k''}$, we can imply

$$v_{k''}(a' + \beta^i_{k''}(b_i - a'_i)e^i) = u_{k''}(b) \tag{16}$$

or

$$g(a' + \beta^i_{k''}(b_i - a'_i)e^i) = UB - \varepsilon. \tag{17}$$

Assume that (16) holds. From Rule (B) and the definition of $v_k(x)$, we have $v_{k''}(a' + \beta(b_i - a'_i)e^i)$ is strictly increasing in single variable $\beta$. From (15) and (16), it follows that

$$v_{k''}(a' + (x_i - a'_i)e^i) = v_{k''}(a' + \beta(b_i - a'_i)e^i) > v_{k''}(a' + \beta^i_{k''}(b_i - a'_i)e^i) = u_{k''}(b),$$

and hence

$$v_{k''}(x) \geq v_{k''}(a' + (x_i - a'_i)e^i) > u_{k''}(b) \geq u_{k''}(x) \text{ with } x_i = a'_i + \beta(b_i - a'_i).$$

This conflicts with $u_{k''}(x) \geq v_{k''}(x)$.

Assume that (17) holds. From Rule (B) and the definition of $g(x)$, it is clear that $g(a' + \beta(b_i - a'_i)e^i)$ is a strictly increasing function in single variable $\beta$. Then by (15) and (17), we can deduce

$$g(a' + (x_i - a'_i)e^i) = g(a' + \beta(b_i - a'_i)e^i) > g(a' + \beta^i_{k''}(b_i - a'_i)e^i) = UB - \varepsilon,$$

which means that

$$g(x) > g(a' + (x_i - a'_i)e^i) > UB - \varepsilon \text{ with } x_i = a'_i + \beta(b_i - a'_i).$$

This conflicts with $g(x) \leq UB - \varepsilon$.

From the above proof results, in either case we must have $x \leq b'$, i.e. $x \in [a', b']$, and the proof is complete. □

*Remark 1* In order to obtain $red[a, b]$, in computation of $\alpha^i_{k'}$ and $\beta^i_{k''}$, the form of (12) is more easily implementable than (9) and (10) in Ref. [18]. This is because the later is computed by solving the nonlinear nonconvex programming, but the former involve in solving the roots of several nonlinear or linear equations in a single variable. And the construct of these equations is similar, so their roots are obtained easily by a likewise computing fashion.

*Remark 2* It can easily be verified, the rectangle $[a, b] = red[a, b]$ still satisfies

$$g(a') \leq UB - \varepsilon, \quad \min_{k=1,\ldots,k_0}\{u_k(b') - v_k(a')\} \geq 0.$$

## 5 Algorithm and its convergence

Based on the previous basic operations in Sect. 4, a robust solution algorithm is presented for solving (P). The basic steps of the algorithm are summarized in the following statement.

**Algorithm statement**

**Step 0** Given convergence tolerance $\varepsilon > 0$. If no feasible solution is known, let $UB = g(x^u) + \varepsilon$ with $X_0 = [x^l, x^u]$; otherwise, let $\hat{x}$ be the best nonisolated feasible solution available, $UB = g(\hat{x})$. Let $Q_0 = \{X_0\}$, $F_0 = \emptyset$. Set $q = 0$.

**Step 1** For each rectangle $X \in Q_q$, compute $redX$, i.e. the remainder of $X$, which we can obtain by using the reduction operation. Then, delete $X$ if $redX = \emptyset$; or replace $X$ by $redX$ if $redX = [a', b'] \neq \emptyset$, and compute an upper bound $V[\mathrm{P1}(X)]$ defined in (9) for $h(x)$ over the feasible solutions in $X$ and delete $X$ if $V[\mathrm{P1}(X)] < 0$.

**Step 2** Let $Q'_q$ be the collection of rectangles that results from $Q_q$ after completion of Step 1. Let $F'_q = F_q \cup Q'_q$.

**Step 3** If $F'_q = \emptyset$ then terminate: $\hat{x}$ is an $\varepsilon$-optimal solution of (P) if $UB = g(\hat{x})$, or the problem (P) is infeasible if $UB = g(x^u) + \varepsilon$.

