Contemporary Mathematics
https://ojs.wiserpub.com/index.php/CM/

UNIVERSAL WISER PUBLISHER

Research Article

Solving Mixed Integer Geometric Programming Problems

Elahe Hajimohamady[1], Mansour Saraj[2*], M. Momeni[3], F. Kiany[3]

[1] Department of Mathematics, Ahvaz Branch, Islamic Azad University, Ahvaz-Iran
[2] Department of Mathematics, Faculty of Mathematical Sciences and Computer, Shahid Chamran University of Ahvaz, Ahvaz-Iran
[3] Department of Mathematics, Faculty of Mathematical Sciences, Ahvaz Branch, Islamic Azad University, Ahvaz-Iran
E-mail: msaraj@scu.ac.ir

Received: 18 November 2024; **Revised:** 8 January 2025; **Accepted:** 16 July 2025

Abstract: Geometric Programming (GP) with integer variables is an area of active research that has drawn significant interest over the past few decades. Our approach first converts all non-convex terms of the problem into convex terms using the negative power transformation method. The convex terms are then addressed through a proposed piecewise linearization method to obtain integer solutions. Finally, the problem is solved using the nonlinear Branch and Bound algorithm. In this paper, we propose the use of the linear Branch and Bound method following piecewise linear approximation. Computational results demonstrate that our proposed method is significantly more efficient and faster than conventional methods. A numerical example is provided to illustrate the efficiency and practicality of the proposed approach.

***Keywords*:** Geometric Programming (GP), mixed integer non-linear programming, branch-and-bound algorithm, negative power transformation, piece-wise linearization method

MSC: 90C05, 90C10, 90C86

1. Introduction

Geometric Programming (GP) is a widely used optimization method, particularly in minimizing engineering design costs. It is a nonlinear optimization problem that has gained substantial attention from researchers in recent years.

The general formulation of a Geometric Programming Problem (GPP) is as follows:

$$\begin{aligned} &\text{Minimize } \ f_0(x) \\ &f_i(x) \le 1 \quad i = 1, \ldots, m \\ &g_j(x) = 1 \quad j = 1, \ldots, p \end{aligned} \tag{1}$$

Where $g_j(x)$, $j = 1, \ldots, p$ are monomials and $f_i(x)$ for $i = 1, \ldots, m$ are posynomials.

Many optimization problems in science, engineering, and other fields involve discrete decision-making and dynamic nonlinear systems that affect the final design quality. Geometric programming extends the application of linear programming to nonlinear optimization problems. It has been successfully applied in diverse fields such as analog/digital circuit design, chemical engineering, mechanical engineering, power control, and communication network systems.

Several studies have proposed methods for solving mixed integer geometric programming problems. Global optimization of mixed integer signomial fractional programing problems is a recent research by Saraj et al. [1]. Since the 1960s, applications have been found in mechanical, civil, and chemical engineering, statistics, finance, and economics. Research has increasingly focused on integer variables and nonlinear systems in optimization, as they directly impact problem-solving efficiency.

Recent studies have explored different methods for tackling these challenges. Zhang et al. [2] conducted a survey on solving mixed integer programming using machine learning. Melo et al. [3] introduced a hybrid algorithm for convex mixed integer nonlinear programming, refining the Branch and Bound technique. Huang et al. studied the problem of learning to select cuts for efficient mixed integer programming [4]. In 2024, Chey et al. investigated machine learning approaches to optimize continuous and mixed integer problems [5]. Lundle et al. [6, 7] proposed a global optimization method using linear relaxation for signomial geometric programming. Other research includes learning algorithms for mixed integer programs, temporal pattern mining, and efficient global optimization approaches [8, 9]. Bazikar and Saraj investigated the problem of solving linear multi-objective geometric problems via reference point approach [10]. Non-linear integer programming is done by Hemmeck et al. [11]. The problem of mixed integer non-linear optimization problem is investigated by Belotti et al. in 2013 [12].

This paper develops an optimization approach for solving geometric programming problems. The paper is structured as follows:

1. Converting Non-Convex Terms into Convex One on using negative power transformation methods and linear approximations. Where only $\lceil \log 2(m-1) \rceil$ binary variables are used.
2. Applying the Nonlinear Branch and Bound Algorithm–Implementing an effective solution approach.
3. Innovation and Numerical Examples–Introducing the proposed method and demonstrating its practical applications.
4. Conclusion & discussion.

