A multi-objective optimization approach for generalized linear multiplicative programming

Minh Hieu Nguyen[1,2] and Thanh Loan Nguyen[3]

[1] Unité de Mathématiques Appliquées, ENSTA, Institut Polytechnique de Paris, 91120 Palaiseau, France
[2] CEDRIC-Cnam, 292 rue Saint Martin, F-75141 Paris Cedex 03, France
[3] INP Clermont Auvergne, Univ Clermont Auvergne, Mines Saint-Etienne, CNRS, UMR 6158 LIMOS, 1 Rue de la Chebarde, 63178 Aubiere Cedex, France
Corresponding author: thanh_loan.nguyen@doctorant.uca.fr

Abstract. Multiplicative programming is a fundamental mathematical optimization problem in which the objective function contains a product of several real-valued functions. This paper deals with a class of multiplicative programming, called generalized linear multiplicative programming (GLMP), in which the objective is to minimize the product of two positive linear functions with general positive powers under linear constraints. Since the objective is a typical non-convex function, GLMP may have multiple local minima, making it computationally challenging. To address this, we propose a multi-objective optimization-based approach. By treating each function as an objective to be minimized, we show that a solution of GLMP is necessarily a non-dominated extreme point located on the vertex of the convex hull of the Pareto front. Then, we use a recursive algorithm to determine the set of all non-dominated extreme points. Notice that the solutions of GLMP can be directly extracted from this set. Furthermore, based on the Weighted Sum Method, it requires only solving one linear program in each iteration. Finally, we provide computational results on a specific instance of GLMP with 0-1 knapsack constraints, indicating that our approach is promising.

Keywords: Linear multiplicative programming · Multi-objective optimization · Pareto front · Weighted Sum Method.

1 Introduction

Multiplicative programming is a fundamental mathematical optimization problem that involves optimizing a product of several real-valued functions subject to general constraints. In this paper, we consider a class of multiplicative programming, called *generalized linear multiplicative programming* (GLMP), which can be formulated as:

$$\min_{x \in \mathcal{X}} P(x)Q(x)^{\rho}, \tag{1}$$

where $\mathcal{X} = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ is a finite set of feasible solutions, $\rho \in \mathbb{R}_+$ and $P(x), Q(x)$ are positive linear functions, $\forall x \in \mathcal{X}$.

GLMP has been widely explored in the literature due to its broad inclusion of various mathematical programming models, such as quadratic programming, linear multiplicative programming, and bilinear programming. Moreover, GLMP plays a crucial role in numerous scientific applications, including portfolio management [8], marketing and service planning [13], and microeconomics [14], among others.

From the algorithmic point of view, even a product of two linear functions (i.e., $\rho = 1$) is not necessarily convex [3], and then GLMP belongs to a class of nonconvex programming problems called global optimization problems, which may have multiple local minima. Thus, finding global solutions to GLMP is NP-hard [12]. In the past several decades, many methods have been developed to tackle GLMP. These methods can be mainly classified as parametric simplex method-based algorithms (e.g., [5]), branch-and-bound methods (e.g., [6]), and various heuristic methods (e.g., [9], [11]).

As an optimal solution of GLMP is an efficient (non-dominated or Pareto-optimal) solution of a multi-objective optimization problem where each function is to be minimized, some methods have been recently proposed to solve GLMP based on multi-objective optimization techniques. For example, when $\rho = 1$, Shao et al. employed primal and dual multi-objective linear programming algorithms to compute the Pareto front, which contains all non-dominated points to solve GLMP [12]. Similarly, Mahmoodian et al. developed a decision-space search algorithm that works directly in the space of decision variables. The algorithm performs similarly to the classical branch-and-bound algorithm for solving single-objective mixed integer linear programs [10].

In this paper, we further develop the multi-objective optimization approach by improving the method proposed in [12]. Specifically, we establish that a global solution of GLMP should be a non-dominated extreme point located on the vertex of the convex hull of the Pareto front. Instead of computing all non-dominated points by branch-and-bound methods, we determine directly all non-dominated extreme points by a recursive algorithm, significantly reducing computational complexity. Notice that in practice, the number of non-dominated extreme points is much less than the number of non-dominated points. Furthermore, based on the Weighted Sum Method, it requires only solving one linear program in each iteration. Finally, we present computational results on a GLMP instance with 0-1 knapsack constraints, demonstrating the effectiveness of our approach.