**Step 4** If $F'_q \neq \emptyset$, then let $[a^q, b^q] := X^q \in \mathrm{argmax}\{V[\mathrm{P1}(X)] \mid X \in F'_q\}$, and let $V[\mathrm{P1}]_q = V[\mathrm{P1}(X_q)]$.

**Step 5** If $V[\mathrm{P1}]_q < \varepsilon$, then terminate: $\hat{x}$ is an essential $\varepsilon$-optimal solution of (P) if $UB = g(\hat{x})$, or the problem (P) is $\varepsilon$-essentially infeasible if $UB = g(x^u) + \varepsilon$.

**Step 6** If $V[\mathrm{P1}]_q \geq \varepsilon$, and $g(b^q) > UB - \varepsilon$, compute $x^q = a^q + \gamma_q(b^q - a^q)$ with $\gamma_q$ satisfying $g(a^q + \gamma_q(b^q - a^q)) = UB - \varepsilon$. If $V[\mathrm{P1}]_q \geq \varepsilon$, and $g(b^q) \leq UB - \varepsilon$, let $x^q = a^q$.

(6.1) If $h(x^q) \geq 0$ then $x^q$ is a new nonisolated feasible solution of (P) with $g(x^q) \leq UB - \varepsilon$: if $h(a^q) \geq 0$, reset $\hat{x} \leftarrow a^q$, $UB \leftarrow g(\hat{x})$. Go to Step 7.

(6.2) If $h(x^q) < 0$, go to Step 7, with $\hat{x}$ unchanged.

**Step 7** Divided $X^q$ into two subrectangles by the branching process. Let $Q_{q+1}$ be the collection of these two subrectangles of $X^q$, $F_{q+1} = F'_{q+1} \backslash \{X^q\}$. Increment $q$, and return to Step 1.

The convergence of the proposed algorithm is given as follows.

**Theorem 6** (Convergence result) *The above algorithm terminates after finitely many steps, yielding either an essential $\varepsilon$-optimal solution of* (P)*, or an evidence that the problem is essentially infeasible.*

*Proof* In Step 3, the event $F'_q = \emptyset$ implies that we cannot find any feasible solution $x$ with $g(x) \leq UB - \varepsilon = g(\hat{x}) - \varepsilon$, hence the conclusion in Step 3 is correct. If $V(P1)_q < \varepsilon$, then $\max(\mathrm{P1}) < \varepsilon$ (see (10)), hence by Theorem 2, the same conclusion in Step 5. Observe that in Step 6, the point $x^q$ exists and satisfies $g(x^q) = UB - \varepsilon$, so if $h(x^q) \geq 0$, then $x^q$ is a nonisolated feasible solution with $g(x^q) \leq g(\hat{x}) - \varepsilon$, justifying Step (6.1). Thus the conclusion is correct if one of the following events occurs:

$$F'_q = \emptyset,\ V[\mathrm{P1}]_q < \varepsilon,\ h(x^q) > 0.$$

It remains to show that at least one of these events must occur, i.e. that for sufficiently large $k$ Steps 6 and 7 cannot occur. To this end, suppose now that the algorithm is infinite. Since each occurrence of Step (6.1) decreases the current best value at least by $\varepsilon > 0$ while $g(x)$ is bounded below it follows that Step (6.1) cannot occur infinitely often. Consequently, for all $q$ sufficiently large, $\hat{x}$ is unchanged, and $h(x^q) \leq 0$, while $V[\mathrm{P1}]_q \geq \varepsilon$. But, as $q \longrightarrow \infty$, we have, by exhaustiveness of the subdivision, diam $X_q \longrightarrow 0$, i.e. $\|b^q - a^q\| \longrightarrow 0$. Denote by $\tilde{x}$ the common limit of $b^q$ and $a^q$ as $q \longrightarrow \infty$. Since

$$\varepsilon \leq V[\mathrm{P1}]_q \leq \min_{k=1,\ldots,k_0} [u_k(b^q) - v_k(a^q)],$$

it follows that

$$\varepsilon \leq \lim_{q \longrightarrow +\infty} V[\mathrm{P1}]_q \leq \min_{k=1,\ldots,k_0} [u_k(\tilde{x}) - v_k(\tilde{x})] = h(\tilde{x}).$$

But by continuity, $h(\tilde{x}) = \lim_{q \longrightarrow +\infty} h(x^q) \leq 0$, a contradiction. Therefore, the algorithm must be finite, and the proof is complete. □

## 6 Numerical results

We now report some numerical examples to verify the performance of the proposed algorithm. All test problems were implemented on a Pentium (R) 4 CPU 2.66 GHz with 512 MB memory microcomputer. Some computational results are summarized in Tables 1 and 2 to show the potential and feasibility of the proposed algorithm, compared with other existing methods, such as linear relaxation.