2. Convex approximation

Lemma 1 Let A be a nonzero $n \times n$ symmetric matrix.

(a) If A is positive semi definite, then $\Delta_k \geq 0$, $k = 1, 2, ..., n$, and not all $\Delta_k = 0$.

(b) A is positive definite iff $\Delta_k > 0$, for all $k = 1, 2, ..., n$.

This guarantees that the transformed function has a convex Hessian matrix, ensuring convexity.

Lemma 2 Let A be a nonzero symmetric matrix. Then, the following conditions will hold.

(a) A is positive semi definite iff all of its eigenvalues are nonnegative.

(b) A is positive definite iff all of its eigenvalues are positive.

Let $f_1 : R^n_{++} \to R$ be defined as $f_1(x) = c_1 \prod_{i=1}^{n} x_i^{\alpha_i}$, where $c_1 > 0$ and $\alpha_i \leq 0$ for all $i = 1, 2, ..., n$. Then f_1 is a convex function.

Proof. Since $c_1 > 0$, it is enough to show that $\widetilde{f}(x) = \prod_{i=1}^{n} x_i^{\alpha_i}$ is convex. Let $g(x) = \ln \widetilde{f}_1(x) = \sum_{i=1}^{n} \ln x_i^{\alpha_i} = \sum_{i=1}^{n} \alpha_i \ln x_i$. Then, we have

$$\nabla g(x) = \left[\frac{\alpha_1}{x_1} \ \frac{\alpha_2}{x_2} \ \cdots \ \frac{\alpha_n}{x_n} \right]^T$$

$$\nabla^2 g(x) = \left[\frac{-\alpha_1}{{x_1}^2} \ 0 \cdots 0 \ 0 \ \frac{-\alpha_2}{{x_2}^2} \ \ddots \ 0 \ 0 \ \ 0 \cdots \frac{-\alpha_n}{{x_n}^2} \right]$$

Since $\alpha_i \le 0$ for all $i = 1, 2, ..., n$, and in addition all eigenvalues of $\nabla^2 g(x)$ are nonnegative, then Lemma 1(a) implies that $\nabla^2 g(x)$ is positive semidefinite. Thus, $g(x) = \ln(x)$ is a convex function which yields that $\widetilde{f_1}(x)$ is a convex function. $\square$

Proposition 1 Let $f_2 : R^n_{++} \to R$ be defined as $f_2(x) = c_2 \prod_{i=1}^n x_i^{\alpha_i}$ where $\alpha_i > 0$, $c_2 < 0$ for all $i = 1, 2, ..., n$ with $1 - \sum_{i=1}^n \alpha_i \ge 0$. Then f_2 is a convex function.

Proof. It is not hard to compute that $[\nabla f_2(x)]_i = c_2 \alpha_i x_i^{\alpha_i - 1} \prod_{j=1,\, j\ne i}^n x_j^{\alpha_j}$ $[\nabla f_2(x)]_i$. In other words,

$$\nabla f_2(x) = \left[c_2 \alpha_1 x_1^{\alpha_1 - 1}\ x_2^{\alpha_2}\ \dots\ x_n^{\alpha_n}\ c_2 \alpha_1 x_1^{\alpha_1 - 1}\ x_2^{\alpha_2}\ \dots\ x_n^{\alpha_n}\ \dots\ c_2 \alpha_n x_1^{\alpha_1 - 1}\ x_2^{\alpha_2}\ \dots\ x_n^{\alpha_n} \right]$$

In addition, it can be verified that

$$[\nabla^2 f_2(x)]_{i,\, j} = \frac{\partial^2 f_2(x)}{\partial x_i \partial x_j} = \frac{\alpha_i \alpha_j}{\partial x_i \partial x_j} f_2(x) \qquad i \ne j \qquad \frac{\alpha_i(\alpha_i - 1)}{\partial x_i^2} f_2 \quad i = j$$

Moreover, the determinant of $\nabla^2 f_2(x)$ can be computed and be shown by induction as

$$\det\left[\nabla^2 f_2(x)\right] = (-c_2)^n \left(\prod_{i=1}^n \alpha_i x_i^{n\alpha_i - 2} \right) \left(1 - \sum_{i=1}^n \alpha_i \right) \tag{2}$$

We will complete the proof by discussing the following two cases.