The paper is organized as follows. In Section 2, we present a mathematical formulation and some preliminaries on multi-objective optimization. In Section 3, we design a recursive algorithm to determine the global solution of GLMP. Computational results on an instance of GLMP with 0-1 knapsack constraints will be presented and discussed in Section 4. Section 5 concludes the paper and outlines directions for future research.

2 Generalized linear multiplicative programming and multi-objective optimization

We recall that an optimal solution to problem (1) is an efficient solution (i.e., non-dominated solution or Pareto-optimal solution) of the following bi-objective optimization problem: [12]

$$\min_{x\in\mathcal{X}} (P(x), Q(x)), \tag{2}$$

Consequently, we can obtain an optimal solution of GLMP (*ρ-LMP solution*) by minimizing $P(x)Q(x)^{\rho}$ over the set of efficient solutions (i.e., Pareto front) to problem (2). Let $(P, Q) = (P(x), Q(x))$ denote the objective values corresponding to a decision vector $x \in \mathcal{X}$. Let $\mathcal{S}$ represent the set of pairs (P, Q) corresponding to all feasible decision vector solutions. This paper will characterize the feasible solutions (including the efficient solutions) to problem (2) by using pairs (P, Q) instead of the decision vector solutions x. Thus, two feasible solutions having the same values of (P, Q) will be considered equivalent i.e., $(P', Q') \equiv (P'', Q'') \iff P' = P''$ and $Q' = Q''$. Furthermore, an efficient solution (P, Q) is also a point (P, Q) on the Pareto front.

In the following, we restate several classes of efficient solutions [15].

1. *Supported efficient solutions* are optimal solutions of a weighted sum single-objective problem

$$\mathcal{W}(\alpha) = \min_{(P,Q)\in\mathcal{S}} P + \alpha Q, \text{ where } \alpha \in \mathbb{R}_+,$$

2. *Non-supported efficient solutions* are efficient solutions that are not optimal solutions of $\mathcal{W}(\alpha)$ for any $\alpha \in \mathbb{R}_+$.

Furthermore, we can distinguish two classes of supported efficient solutions.

a. Supported solutions are located on the vertex of the convex hull of the Pareto front called *extremal supported efficient solutions*.

b. Supported solutions are located on the relative inferior faces of the convex hull of the Pareto front called *non-extremal supported efficient solutions*.

Fig. 1 illustrates an example of bi-objective optimization problem (2) with different classes of efficient solutions on the Pareto front.

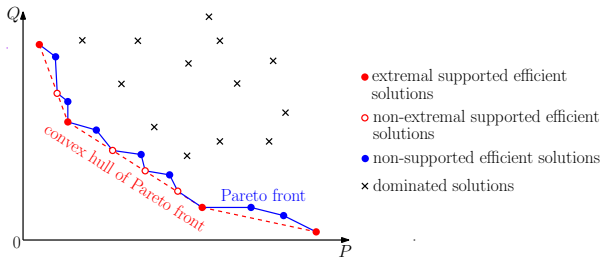


Fig. 1: Illustration of extremal/non-extremal supported efficient solutions, non-supported efficient solutions on the Pareto front.

3 Algorithm for solving GLMP

In this section, we first show that a ρ-LMP solution is a supported efficient solution. Furthermore, it is necessarily an extremal supported efficient solution.

Theorem 1. *Let (P_*, Q_*) be a ρ-LMP solution. Then it is a supported efficient solution. Moreover, it is necessarily an extremal supported efficient solution.*

Proof. Since (P_*, Q_*) is a ρ-LMP solution, we have $PQ^\rho \geq P_* Q_*^\rho, \forall (P,Q) \in \mathcal{S}$.

Using Young's inequality for products (See Appendix 1), we have

$$\frac{1}{\rho+1}\frac{P}{P_*} + \frac{\rho}{\rho+1}\frac{Q}{Q_*} \geq (\frac{P}{P_*})^{\frac{1}{\rho+1}}(\frac{Q}{Q_*})^{\frac{\rho}{\rho+1}} = \sqrt[\rho+1]{\frac{PQ^\rho}{P_* Q_*^\rho}} \geq 1, \ \forall (P,Q) \in \mathcal{S}, \quad (3)$$

Multiplying both sides of (3) by $(\rho+1)P_* > 0$ gives

$$P + \alpha_* Q \geq P_* + \alpha_* Q_*, \forall (P,Q) \in \mathcal{S},$$

where $\alpha_* = \rho P_*/Q_* > 0$. Thus, (P_*, Q_*) is an optimal solution of $\mathcal{W}(\alpha_*)$ which implies it is a supported efficient solution.