To illustrate how the proposed robust algorithm works, we first give a simple example to show the solving procedure of the proposed algorithm. Furthermore, this example is meant to show that by using $V[\mathrm{P1}(X)] = \min\{V[\mathrm{LP2}(X)], \beta(X)\}$ (see (9)) one could compute an upper bound as tight as we wish.

**Example 1** (See [18,19])

$$\begin{aligned}
&\min\ y_3^{0.8} y_4^{1.2}\\
&\mathrm{s.t.}\ y_1 y_4^{-1} + y_2^{-1} y_4^{-1} \le 1,\\
&\quad -y_1^{-2} y_3^{-1} - y_2 y_3^{-1} \le 1,\\
&\quad 0.1 \le y_1 \le 1\,,\ 5 \le y_2 \le 10\,,\ 8 \le y_3 \le 15\,,\ 0.01 \le y_4 \le 1.
\end{aligned}$$

First, let $x_i = \exp y_i,\ i = 1, 2, 3, 4$, we transform the above problem into the form:

$$\text{(P)}\ \min\{g(x) \mid h(x) \ge 0,\ x \in X_0 = [x^l, x^u]\},$$

where,

$g(x) = \exp(0.8x_3 + 1.2x_4)$,
$h(x) = \min\{\exp(x_2 + x_4) - (\exp(x_1 + x_2) + 1),\ \exp(2x_1 + x_3) + \exp(2x_1 + x_2) + 1\}$,
$x^l = (\ln 0.1, \ln 5, \ln 8, \ln 0.01)$,
$x^u = (\ln 1, \ln 10, \ln 15, \ln 1)$.

Give an $\varepsilon > 0$, and let $UB$ be the best so far known upper bound of the optimal value of $g(x)$. Then, we consider the auxiliary problem over a partition set $X \subseteq X_0$ as follows:

$$\text{(P1)}\ \max\{h(x) \mid g(x) \le UB - \varepsilon, x \in X = [a, b]\}.$$

To compute an upper bound of the optimal value of (P1) over $X$, an additional variable $x_5$ is introduced, thus the problem (P1) is equivalent to the following problem:

$$\begin{aligned}
&\text{(P2)}\ \max\ x_5\\
&\quad \mathrm{s.t.}\ x_5 \le \exp(x_2 + x_4) - (\exp(x_1 + x_2) + 1),\\
&\qquad x_5 \le \exp(2x_1 + x_3) + \exp(2x_1 + x_2) + 1,\\
&\qquad \exp(0.8x_3 + 1.2x_4) \le UB - \varepsilon,\\
&\qquad x \in X.
\end{aligned}$$

Consequently, from Theorem 3, we can obtain the linear relaxation program (LP2) in $(x, x_5)$ of the problem (P2) over the partition set $X$ as follows:

$$\begin{aligned}
&\max\ x_5\\
&\mathrm{s.t.}\ x_5 \le \exp(a_2 + a_4) + A1(x_2 + x_4 - (a_2 + a_4)) - (A2(1 + x_1 + x_2 - \ln A_2) + 1),\\
&\quad x_5 \le \exp(2a_1 + a_3) + A3(2x_1 + x_3 - (2a_1 + a_3)) + \exp(2a_1 + a_2)\\
&\qquad + A4(2x_1 + x_2 - (2a_1 + a_2)) + 1,\\
&\quad A_5(1 + 0.8x_3 + 1.2x_4 - \ln A_5) \le UB - \varepsilon,\\
&\quad x \in X,
\end{aligned}$$