Case i: If $1 - \sum_{i=1}^n \alpha_i = 0$, then it is not hard to verify that $\nabla^2 f_2(x)x$ is a zero vector for any $x \in R^n$. Hence, $x^T \nabla^2 f_2(x) = 0$ which shoes that $\nabla^2 f_2(x)$ is a positive semidefinite matrix by definition. Therefore, $f_2(x)$ is convex under Case ii.

Case ii: If $1 - \sum_{i=1}^n \alpha_i > 0$ then we know from (1) that $\Delta_i = (-c_2)^i \left(\prod_{j=1}^i \alpha_j x_j^{n\alpha_j - 2} \right) \left(1 - \sum_{j=1}^i \alpha_j \right)$.

Where Δ_i denotes the i-th principal minor of the Hessian matrix of $f_2(x)$. Note that $c_2 < 0$, $\alpha_i > 0$ for all $i = 1, 2, ..., n$, and $1 - \sum_{i=1}^n \alpha_i > 0$. Therefore, it can be seen that $\Delta_i > 0$ for all $i = 1, 2, ..., n$, which implies (by Lemma 1(b)) that $\nabla^2 f_2(x)$ is a positive definite matrix. This shows that $f_2(x)$ is strictly convex under this case. From all the previous, the desired result follows. $\square$

Proposition 2 [6] Suppose $\underline{x}_i$ and $\underline{x}_i$ be an upper and lower bound of x_i, respectively, and let ε be the accuracy of the computer. The lowest possible value of α_j is in such a way that there exist a computationally distinguishable distance between the inverse transformation function and the piece wise linear under estimator which can be selected by the following statements:

(i) If $\left(\underline{x}_j, \underline{x}_j\right) \le 0$ then $\alpha_j \ge 2\sqrt{\varepsilon} / \left(\mathrm{Ln}\left(\underline{x}_j / \underline{x}_j \right) \right)$.

(ii) If

$$\left(\underline{x}_j, \underline{x}_j\right) > 0 \text{ then } \alpha_j \ge \left(4\,\mathrm{Ln}\left(\underline{x}_j / \underline{x}_j \right) - \sqrt{G} / \left(2\,\mathrm{Ln}\left(\underline{x}_j / \underline{x}_j \right),\ \left(\underline{x}_j, \underline{x}_j \right) \right) \right) \tag{3}$$

Where $G = 16(\underline{x}_j / \underline{x}_j)$, $(\underline{x}_j, \underline{x}_j) - 2\sqrt{\varepsilon}(\underline{x}_j, \underline{x}_j)$ and $\varepsilon \le (\mathrm{Ln}(\underline{x}_j / \underline{x}_j) / \mathrm{Ln}(\underline{x}_j, \underline{x}_j))^2$.

After finding the lowest value of α_j, we should approximate the inverse transformation function by piecewise linear approximation. An injective function for modelling piecewise linear function is proposed as following remark:

Remark 1 [13] An injective function $B : \{1, \dots, m\} \to \{0, 1\}$, $\theta = [m]$ where the vectors B and $B(p+1)$ differ in at most one component for all $p \in \{1, 2, \dots, m+1\}$, can be constracted, where $B(p) = (u_1, \dots, u_\theta)\ \forall\ uk \in \{0, 1\}$, $k = 1, \dots, \theta$ and $B(0) = B(1)$.

Some terms are introduced below: (FA TSAI 2011)

$$S^+(k) = \{p \mid \forall B(P) \text{ and } B(p+1),\, uk = 1,\, p = 1,\, ...,\, m-1\} \cup \{p | \forall\, B(p),\, uk = 1,\, p \in \{0,\, ...,\, m\}\}$$

$$S^-(k) = \{p \mid \forall B(P) \text{ and } B(p+1),\, uk = 0,\, p = 1,\, ...,\, m-1\} \cup \{p | \forall\, B(p),\, uk = 0,\, p \in \{0,\, ...,\, m\}\}$$

To approximate a univariate function with the proposed method by Vielma and Nemhauser, we have the following theorem:

Theorem 1 Given a univariate function $f(x)$, $a_0 \le x \le a_m$ denote $L(f(x))$ as a piece wise linear function of $f(x)$, where $a_0 < a_1 < ... < a_m$ are $m+1$ break point of $L(f(x))$, $L(f(x))$ can be denote as:

$$L(f(x)) = \sum_{p=0}^{m} f(ap)\,\lambda p \qquad x = \sum_{p=0}^{m} ap\lambda p \qquad \sum_{p=0}^{m} \lambda p = 1$$

$$\sum_{p \in s+(k)} \lambda p \le u_k \qquad \sum_{p \in s-(k)} \lambda p \le 1 - u_k \qquad \forall \lambda_p \in R_+,\ \ \forall u_k \in \{0,\, 1\} \tag{4}$$

Above theorem goes beyond other piecewise linearization formulation, including the Special Ordered Set type 2 (SOS2) model without binary variables.