Now let (P_i, Q_i), (P_j, Q_j) be two extremal supported efficient solutions and (P_k, Q_k) be a supported efficient solution such that they are collinear and $P_i < P_k < P_j$, $Q_i > Q_k > Q_j$. We will show that $P_k Q_k^\rho > min\{P_i Q_i^\rho, P_j Q_j^\rho\}$.

Since they are collinear, there exist $\lambda_1, \lambda_2 \in \mathbb{R}_+$ and $\lambda_1 + \lambda_2 = 1$ such that $P_k = \lambda_1 P_i + \lambda_2 P_j$ and $Q_k = \lambda_1 Q_i + \lambda_2 Q_j$.

Using Young's inequality for products gives

$$P_k Q_k^\rho = (\lambda_1 P_i + \lambda_2 P_j)(\lambda_1 Q_i + \lambda_2 Q_j)^\rho \geq (P_i^{\lambda_1} P_j^{\lambda_2})(Q_i^{\lambda_1} Q_j^{\lambda_2})^\rho$$
$$= (P_i Q_i^\rho)^{\lambda_1}(P_j Q_j^\rho)^{\lambda_2} \geq \min\{P_i Q_i^\rho, P_j Q_j^\rho\}^{\lambda_1+\lambda_2} = \min\{P_i Q_i^\rho, P_j Q_j^\rho\},$$

Notice that the "=" sign in the above inequalities do not hold since $(P_i, Q_i) \not\equiv (P_j, Q_j)$. Consequently, we have $P_k Q_k^\rho > min\{P_i Q_i^\rho, P_j Q_j^\rho\}$. In other words, a ρ-LMP solution is necessarily an extremal supported efficient solution. □

Then we show the monotonic relationship between $\alpha \in \mathbb{R}_+$ and the optimal solution values of $\mathcal{W}(\alpha)$.

Lemma 1. *Given $0 \leq \alpha' < \alpha''$ and let (P', Q'), $(P'', Q'') \in \mathcal{S}$ be the solutions of $\mathcal{W}(\alpha')$ and $\mathcal{W}(\alpha'')$, respectively. Then $P' \leq P''$ and $Q' \geq Q''$.*

Proof. The optimality of (P', Q') and (P'', Q'') gives

$$P' + \alpha' Q' \leq P'' + \alpha' Q'', \text{ and} \quad (4a)$$
$$P'' + \alpha'' Q'' \leq P' + \alpha'' Q' \quad (4b)$$

Adding (4a) and (4b) gives $(\alpha' - \alpha'')(Q' - Q'') \leq 0$. Since $\alpha' < \alpha''$, $Q' \geq Q''$. In addition, the inequality (4a) implies $P' - P'' \leq \alpha'(Q'' - Q') \leq 0$. □

As a consequence of Lemma 1, we state the following corollary.

Corollary 1. *If $\alpha' < \alpha''$ and (P', Q') is the solution of both $\mathcal{W}(\alpha')$ and $\mathcal{W}(\alpha'')$ then (P', Q') is the unique optimal solution of $\mathcal{W}(\alpha)$ for all $\alpha' < \alpha < \alpha''$.*

Definition 1 (Consecutive extremal supported efficient solutions). *Let (P_i, Q_i) and (P_j, Q_j) be two extremal supported efficient solutions such that $P_i < P_j$ and $Q_i > Q_j$. They are considered consecutive supported efficient solutions if there is no other extremal supported efficient solution located between (P_i, Q_i) and (P_j, Q_j) on the convex hull of the Pareto front. In other words, there is no other extremal supported efficient solution (P, Q) such that $P_i < P < P_j$ and $Q_i > Q > Q_j$.*

The following lemma gives us a condition for verifying whether two extremal supported efficient solutions are consecutive.