where,

$$A_1 = \frac{\exp(b_2 + b_4) - \exp(a_2 + a_4)}{(b_2 + b_4) - (a_2 + a_4)}, \quad A_2 = \frac{\exp(b_1 + b_2) - \exp(a_1 + a_2)}{(b_1 + b_2) - (a_1 + a_2)},$$
$$A_3 = \frac{\exp(2b_1 + b_3) - \exp(2a_1 + a_3)}{(2b_1 + b_3) - (2a_1 + a_3)}, \quad A_4 = \frac{\exp(2b_1 + b_2) - \exp(2a_1 + a_2)}{(2b_1 + b_2) - (2a_1 + a_2)},$$
$$A_5 = \frac{\exp(0.8b_3 + 1.2b_4) - \exp(0.8a_3 + 1.2a_4)}{(0.8b_3 + 1.2b_4) - (0.8a_3 + 1.2a_4)}.$$

Then, by solving the linear program subproblem (LP2($X$)), its optimal value $V[\text{LP2}(X)]$ can be obtained and is severed as a valid upper bound for the globally optimal value of (P1) over $X$. In addition, Theorem 4 gives another upper bound $\beta(X)$ for the problem (P1). Therefore, in order to obtain a better upper bound $V[\text{P1}(X)]$ of the optimal value of the problem (P1), we compute $V[\text{P1}(X)]$ such that $V[\text{P1}(X)] = \min\{V[\text{LP2}(X)], \beta(X)\}$ in Step 1 of the proposed algorithm (see (9)).

Following the robust solution method, solving Example 1 with tolerance $\varepsilon = 10^{-5}$, the proposed algorithm yielded an essential $\varepsilon$-optimal solution

$$\hat{y} = (0.1000, 9.9999, 8.0000, 0.2000)$$

with objective function value 0.7651 at iteration 124, and confirmed its essential $\varepsilon$-optimality at iteration 132. The computation required 1.484 s, and got the essential $\varepsilon$-optimal solution through 34 cycles of incumbent transcending, with intermediate results for the first nine and last nine cycles as given in Table 1.

(By cycle we mean a sequence of iterations required for transcending a given incumbent; $\hat{y}$ is the new incumbent found at the end of the cycle, and Iter indicates the iteration where $\hat{y}$ is found.)

Additionally, it has been observed that, when $V[\text{P1}(X)] = \beta(X)$ is utilized as an upper bound of the optimal value to problem (P1) (as defined in [15]), the computational cost of Example 1 will be different from choosing $V[\text{P1}(X)] = \min\{V[\text{LP2}(X)], \beta(X)\}$. Indeed, if the linearization technique don't be considered ( i.e. the linear programming (LP2) don't be used), then, with same $\varepsilon = 10^{-5}$, the robust solution algorithm found the same essential $\varepsilon$-optimal solution and essential $\varepsilon$-optimal value at iteration 156, and confirmed its essential $\varepsilon$-optimality at iteration 175. The computation required 2.000 s on the same computer.

This illustrates that the linearization technique may improve the computational efficiency and it may be necessary to the robust solution method.

Furthermore, we choose seven other examples to test our algorithm, which are all come from the literature, and the computational results of some examples are listed in Table 2 below.

**Example 2** (See [13,18,20])

$$\begin{aligned} \min\ & 0.5y_1y_2^{-1} - y_1 - 5y_2^{-1} \\ \text{s.t.}\ & 0.01y_2y_3^{-1} + 0.01y_2 + 0.0005y_1y_3 \le 1, \\ & 70 \le y_1 \le 150,\ 1 \le y_2 \le 30,\ 0.5 \le y_3 \le 21. \end{aligned}$$

**Example 3** (See [19])

$$\begin{aligned} \min\ & y_0 \\ \text{s.t.}\ & 3.7y_0^{-1}y_1^{0.85} + 1.985y_0^{-1}y_1 + 700.3y_0^{-1}y_2^{-0.75} \le 1, \\ & 0.7673y_2^{0.05} - 0.05y_1 \le 1, \\ & 5 \le y_0 \le 15,\ 0.1 \le y_1 \le 5,\ 380 \le y_2 \le 450. \end{aligned}$$