3. Non-linear branch-and-bound algorithm

A node in the branch-and-bound tree is uniquely defined by a set of bounds, $(l,\, u)$, on the integer variables and corresponds to the Non Linear Programming (NLP) $(l,\, u)$

$$\min f(x)$$

$$s.t\ \ g(x) \le 0$$

$$x \in X,\ \ l_i \le x_i \le u_i \quad \forall i \in I \quad I \subseteq \{1, 2, \ldots, n\}$$

We note that the root node relaxation corresponds to NLP$(-\infty,\, \infty)$. Next, we describe the branching and pruning rules for branch-and-bound.

Branching: If the solution $\acute{x}$ of (NLP$(l,\, u)$) is feasible but not integer, then we branch on any non-integer variable, say $\acute{x_i}$. Branching introduces two new NLP nodes, also refers to as child nodes of (NLP $(l,\, u)$). In particular, we initialize bounds for two new problems as $(l^-,\, u^-) = (l,\, u)$ and $(l^+,\, u^+) = (l,\, u)$ and then modify the bound corresponding to the branching variable

$$u_i^- = \lfloor \acute{x}_i \rfloor \qquad l_i^+ = \lceil \acute{x_i} \rceil$$

The two new NLP problems are then defined as NLP(l^-, u^-) and NLP(l^+, u^+). In ractice, the new problems are stored on a heap H, which is updated with these two new problems.

Pruning rules: The pruning rules for NLP branch-and-bound are based on optimality and feasibility of NLP sub problems. We let U be an upper bound on the optimal value of non-linear programming problem. (Initialized as $U = \infty$).

• **Infeasible nodes.** If any node, (NLP (l, u)) is infeasible, then any problem in the sub-tree rooted at this node is also infeasible. Thus, we can prune infeasible nodes.

• **Integer feasible nodes.** If the solution, $x^{(l, u)}$ of (NLP(l, u)) is integral, then we obtain a new incumbent solution if $f(x^{(l, u)}) < U$, and we set $x^* x^{(l, u)}$ and $U = f(x^{(l, u)})$. Otherwise, we prune the node because its solution is dominated by the upper bound.

• **Upper bounds on NLP nodes.** If the optimal value of (NLP(l, u)), $f(x^{(l, u)})$ (or in fact a lower bound on the optimal value) is dominated by the upper bound, that is, if $f(x^{(l, u)}) \geq U$, then we can prune this node because there cannot be any better integer solution in the sub-tree rooted at (NLP(l, u)).

The complete nonlinear branch-and-bound algorithm is described in Algorithm 1.

Algorithm 1 Branch-and-bound for MINLP

```
Choose a tolerance $\varepsilon > 0$, set $U = \infty$, and initialize the heap of open problems $H = ;\ \phi$
Add (NLP$(-\infty, +\infty)$) to the heap: $H = H \cup$ NLP$(-\infty, +\infty)$.
**while** $H \neq \phi$ **do**
    Remove a problem (NLP$(l; u)$) from the heap: $H = H - $(NLP$(l; u)$)
    Solve (NLP$(l; u)$) and let its solution be $x^{(l, u)}$
    **if** (NLP$(l; u)$) is infeasible **then**
        Node can be pruned because it is infeasible.
    **else if** $f(x^{(l, u)}) > U$ **then**
        Node can be pruned, because it is dominated by upper bound.
    **else if** $x_I^{(l, u)}$ integral **then**
        Update incumbent solution: $x^* = x^{(l, u)}$, $u = f(x^{(l, u)})$
    **else**
        Branch On Variable $(x_i^{(l, u)}, l, u, H)$, see Algorithm 2.
    **end if**
**end while**
```

The following Proposition establishes the convergence of algorithm:

Proposition 3 For solving nonlinear branch-and-bound, we assume that the problem functions f and g are convex and twice continuously differentiable and that X is a bounded polyhedral set. Then it follows that branch-and-bound terminates at an optimal solution after searching a finite number of nodes or with an indication that the non-linear problem has no solution.