Theorem 2. *Let (P_i, Q_i) and (P_j, Q_j) be two extremal supported efficient solutions such that $P_i < P_j$ and $Q_i > Q_j$. Then (P_i, Q_i) and (P_j, Q_j) are two consecutive extremal supported efficient solutions on the convex hull of the Pareto front if and only if (P_i, Q_i) is an optimal solution of $\mathcal{W}(\alpha_k)$ where $\alpha_k = (P_j - P_i)/(Q_i - Q_j)$.*

Proof. Suppose that (P_i, Q_i) and (P_j, Q_j) are optimal solutions of $\mathcal{W}(\alpha_i)$ and $\mathcal{W}(\alpha_j)$, respectively, where $0 < \alpha_i < \alpha_j$. Let $\alpha_k = (P_j - P_i)/(Q_i - Q_j)$. We first show that $\alpha_i \leq \alpha_k \leq \alpha_j$.

The optimality of (P_i, Q_i) and (P_i, Q_i) gives

$$P_i + \alpha_i Q_i \leq P_j + \alpha_i Q_j,$$
$$P_j + \alpha_j Q_j \leq P_j + \alpha_j Q_j,$$

Thus, we obtain $\alpha_i \leq \frac{P_j - P_i}{Q_i - Q_j} \leq \alpha_j$, which implies $\alpha_i \leq \alpha_k \leq \alpha_j$.

$\Longrightarrow$ Assume that (P_i, Q_i) and (P_j, Q_j) are two consecutive extremal supported efficient solutions. We consider the following three cases.

1. If $\alpha_k = \alpha_i$ then obviously (P_i, Q_i) is an optimal solution of $\mathcal{W}(\alpha_k)$.

2. If $\alpha_k = \alpha_j$, we have $\alpha_j = (P_j - P_i)/(Q_i - Q_j)$ which implies

$$P_i + \alpha_j Q_i = P_j + \alpha_j Q_j,$$

Hence, (P_i, Q_i) is also an optimal solution of $\mathcal{W}(\alpha_k)$.

3. If $\alpha_i < \alpha_k < \alpha_j$, let (P_k, Q_k) be a supported efficient solution which is an optimal solution of $\mathcal{W}(\alpha_k)$.

If $(P_k, Q_k) \equiv (P_i, Q_i)$ then obviously (P_i, Q_i) is an optimal solution of $\mathcal{W}(\alpha_k)$.

If $(P_k, Q_k) \equiv (P_j, Q_j)$ then $\alpha_k = (P_k - P_i)/(Q_i - Q_k)$ which implies

$$P_i + \alpha_k Q_i = P_k + \alpha_k Q_k,$$

Hence, (P_i, Q_i) is also an optimal solution of $\mathcal{W}(\alpha_k)$.

If $(P_k, Q_k) \not\equiv (P_i, Q_i)$ and $(P_k, Q_k) \not\equiv (P_j, Q_j)$ then $P_i < P_k < P_j$ and $Q_i > Q_k > Q_j$. Since (P_i, Q_i) and (P_j, Q_j) are two consecutive extremal supported efficient solutions, (P_k, Q_k) is necessarily collinear with (P_i, Q_i) and (P_j, Q_j). Thus, we have

$$\frac{P_k - P_i}{Q_i - Q_k} = \frac{P_j - P_i}{Q_i - Q_j} = \alpha_k,$$

Hence, $P_i + \alpha_k Q_i = P_k + \alpha_k Q_k$ which implies (P_i, Q_i) is an optimal solution of $\mathcal{W}(\alpha_k)$.

$\Longleftarrow$ Assume that (P_i, Q_i) is an optimal solution of $\mathcal{W}(\alpha_k)$. Notice that a supported solution located between (P_i, Q_i) and (P_j, Q_j) is necessarily an optimal solution of $\mathcal{W}(\alpha)$ where $\alpha_i < \alpha < \alpha_j$ due to Lemma 1. We consider the following three cases.

1. If $\alpha_k = \alpha_i$, we have $P_i + \alpha_i Q_i = P_j + \alpha_i Q_j$ which implies (P_j, Q_j) is also an optimal solution of $\mathcal{W}(\alpha_i)$.

Since (P_j, Q_j) is the optimal solution of both $\mathcal{W}(\alpha_i)$ and $\mathcal{W}(\alpha_j)$, it is the unique optimal solution of $\mathcal{W}(\alpha)$ for any $\alpha_i < \alpha < \alpha_j$. Thus, (P_i, Q_i) and (P_j, Q_j) are two consecutive extremal supported efficient solutions.