**Table 1** Numerical results for Example1

| Cycle | $\hat{y}$ | $g(\hat{y})$ | Iter |
|---|---|---|---|
| 1 | (0.200625,7.603680,11.70019,0.419761) | 2.524389 | 3 |
| 2 | (0.177828,7.071067,8.000000,0.392486) | 1.718151 | 6 |
| 3 | (0.100000,7.071068,8.000000,0.284826) | 1.169413 | 7 |
| 4 | (0.118704,8.830369,9.647354,0.251400) | 1.169403 | 17 |
| 5 | (0.123975,9.250108,8.996008,0.263391) | 1.169393 | 18 |
| 6 | (0.100000,7.351431,8.000000,0.244076) | 0.971635 | 19 |
| 7 | (0.118703,9.396957,8.785103,0.229307) | 0.971625 | 22 |
| 8 | (0.111344,9.617737,8.483395,0.234710) | 0.971615 | 24 |
| 9 | (0.100000,8.574048,8.000000,0.225942) | 0.885665 | 25 |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| 26 | (0.100000,9.995954,8.004320,0.200054) | 0.765660 | 93 |
| 27 | (0.100031,9.995127,8.001718,0.200095) | 0.765650 | 96 |
| 28 | (0.100000,9.995954,8.000000,0.200054) | 0.765329 | 99 |
| 29 | (0.100013,9.999154,8.001220,0.200031) | 0.765319 | 108 |
| 30 | (0.100000,9.998664,8.000000,0.200021) | 0.765178 | 109 |
| 31 | (0.100005,9.999482,8.000449,0.200011) | 0.765168 | 121 |
| 32 | (0.100005,9.999703,8.000365,0.200010) | 0.765158 | 122 |
| 33 | (0.100003,9.999786,8.000299,0.200009) | 0.765148 | 123 |
| 34 | (0.100000,9.999903,8.000000,0.200004) | 0.765102 | 124 |

**Example 4** (See [13,20])

$$
\begin{aligned}
\min\ & 168y_1y_2 + 3651.2y_1y_2y_3^{-1} + 4\times 10^4 y_4^{-1}\\
\text{s.t.}\ & 1.0425y_1y_2^{-1} \le 1,\\
& 3.5\times 10^{-4}y_1y_2 \le 1,\\
& 1.25y_1^{-1}y_4 + 41.63y_1^{-1} \le 1,\\
& 40 \le y_1 \le 44\,,\ 40 \le y_2 \le 45\,,\ 60 \le y_3 \le 70\,,\ 0.1 \le y_4 \le 1.4.
\end{aligned}
$$

**Example 5** (See [13,20])

$$
\begin{aligned}
\min\ & 5.3578y_3^2 + 0.8357y_1y_5 + 37.2392y_1\\
\text{s.t.}\ & 0.00002584y_3y_5 - 0.00006663y_2y_5 - 0.0000734y_1y_4 \le 1,\\
& 0.000853007y_2y_5 + 0.00009395y_1y_4 - 0.00033085y_3y_5 \le 1,\\
& 1330.3294y_2^{-1}y_5^{-1} - 0.42y_1y_5^{-1} - 0.30586y_2^{-1}y_3^2y_5^{-1} \le 1,\\
& 0.00024186y_2y_5 + 0.00010159y_1y_2 + 0.00007379y_3^2 \le 1,\\
& 2275.1327y_3^{-1}y_5^{-1} - 0.2668y_1y_5^{-1} - 0.40584y_4y_5^{-1} \le 1,\\
& 0.00029955y_3y_5 + 0.00007992y_1y_3 + 0.00012157y_3y_4 \le 1,\\
& 78.0 \le y_1 \le 102.0\,,\ 33.0 \le y_2 \le 45.0\,,\ 27 \le y_3 \le 45.0,\\
& 27.0 \le y_4 \le 45.0\,,\ 27.0 \le y_5 \le 45.0.
\end{aligned}
$$