We note that the convergence analysis of nonlinear branch-and-bound requires only to ensure that every node that is pruned is solved to global optimality. The convexity assumptions are one convenient sufficient condition that ensures global optimality, but clearly not the only one!

Algorithm 2 Branch on a fractional variable

```
Subroutine: $S \leftarrow$ Branch On Variable $\left(x_i^{(l, u)}, l, u, H\right)$
// Branch on a fractional $x_i^{(l, u)}$, $i \in I$
Set $u_i^- = \lfloor x_i^{(l, u)} \rfloor$, $l^- = l$, $l_i^+ = \lfloor x_i^{(l, u)} \rfloor$, $u^+ = u$
Add NLP$(l^-, u^-)$ and $NLP(l^+, u^+)$ to the heap $H = H\{$NLP$(l^-, u^-)$, NLP$(l^+, u^+)\}$.
```

4. Proposed method

Linear branch and bound method for integrating geometric problems:

Step 1:

Solved the convex geometric problem without considering integer variables. There are two cases:

Case 1: If the variables get the integer values, the problem is solved.

Case 2: If the variables not integer, we go to the second step.

Step 2:

1. Let $Z_I = \infty$.

2. Branching: consider the variable x_I that is not integer and put $x^* = x_I$. We have $[x_I] < x_I < [x_I + 1]$. The problem is divided into two sub-problems p_1, p_2.

Problem p_1: The problem p_1 is the original problem that added the condition $x_I + t = [x_I]$ to the problem and solved the problem again.

Problem p_2: The problem p_2 is the original problem that added the condition $x_I = t + [x_I + 1]$ to the problem and solved the problem again.

3. We solve the problem p_1 and p_2 and replace Z_I with whichever one had a better answer.

4. Sub-branches have one of the following three states:

i. All variables have the integer value.

ii. The sub-problem is infeasible.

iii. The value of the function becomes worse than Z_I.

To present the proposed method.

Example 1 Lets consider a very applied physically example in which we consider the problem of designing a gravel box. We want to find a box with the integer dimensions that has the lowest cost of transportation. The problem is as follows:

$$\text{Min}\, 40{x_1}^{-1}{x_2}^{-1}{x_3}^{-1} + 40x_2x_3 + 20x_1x_3 + 10x_1x_2$$

After solving the following problem by dual method we have:

$$x_1^* = 2m \quad x_2^* = 1m \quad x_3^* = 0.5m$$

The volume of the box is 1 m^3, the number of trips = 400, and the minimum total cost = \$ 100.

Consider the objective function of the gravel box design problem, we want to solve this problem again so that the dimensions of the box are integers. This function is non-convex. Using the NPT method, we make the problem convex and then piecewise linear approximation.

$$\text{Min}\, 40{x_1}^{-1}{x_2}^{-1}{x_3}^{-1} + 40x_2x_3 + 20x_1x_3 + 10x_1x_2$$

$$1 \le x_1 \le 3, \;\; 1 \le x_2 \le 3, \;\; 0.2 \le x_3 \le 2$$

$$x_1, x_2 \in Z$$

The final solution is presented in Table 1, without considering the integer variables.

The convexity of the following problem with the assumption $\beta_1 = \beta_2 = \beta_3 = 0.25$ and the four break points of convexity of the following problem is:

$$\mathrm{Min}\,40x_1^{-1}x_2^{-1}x_3^{-1}+40y_2^{-\frac{1}{0.25}}y_3^{\frac{1}{0.25}}+20y_1^{-\frac{1}{0.25}}y_3^{-\frac{1}{0.25}}+10y_1^{-\frac{1}{0.25}}y_2^{-\frac{1}{0.25}}$$

$$s.t \quad y_i=l\left(x_i^{-\beta_i}\right)=\sum_{p=0}^{m}f\left(a_p\right)\lambda_p \quad i=1,2,3$$

$$x_i=\sum_{p=0}^{m}\left(a_p\right)\lambda_p \quad i=1,2,3$$

$$\sum_{p=0}^{m}\lambda_p=1 \qquad \sum_{p\varepsilon s^{+}(k)}\lambda_p\le u_k \qquad \sum_{p\varepsilon s^{-}(k)}\lambda_p\le 1-u_k$$