2. If $\alpha_k = \alpha_j$, we have $P_i + \alpha_j Q_i = P_j + \alpha_j Q_j$ which implies (P_i, Q_i) is also an optimal solution of $\mathcal{W}(\alpha_j)$. Similar to the case above, (P_i, Q_i) is the unique optimal solution of $\mathcal{W}(\alpha)$ for any $\alpha_i < \alpha < \alpha_j$. Hence, (P_i, Q_i) and (P_j, Q_j) are two consecutive extremal supported efficient solutions.

3. If $\alpha_i < \alpha_k < \alpha_j$, we have $P_i + \alpha_k Q_i = P_j + \alpha_k Q_j$ due to $\alpha_k = (P_j - P_i)/(Q_i - Q_j)$. Furthermore, as (P_i, Q_i) is an optimal solution of $\mathcal{W}(\alpha_k)$ which implies (P_j, Q_j) is also an optimal solution of $\mathcal{W}(\alpha_k)$.

Since (P_i, Q_i) is an optimal solution of both $\mathcal{W}(\alpha_i)$ and $\mathcal{W}(\alpha_k)$, it is the unique optimal solution of $\mathcal{W}(\alpha)$ where $\alpha_i < \alpha < \alpha_k$. Similarly, (P_j, Q_j) is the unique optimal solution of $\mathcal{W}(\alpha)$ where $\alpha_k < \alpha < \alpha_j$. Furthermore, for any optimal solution (P_k, Q_k) of $\mathcal{W}(\alpha_k)$ which is different to (P_i, Q_i), we have $P_i + \alpha_k = P_k + \alpha_k Q_k$ which implies

$$\alpha_k = \frac{P_k - P_i}{Q_i - Q_k} \implies \frac{P_j - P_i}{Q_i - Q_j} = \frac{P_k - P_i}{Q_i - Q_k},$$

Thus, (P_i, Q_i), (P_k, Q_k) and (P_j, Q_j) are collinear which implies (P_k, Q_k) is not an extremal supported efficient solution. Consequently, (P_i, Q_i) and (P_j, Q_j) are consecutive extremal supported efficient solutions. □

We now present Procedure FIND_GLMP(), an iterative algorithm for identifying the solution of the GLMP and all extremal supported efficient solutions of its corresponding bi-objective optimization problem. The algorithm begins by computing two extremal supported efficient solutions, (P_0, Q_0) and (P_1, Q_1), which are obtained by minimizing $P(x)$ and $Q(x)$, respectively (or by lexicographically minimizing in case of multiple optimal solutions). The algorithm initializes a solution set E for the GLMP, starting with the list of extremal supported efficient solutions $I \leftarrow \{(P_0, Q_0), (P_1, Q_1)\}$. Additionally, a search list $L = \{[(P_0, Q_0), (P_1, Q_1)]\}$ is initialized to store the areas to be explored.

Procedure 1 Finding ρ-LMP solutions

Input: A general GLMP.
Output: ρ-LMP solutions.

procedure *FIND_GLMP()*
  minimize $P(x), Q(x)$ to obtain $(P_0, Q_0), (P_1, Q_1)$, respectively
  $I \leftarrow \{(P_0, Q_0), (P_1, Q_1)\}$, $L \leftarrow \{[(P_0, Q_0), (P_1, Q_1)]\}$
  while $L \neq \emptyset$ **do**
    select and remove $[(P_i, Q_i), (P_j, Q_j)]$ from L
    $\alpha \leftarrow \frac{P_j - P_i}{Q_i - Q_j}$
    if (P_i, Q_i) is an optimal solution of $\mathcal{W}(\alpha)$ **then**
      $I \leftarrow I \cup \{(P_i, Q_i), (P_j, Q_j)\}$
      continue to the next pair in L
    else
      solve $\mathcal{W}(\alpha)$ to obtain a solution (P_m, Q_m)
      $L \leftarrow L \cup \{[(P_i, Q_i), (P_m, Q_m)], [(P_m, Q_m), (P_j, Q_j)]\}$
    end if
  end while
  $E \leftarrow \text{argmin}_{(P,Q)\in I}\, PQ^{\rho}$
  return E
end procedure