**Table 2** Computational results for test examples

| No. | Ref. | Iter | $\epsilon$ | Optimal solution | Optimal value |
|---|---|---|---|---|---|
| 1 | [ours] | 132 | $10^{-5}$ | (0.1000,9.9999,8.0000,0.2000) | 0.7651 |
| | [18] | 175 | | (0.1015,7.31972,8.0169,0.2395) | 0.9514 |
| | [19] | 171 | | (0.1358, 9.9324, 8.6973, 0.2365) | 1.0000 |
| 2 | [ours] | 328 | 0.01 | (150,30,4.9620) | −147.6667 |
| | [13] | 1829 | | (88.724706796,7.672652781,1.317862596) | −83.2497... |
| | [18] | 1754 | | (88.6274,7.9621,1.3215) | −83.6898 |
| | [20] | 1809 | | | −83.2497... |
| 3 | [ours] | 6472 | 0.01 | (12.0475,0.8167,444.9416) | 12.0475 |
| | [19] | 67 | | (11.9538,0.8150,445.1249) | 11.9538 |
| 4 | [ours] | 968 | 0.1 | (43.0473,44.9317,69.9359,1.1338) | $4.6120 \times 10^5$ |
| | [13] | 2100 | | (43.0137..., 44.8148...,66.4239..., 1.1070...) | 623249.876... |
| | [20] | 1717 | | | 623249.8752... |
| 5 | [ours] | 122 | 0.1 | (78.2135,33.2135,29.6588,44.757,37.6808) | $1.008851 \times 10^4$ |
| | [13] | 341 | | (78, 32.9999..., 29.9957..., 45, 36.7753...) | 10122.4931... |
| | [20] | 204 | | | 10122.3811... |

In Table 2, the notations have been used for column headers: No.: number; Ref.: reference; Iter: number of algorithm iteration.

**Example 6** (See [21])

$$
\begin{aligned}
\min\ & x_{12}(12.626260 - 1.231059x_1) + x_{13}(12.626260 - 1.231059x_2) \\
& + x_{14}(12.626260 - 1.231059x_3) + x_{15}(12.626260 - 1.231059x_4) \\
& + x_{16}(12.626260 - 1.231059x_5) \\
\text{s.t.}\ & x_{12} - x_{11} \le 0,\ x_{11} - x_{12} \le 50,\ x_{10} - x_4 \le 0,\ x_9 - x_{10} \le 0, \\
& x_8 - x_9 \le 0,\ 2x_7 - x_1 \le 1,\ x_3 - x_4 \le 0,\ x_2 - x_3 \le 0,\ x_1 - x_2 \le 0, \\
& x_4x_{16} - 50x_4 - x_5x_{16} \le -450, \\
& 50x_4 + x_5x_{16} + x_{10}x_{15} - 50x_{10} - x_4x_{15} - x_4x_{16} \le 0, \\
& 50x_{10} + x_4x_{15} + x_9x_{14} - 50x_9 - x_3x_{14} - x_8x_{15} \le 0, \\
& 50x_8 + 50x_9 + x_3x_{14} + x_8x_{13} - x_2x_{13} - x_9x_{14} \le 500, \\
& 50x_7 + x_2x_{13} + x_7x_{12} - 50x_8 - x_1x_{12} - x_8x_{13} \le 0, \\
& 50x_8 + x_1x_{12} + x_8x_{13} - 50x_7 - x_2x_{13} - x_7x_{12} \le 0, \\
& x_6x_{11} + x_1x_{12} - x_7x_{11} - x_6x_{12} \le 0, \\
& 100x_6 + 0.0975x_1^2 - 3.475x_1 - 9.75x_1x_6 \le 0, \\
& 100x_7 + 0.0975x_2^2 - 3.475x_2 - 9.75x_2x_7 \le 0, \\
& 100x_8 + 0.0975x_3^2 - 3.475x_3 - 9.75x_3x_8 \le 0, \\
& 100x_9 + 0.0975x_4^2 - 3.475x_4 - 9.75x_4x_9 \le 0, \\
& 100x_{10} + 0.0975x_5^2 - 3.475x_5 - 9.75x_5x_{10} \le 0, \\
& (1, 1, 9, 9, 9, 1, 1, 1, 1, 1, 50, 0, 1, 50, 50, 0) \le x, \\
& x \le (8.037732, 9, 9, 9, 9, 1, 4.518866, 9, 9, 9, 100, 50, 50, 50, 50, 0).
\end{aligned}
$$

By using a well devised branch and cut algorithm in Ref. [**?** ], with $\varepsilon = \eta = 10^{-5}$, an $(\varepsilon, \eta)-$ approximate optimal solution was given as

$$x^* = (8.03773, 8.161, 9, 9, 9, 1, 1.07026, 1.90837, \\ 1.90837, 1.90837, 50.5042, 0.504236, 7.26387, 50, 50, 0)$$

with objective function value 174.788. However, in this paper, with $\varepsilon = 10^{-5}$, the robust solution algorithm found the essential $\varepsilon$-optimal solution

$$\hat{x} = (8.037732, 9, 9, 9, 9, 1, 1, 1.15686274509804, \\ 1.15686274509804, 1.15686274509804, 50, 0, 1, 50, 50, 0)$$

with objective function value 156.219629 at iteration 3.