Table 1. The solutions of the problem without considering the integer variables by lingo

Variable	Value	Reduced cost
$U1$	1.000000	2.800357
$U2$	1.000000	5.524081
$U3$	1.000000	23.75966
$U4$	0.000000	−5.504592
$U5$	1.000000	78.77829
$U6$	0.000000	−26.25066
$X1$	1.844603	0.000000
$X2$	1.204137	0.000000
$X3$		
$Y2$	0.9606432	0.000000
$Y3$	1.274361	0.000000
$Y1$	0.8603850	0.000000
LANDA0	0.000000	0.000000
LANDA1	3107943	0.000000
LANDA2	0.6892057	0.000000
LANDA3	0.000000	0.000000
LANDA4	0.000000	1.052905
LANDA5	0.5917260	0.000000
LANDA6	0.4082740	0.000000
LANDA7	0.000000	0.000000
LANDA8	0.000000	0.000000
LANDA9	0.000000	0.000000
LANDA10	0.4503939	0.000000
LANDA11	0.5496061	0.000000
LANDA12	0.000000	0.000000
LANDA13	0.000000	20.50109
LANDA14	0.000000	0.000000

Local optimal solution for example 1 can be observed in Table 2.

Table 2. Local optimal solution

Local optimal solution found	
Objective value	91.00418
Objective bound	91.00418
Infeasibilities	0.2220446E-15
Extended solver steps	48
Total solver iterations	1,393

Now, after obtaining the integer variables and finding the integer solution to the problem with the nonlinear branch and bound algorithm, we have the following results, shown in Table 3.

Table 3. Final result after effecting the nonlinear branch and bound algorithm

Variable	Value	Reduced cost
$U1$	1.000000	2.921953
$U2$	1.000000	5.763946
$U3$	1.000000	0.000000
$U4$	0.000000	0.000000
$U5$	1.000000	80.50255
$U6$	0.000000	−26.82522
$X1$	2.000000	1.251907
$X2$	1.000000	−23.13104
$X3$	0.4949785	0.000000
$Y2$	1.000000	0.000000
$Y3$	1.258137	0.000000
$Y1$	0.8408964	0.000000
LANDA0	0.000000	0.000000
LANDA1	0.000000	0.000000
LANDA2	1.000000	0.000000
LANDA3	0.000000	0.000000
LANDA4	0.000000	1.098623
LANDA5	1.000000	0.000000
LANDA6	0.000000	1.690446
LANDA7	0.000000	0.000000
LANDA8	0.000000	0.000000
LANDA9	0.000000	0.000000
LANDA10	0.4100429	0.000000
LANDA11	0.5899571	0.000000
LANDA12	0.000000	0.000000
LANDA13	0.000000	0.000000
LANDA14	0.000000	0.000000

Local optimum solution after effecting the nonlinear branch and bound algorithm is given in Table 4.

Table 4. Local optimal solution

Local optimal solution found	
Objective value	92.33431
Objective bound	92.33431
Infeasibilities	0.1110223E-15
Extended solver steps	53
Total solver iterations	2,647

By applying the linear branch and bound algorithm after convexification and piecewise linear approximation, we have the gravel box design problem's result that is shown in Table 5.

Table 5. Gravel box design problem's result after convexification and piecewise linear approximation

Variable	Value	Reduced cost
$U1$	0.000000	-2.921953
$U2$	1.000000	8.685900
$U3$	1.000000	0.000000
$U4$	0.000000	0.000000
$U5$	1.000000	80.50255
$U6$	0.000000	-26.82522
$X1$	2.000000	0.000000
$X2$	1.000000	-18.53904
$X3$	0.4949785	0.000000
$Y2$	1.000000	0.000000
$Y3$	1.258137	0.000000
$Y1$	0.8408964	0.000000
LANDA0	0.000000	0.000000
LANDA1	0.000000	0.000000
LANDA2	1.000000	0.000000
LANDA3	0.000000	0.000000
LANDA4	0.000000	6.942530
LANDA5	1.000000	0.000000
LANDA6	0.000000	1.690446
LANDA7	0.000000	0.000000
LANDA8	0.000000	0.000000
LANDA9	0.000000	0.000000
LANDA10	0.4100429	0.000000
LANDA11	0.5899571	0.000000
LANDA12	0.000000	0.000000
LANDA13	0.000000	0.000000
LANDA14	0.000000	0.000000
$T1$	0.000000	0.000000
$T2$	0.000000	0.000000

Table 6 represents the Local optimal solution after convexification to the problem.