The algorithm iterates through the search list L, refining the set of solutions as follows. For each pair of solutions (P_i, Q_i) and (P_j, Q_j) in L, it calculates a weight

$$\alpha = \frac{P_j - P_i}{Q_i - Q_j}$$

and considers the problem $\mathcal{W}(\alpha)$. According to Theorem 2, we first verify whether (P_i, Q_i) is an optimal solution of $\mathcal{W}(\alpha)$. If it is, then (P_i, Q_i) and (P_j, Q_j) are consecutive extremal supported efficient solutions, meaning no further refinement is needed. In this case, we add them to the extremal supported efficient solution list I and proceed to the next pair in L. Otherwise, we solve $\mathcal{W}(\alpha)$ to obtain a new solution (P_m, Q_m), ensuring that $(P_m, Q_m) \not\equiv (P_i, Q_i)$ and $(P_m, Q_m) \not\equiv (P_j, Q_j)$. The algorithm then updates L by replacing the interval $[(P_i, Q_i), (P_j, Q_j)]$ with two new search intervals:

$$[(P_i, Q_i), (P_m, Q_m)] \quad \text{and} \quad [(P_m, Q_m), (P_j, Q_j)].$$

This process continues iteratively until L becomes empty, ensuring that all extremal supported efficient solutions have been identified in I. Finally, E is defined as the set of optimal solutions that minimize PQ^{ρ} among all extremal supported efficient solutions (P, Q). As stated in Theorem 1, E represents the solution set of the GLMP with respect to the value of ρ.

4 Experimental study on the GLMP with 0-1 knapsack constraints

4.1 Definition and modeling

The 0-1 knapsack problem is a fundamental combinatorial optimization problem in which a decision-maker selects a subset of n items, each defined by a profit value, a weight, and a binary decision variable, while ensuring that the total weight does not exceed a given capacity. In this study, we examine the GLMP with 0-1 knapsack constraints, where the objective is to minimize a multiplicative function while adhering to a knapsack-type feasibility constraint. This problem extends the classical knapsack problem, which has wide-ranging applications in resource allocation, logistics, and financial decision-making [2].

Kuno's work [7] makes a significant contribution to solving multiplicative programming problems with 0-1 knapsack constraints by developing a branch-and-bound algorithm. However, this approach is specifically designed for a restricted class of problems and does not generalize to broader instances of multiplicative programming with knapsack constraints. On the other hand, our work introduces a more generalized framework that extends beyond the specific problem class tackled by Kuno, allowing it to efficiently handle a wide range of multiplicative optimization problems without relying on specialized structural assumptions or restrictive conditions.

Formally, given a set of n items, each item i is associated with two objective values, p_i and q_i (e.g., representing different profit measures or utility functions, a weight w_i, and a binary decision variable x_i, where $x_i = 1$ if the item is selected and $x_i = 0$ otherwise. Let $\mathcal{X}$ denote the set of all feasible selections. The knapsack has a total capacity C, which represents a lower bound on the total weight of selected items. The GLMP with 0-1 knapsack constraints is then formulated as:

$$
\begin{aligned}
\min \quad & P(x)Q(x)^{\rho} \\
\text{subject to} \quad & P(x) = \sum_{i=1}^{n} p_i x_i, \\
& Q(x) = \sum_{i=1}^{n} q_i x_i, \\
& \sum_{i=1}^{n} w_i x_i \geq C, \\
& x_i \in \{0,1\}, \quad i = 1, \ldots, n.
\end{aligned}
$$

This formulation introduces multiplicative interactions between selected items, making the problem significantly more complex than its additive counterpart. As discussed in Section 3, solving the GLMP 0-1 knapsack problem requires solving $\mathcal{W}(\alpha) = \min_{x\in\mathcal{X}} P(x) + \alpha Q(x) \iff \min_{x\in\mathcal{X}} \sum_{i=1}^{n}(p_i + \alpha q_i)x_i$. This transformation effectively reduces the original problem to a single-objective 0-1

knapsack problem with a unique profit $p = p_i + \alpha q_i$. In the following sections, we present experimental instances and analyze the results obtained using our proposed method for solving the GLMP 0-1 knapsack problem.