**Example 7** (See [21])

$$\begin{aligned}
\min\ & (3 + x_1x_3)(x_1x_2x_3x_4 + 2x_1x_3 + 2)^{2/3} \\
\text{s.t.}\ & -3(2x_1x_2 + 3x_1x_2x_4)(2x_1x_3 + 4x_1x_4 - x_2) \\
& -(x_1x_3 + 3x_1x_2x_4)(4x_3x_4 + 4x_1x_3x_4 + x_1x_3 - 4x_1x_2x_4)^{1/3} \\
& + 3(x_4 + 3x_1x_3x_4)(3x_1x_2x_3 + 3x_1x_4 + 2x_3x_4 - 3x_1x_2x_4)^{1/4} \le -309.219315, \\
& -2(3x_3 + 3x_1x_2x_3)(x_1x_2x_3 + 4x_2x_4 - x_3x_4)^2 \\
& + (3x_1x_2x_3)(3x_3 + 2x_1x_2x_3 + 3x_4)^4 - (x_2x_3x_4 + x_1x_3x_4)(4x_1 - 1)^{3/4} \\
& -3(3x_3x_4 + 2x_1x_3x_4)(x_1x_2x_3x_4 + x_3x_4 - 4x_1x_2x_3 - 2x_1)^4 \le -78243.910551, \\
& -3(4x_1x_3x_4)(2x_4 + 2x_1x_2 - x_2 - x_3)^2 \\
& + 2(x_1x_2x_4 + 3x_1x_3x_4)(x_1x_2 + 2x_2x_3 + 4x_2 - x_2x_3x_4 - x_1x_3)^4 \le 9618, \\
& 0 \le x_i \le 5, \quad i = 1, 2, 3, 4.
\end{aligned}$$

With $\varepsilon = 0.1$ the essential $\varepsilon$-optimal solution was found as

$$\hat{x} = (4.99671032804590, 0.02158432903879, 0.04460296046798, 4.99584081114434)$$

with objective function value 5.88861758106183 at iteration 12097.

**Example 8** (See [21])

$$\begin{aligned}
\min\ & 4(x_1^2x_3 + 2x_1^2x_2x_3^2x_5 + 2x_1^2x_2x_3)(5x_1^2x_3x_4^2x_5 + 3x_2)^{3/5} \\
& + 3(2x_4^2x_5^2)(4x_1^2x_4 + 4x_2x_5)^{5/3} \\
\text{s.t.}\ & -2(2x_1x_5 + 5x_1^2x_2x_4^2x_5)(3x_1x_4x_5^2 + 5 + 4x_3x_5^2)^{1/2} \le -7684.470329, \\
& 2(2x_1x_2^2x_3x_4^2)(2x_1x_2x_3x_4^2 + 2x_2x_4^2x_5 - x_1^2x_5^2)^{3/2} \le 1286590.314422, \\
& 0 \le x_i \le 5, \quad i = 1, 2, 3, 4, 5.
\end{aligned}$$

With $\varepsilon = 0.1$, the robust solution algorithm found the essential $\varepsilon$-optimal solution

$$\hat{x} = (4.99290412514017, 4.99136112876356, 0.14607278644884, \\ 1.17375845734759, 0.95455337948153)$$

with objective function value 28745.10753904065 at iteration 12014.

From the above computational results, we can obtain that solving all of the problems by the robust solution algorithm in this paper yields the essential $\varepsilon$-optimal solutions with much better objective function values and being feasible. In addition, it is observed that, in the example 3, the computational solution $y^* = (11.9538, 0.8150, 445.1249)$ of Ref. [19] doesn't satisfy the constraint $0.7673y_2^{0.05} - 0.05y_1 \le 1$, i.e. $y^*$ is infeasible, but our solution is feasible. This illustrates the potential advantage of the robust solution approach: not only a robust solution is obtained, less computational effort may be required for reaching a better objective function value.