Table 6. Local optimal solution

Local optimal solution found	
Objective value	92.33431
Objective bound	92.33431
Infeasibilities	0.1110223E-15
Extended solver steps	18
Total solver iterations	1,491

By comparing the solutions with the proposed method, we have the following results that can be observed in Table 7.

Table 7. Results for solver iterations, solver steps & objective function value

Total solver iterations	Extended solver steps	Objective function	
-	-	100	Dual solution
592	5	100	The solution without convexification
1,393	48	91.00418	The solution of the problem after convexification and piecewise linear approximation before integration
2,647	53	92.32431	The solution to the problem after integration
1,491	18	92.32431	Proposed method

Example 2 Here we have taken gravel box design problem with minor modification from [14]. A total of 800 cubic-meters of gravel is to be ferried across a river on a barrage. A box (with an open top) is to be built for this purpose. The transport cost per round trip of barrage of box is Rs 0.05; the cost of materials of the ends of the box are Rs 20/m^2, and other two sides and bottom are made from available scrap materials. Find the dimension of the box that is to be built for this purpose to minimize the transport cost and material cost. Let length = x_1 m, width = x_2 m, height = x_3 m. The area of the ends of the gravel box = x_2x_3 m^2. Area of the sides = x_1x_3 m^2. Area of the bottom = x_1x_2 m^2. The volume of the gravel box = $x_1x_2x_3$ m^3. Transport cost: Rs $\frac{800}{x_1x_2x_3}$. Material cost: $40x_2x_3$. So the multi-objective geometric programming is

$$\min g_{01}(x_1, x_2, x_3) = \frac{40}{x_1x_2x_3} + 40x_2x_3$$

$$\min g_{02}(x_1, x_2, x_3) = \frac{800}{x_1x_2x_3}$$

$$s.t \quad x_1x_2 + 2x_1x_3 \leq 4$$

$$0.1 \leq x_1 \leq 2 \quad 1 \leq x_2 \leq 6 \quad 1 \leq x_3 \leq 3 \quad x_1, x_1, x_1 \in \mathbb{Z}$$

After convexification we have:

$$\min g_{01}\left(x_1, x_2, x_3\right) = 40x_1^{-1}x_2^{-1}x_3^{-1} + 40{x_2}^{-1/\beta_2}{x_3}^{-1/\beta_3}$$

$$\min g_{02}\left(x_1, x_2, x_3\right) = 800x_1^{-1}x_2^{-1}x_3^{-1}$$

$$s.t \quad 1/4{x_1}^{-\frac{1}{\beta_1}}{x_2}^{-\frac{1}{\beta_2}} + 1/2{x_1}^{-1/\beta_1}{x_3}^{-1/\beta_3} \leq 1$$

$$0.1 \leq x_1 \leq 2 \quad 1 \leq x_2 \leq 6 \quad 1 \leq x_3 \leq 3 \quad x_1, x_1, x_1 \in \mathbb{Z}$$

$$\beta_1 = 0.01, \quad \beta_2 = 0.01, \quad \beta_3 = 0.02$$

$$\min g_{01}\left(x_1, x_2, x_3\right) = 40x_1^{-1}x_2^{-1}x_3^{-1} + 40{y_2}^{-1/\beta_2}{y_3}^{-1/\beta_3}$$

$$\min g_{02}\left(x_1, x_2, x_3\right) = 800x_1^{-1}x_2^{-1}x_3^{-1}$$

$$s.t \quad 1/4{y_1}^{-\frac{1}{\beta_1}}{y_2}^{-\frac{1}{\beta_2}} + 1/2{y_1}^{-1/\beta_1}{y_3}^{-1/\beta_3} \leq 1$$

$$y_i = l\left(x_i^{-\beta_i}\right) = \sum_{p=0}^{m} f\left(a_p\right)\lambda_p \quad i = 1, 2, 3$$

$$x_i = \sum_{p=0}^{m} \left(a_p\right)\lambda_p \quad i = 1, 2, 3$$

$$\sum_{p=0}^{m} \lambda_p = 1 \qquad \sum_{p \varepsilon s^+(k)} \lambda_p \leq u_k \qquad \sum_{p \varepsilon s^-(k)} \lambda_p \leq 1 - u_k$$

$$0.1 \leq x_1 \leq 2 \quad 1 \leq x_2 \leq 6 \quad 1 \leq x_3 \leq 3 \quad x_1, x_2, x_3 \in \mathbb{Z}$$

$$\lambda_i \geq 0 \quad i = 1, 2, \ldots, 14 \quad u_k \in \{0.1\} \quad k = 1, 2, \ldots, 6$$

The final comparison result is given in Table 8.