4.2 Numerical results

We generated problem instances with the number of items n ranging from 20 to 500, increasing by 20 each time. For each instance, item weights w_i were randomly selected from the range $[1, 15]$. The two profit values, p_i and q_i, were independently drawn from the intervals $[20, 40]$ and $[10, 50]$, respectively. For a single-objective 0-1 knapsack problem, the ratio

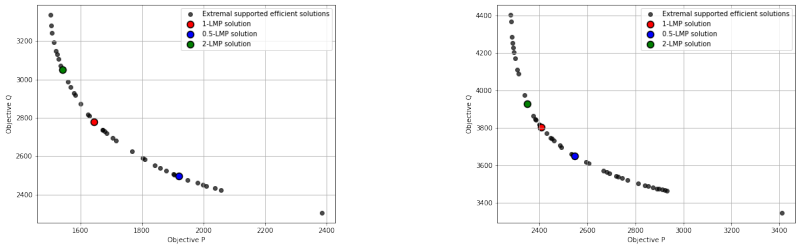
$$r = \frac{C}{\sum_{i=1}^{n} w_i}$$

indicates how constrained the knapsack capacity is relative to the total item weights. When $r \approx 0.5$, the problem reaches a critical point where many feasible combinations exist, but the minimal capacity enforces strict selection constraints, intensifying competition among them and making the optimization particularly challenging. In this study, we focus on $r = 0.5$, as it is known to be the hardest setting for both single-objective and multi-objective knapsack problems [1]. Notice that dynamic programming and branch-and-bound algorithms work well for small 0-1 knapsack problems but become inefficient as the problem size grows. Dynamic programming requires too much memory and computation, while branch-and-bound struggles with a rapidly expanding search space. To address these issues, we employ the CPLEX solver to solve the single-objective 0-1 knapsack problem $\mathcal{W}(\alpha)$. CPLEX is specifically designed to efficiently handle large-scale instances, delivering optimal solutions in a reasonable computational time.

Table 1 presents the numerical results for computing the ρ-LMP solution (P, Q) of the GLMP 0-1 knapsack problem for three different values of ρ: 0.5, 1, and 2. The column n_{items} denotes the number of items considered, and the "ESES" column indicates the number of extremal supported efficient solutions. "Time" and "Iterations" refer to the computational time in seconds and the number of iterations required by Algorithm 1 to obtain all extremal supported efficient solutions for each instance, respectively. Additionally, Fig. 2 presents the ρ-LMP solution for the GLMP 0-1 knapsack constraints, along with all extremal supported efficient solutions of the corresponding bi-objective optimization problem. The results are shown for instances with 300 and 400 items, considering three cases of ρ, where $\rho \in \{0.5, 1, 2\}$. All computational experiments were performed on a system featuring an Intel Core i5-10500 CPU (3.10 GHz), 15 GB RAM with 6 cores and 12 threads.

Table 1: Numerical results for GLMP 0-1 knapsack problem

$n_{\textbf{items}}$	**Time**	**ESES**	**Iterations**	$\rho = 0.5$		$\rho = 1$		$\rho = 2$	
				P	Q	P	Q	P	Q
20	0.2965	7	11	175	195	202	149	268	117
40	0.3115	8	13	392	345	448	280	503	256
60	0.5993	15	27	600	345	677	288	677	288
80	0.5495	10	17	520	689	731	388	751	382
100	0.7708	18	33	900	669	1039	550	1140	517
120	0.8115	19	35	956	818	1051	702	1116	672
140	0.8470	19	35	1328	832	1369	798	1447	772
160	2.2811	48	93	1504	1049	1791	824	1893	791
180	1.2663	27	51	1632	1293	1773	1150	1943	1069
200	1.1329	23	43	1768	1185	1869	1099	2009	1046
220	2.0099	44	85	1949	1372	2146	1188	2342	1112
240	0.9087	18	33	1694	2084	2407	1138	2486	1112
260	1.1313	26	49	2371	1502	2550	1375	2707	1318
280	0.9467	25	47	2551	1681	2678	1569	2811	1513
300	1.8231	38	73	2495	1920	2780	1645	3050	1542
320	3.5265	75	147	2842	2215	3143	1924	3366	1824
340	1.1231	29	55	3319	1927	3552	1743	3756	1673
360	2.7074	53	103	3308	2261	3514	2070	3758	1967
380	3.0421	53	103	3424	2454	3593	2295	3943	2134
400	2.0309	45	87	3649	2548	3804	2408	3930	2351
420	1.7259	37	71	4002	2329	4203	2166	4465	2076
440	1.8541	41	79	4086	2777	4443	2503	4709	2388
460	2.1846	43	83	3330	3811	4378	2361	4745	2227
480	3.3043	57	111	4530	2849	4909	2543	5170	2446
500	4.7188	95	187	4952	3133	5360	2805	5816	2638