**Acknowledgements** The authors are grateful to the responsible editor and the anonymous referees for their valuable comments and suggestions, which have greatly improved the earlier version of this paper. The research is supported by the National Natural Science Foundation of China under Grant 10671057 and by Program for Science & Technology Innovation Talents in Universities of Henan Province.

## References

1. Avriel, M., Williams, A.C.: An extension of geometric programming with applications in engineering optimization. J. Eng. Math. **5**(3), 187–199 (1971)
2. Jefferson, T.R., Scott, C.H.: Generalized geometric programming applied to problems of optimal control: I. Theory. J. Optim. Theory Appl. **26**, 117–129 (1978)
3. Nand, K.J.: Geometric programming based robot control design. Comput. Indust. Eng. **29**(1–4), 631–635 (1995)
4. Das, K., Roy, T.K., Maiti, M.: Multi-item inventory model with under imprecise objective and restrictions: a geometric programming approach. Production Plan. Control **11**(8), 781–788 (2000)
5. Jae Chul, C., Bricker Dennis, L.: Effectiveness of a geometric programming algorithm for optimization of machining economics models. Comput. Oper. Res. **23**(10), 957–961 (1996)
6. EI Barmi, H., Dykstra, R.L.: Restricted multinomial maximum likelihood estimation based upon Fenchel duality. Stat. Probab. Lett. **21**, 121–130 (1994)
7. Bricker, D.L., Kortanek, K.O., Xu, L.: Maximum Likelihood Estimates with Order Restrictions on Probabilities and Odds Ratios: A Geometric Programming Approach. Applied Mathematical and Computational Sciences, the University of IA, Iowa City, IA (1995)
8. Jagannathan, R.: A stochastic geometric programming problem with multiplicative recourse. Oper. Res. Lett. **9**, 99–104 (1990)
9. Sönmez, A.I., Baykasoglu, A., Dereli, T., Filiz, I.H.: Dynamic optimization of multipass milling operations via geometric programming. Int. J. Machine Tools Manuf. **39**, 297–320 (1999)
10. Scott, C.H., Jefferson, T.R.: Allocation of resources in project management. Int. J. Syst. Sci. **26**, 413–420 (1995)
11. Maranas, C.D., Floudas, C.A.: Global optimization in generalized geometric programming. Comput. Chem. Eng. **21**(4), 351–369 (1997)
12. Rijckaert, M.J., Martens, X.M.: Analysis and optimization of the Williams-Otto process by geometric programming. AICHE J **20**(4), 742–750 (1974)
13. Shen, P., Zhang, K.: Global optimization of signomial geometric programming using linear relaxation. Appl. Math. Comput. **150**, 99–114 (2004)
14. Wang, Y., Zhang, K., Gao, Y.: Global optimization of generalized geometric programming. Comput. Math. Appl. **48**, 1505–1516 (2004)
15. Tuy, H.: Robust solution of nonconvex global optimization problems. J. Glob. Optim. **32**, 357–374 (2005)
16. Tuy, H.: Monotonic optimization: problems and solution approaches. SIAM J. Optim. **11**(2), 464–494 (2000)
17. Tuy, H., Al-Khayyal, F., Thach, P.T. : Monotonic optimization: branch and cut methods. In: Audet, C., Hansen, P., Savard, G. (eds.) Essays and Surveys on Global Optimization, pp. 39–38. Springer, Berlin (2005)
18. Qu, S.-J., Zhang, K.-C., Ji, Y.: A new global optimization algorithm for signomial geometric programming via Lagrangian relaxation. Appl. Math. Comput. **184**(2), 886–894 (2007)
19. Wang, Y., Liang, Z.: A deterministic global optimization algorithm for generalized geometric programming. Appl. Math. Comput. **168**, 722–737 (2005)
20. Shen, P., Jiao, H.: A new rectangle branch-and-pruning approach for generalized geometric programming. Appl. Math. Comput. **183**, 1027–1038 (2006)
21. Tuy, H.: Polynomial optimization: a robust approach. Pacific J. Optim. **1**, 357–374 (2005)