Table 8. Final comparison results for example 2

	x_1	x_2	x_3	Objective function	Steps	Iteration
Nonlinear branch and bound method	1	2	1	130	10	3,147
Proposed method	1	2	1	130	3	1,031

5. Conclusion

We have presented an integrated method for solving mixed integer geometric programming problems by combining convexification, piecewise linearization, and a linear branch-and-bound algorithm. The proposed approach offers significant improvements in computational efficiency and solution quality over conventional techniques. Comparative analysis and numerical results validate its effectiveness. Future work will explore scaling the method for large-scale industrial problems and integrating machine learning to enhance approximation accuracy and branching decisions.

Conflict of interest

The authors are hereby declare that, they have no any financial or proprietary interests in any material discussed in the article entitled "Solving Mixed Integer Geometric Programming Problems". The authors also declare that no any conflict of interest exist.

References

[1] Shirin Nejad J, Saraj M, Shokrolahi Yancheshmeh S, Kiany Harchegani F. Global optimization of mixed integer signomial fractional programming problems. *Measurement and Control*. 2024; 57(8): 1211-1217.

[2] Zhang J, Liu C, Yan J, Li X, Zhen H-L, Yuan M. A survey for solving mixed integer programming via machine learning. *Neurocomputing*. 2023; 519: 205-217. Available from: https://doi.org/10.1016/j.neucom.2022.11.024.

[3] Melo W, Fampa M, Raupp F. Integrating non-linear branch-and-bound and outer approximation for convex mixed integer non-linear programming. *Journal of Global Optimization*. 2014; 60(2): 373-389. Available from: https://doi.org/10.1007/s10898-014-0217-8.

[4] Huang Z, Wang K, Liu F, Zhen H-L, Zhang W, Yuan M, et al. Learning to select cuts for efficient mixed integer programming. *Pattern Recognition*. 2022; 123: 108353. Available from: https://doi.org/10.1016/j.patcog.2021.108353.

[5] Che X, Liu J, Yin W. Learning to optimize: a tutorial for continuous and mixed integer optimization. *Science China Mathematics*. 2024; 67: 1191-1262. Available from: https://doi.org/10.1007/s11425-023-2293-3.

[6] Lundell A, Westerlund T. Solving global optimization problems using reformulations and signomial transformation. *Computers & Chemical Engineering*. 2018; 116: 122-134. Available from: https://doi.org/10.1016/j.compchemeng.2017.10.035.

[7] Lundell A, Skjäl A, Westerlund T. A reformulation framework for global optimization. *Journal of Global Optimization*. 2013; 57(1): 115-141. Available from: https://doi.org/10.1007/s10898-012-9877-4.

[8] Mansouri F. *Mining Temporal Patterns for Prediction: A Mixed Integer Programming Approach*. USA: Oregon State University; 2023.

[9] Tsai JF, Lin MH. An efficient global approach for posynomial geometric programming problems. *INFORMS Journal on Computing*. 2011; 23(3): 483-492. Available from: https://doi.org/10.1287/ijoc.1100.0403.

[10] Bazikar F, Saraj M. Solving linear multi-objective geometric problems via reference point approach. *Malaysian Science*. 2014; 43(8): 1271-1274.

[11] Hemmecke R, Koppe M, Lee J, Weismantel R. Non-linear integer programming. In: *50 Years of Integer Programming 1958-2008*. Berlin, Heidelberg: Springer; 2010. p.561-618.

[12] Belotti P, Kirches C, Leyffer S, Linderoth J, Luedtke J, Mahajan A. *Mixed Integer Nonlinear Programming*. UK: Cambridge University Press; 2013.

[13] Vielma JP, Nemhauser G. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming*. 2011; 128: 49-72. Available from: https://doi.org/10.1007/s10107-009-0295-4.

[14] Das P, Kumar Rot T. Multi-objective geometric programming and its application in gravel box problem. *Journal of Global Research in Computer Science*. 2014; 5(7): 6-11.