(a) The case of 300 items (b) The case of 400 items

Fig. 2: ρ-LMP solutions with $\rho \in \{0.5, 1, 2\}$

As shown in Table 1, our algorithm demonstrates high computational efficiency, consistently solving large-scale instances in a very short runtime. Although the GLMP 0-1 knapsack problem can theoretically have multiple solutions, all tested cases with three values of ρ returned a unique optimal solution. According to 2, the algorithm terminates once all pairs of consecutive extremal supported efficient solutions have been identified, making the number of iterations approximately twice the number of extremal supported efficient solutions. As a result, the computational time is more dependent on the number of extremal supported efficient solutions than on the total number of items. This explains why instances with fewer items can sometimes take longer to solve. In Table 1, the 320-item instance with 75 extremal supported efficient solutions takes 3.5265 seconds, while the 400-item instance, with only 45 solutions, requires just 2.0309 seconds. While larger instances generally tend to have more extremal supported efficient solutions, this is not always the case. Since the algorithm's runtime is closely tied to the number of these solutions, problem complexity is influenced not only by the input size but also by the structure of the Pareto front.

Furthermore, the results show how different values of ρ influence the corresponding ρ-LMP solutions. For a fixed number of items, the values of P in optimal solutions follows the order $P_{\rho=0.5} < P_{\rho=1} < P_{\rho=2}$, while those of Q follow the opposite trend: $Q_{\rho=2} < Q_{\rho=1} < Q_{\rho=0.5}$. This suggests that in the obtained optimal solutions, a larger value of ρ corresponds to a higher value of P and a lower value of Q.

5 Conclusion

In this paper, we developed a multi-objective optimization approach to solve *generalized linear multiplicative programming* (GLMP) where the objective is a product of two positive linear functions with general positive powers under linear constraints. By considering a bi-objective optimization problem, we first demonstrated that a solution of GLMP is necessarily an extremal supported efficient solution on the Pareto front. Then, we presented a recursive algorithm to determine all the extremal supported efficient solutions. Based on the Weighted Sum Method, it requires only solving one linear program in each iteration. Finally, computational experiments on some instances of GLMP with 0-1 knapsack constraints have shown the effectiveness of our approach.

Future work should enhance the proposed algorithm to solve GLMP more efficiently by avoiding the need to identify all extremal supported efficient solutions. Additionally, we aim to explore extensions of our results to multiplicative programming involving the product of more than two positive linear functions.

Acknowledgments

This work has been carried out at the Energy4Climate Interdisciplinary Center (E4C) of IP Paris and Ecole des Ponts ParisTech, which is in part supported by the 3rd Programme d'Investissements d'Avenir [ANR-18-EUR-0006-02].

References

1. Visée, M., Teghem, J., Pirlot, M., Ulungu, E.L.: Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. In: Journal of Global Optimization, 12, pp. 139–155, 1998.
2. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, 2004.
3. Avriel, M., Diewert W.E., Schaible S., Zhang I.: Generalized concavity. Plenum Press, New York, 2010.
4. Bennett, K.F., Mangasarian, O.L.: Bilinear separation of two sets in n-space. In: Comput Optim Appl, 2, pp. 207–227, 1994.
5. Kuno, T., Konno, H.: Linear multiplicative programming. In: Mathematic Programming, 56, pp. 51-64, 1992.
6. Kuno, T.: A finite branch-and-bound algorithm for linear multiplicative programming. In: Comput Optim Appl, 20, pp. 119–135, 2001.
7. Kuno, T.: Solving a class of multiplicative programs with 0-1 knapsack constraints. In: *Journal of Optimization Theory and Applications*. Vol. 103, No. 1. New York: Plenum Publishing Corporation, pp. 121–135, 1999.
8. Konno, H., Inori, M.: Bond portfolio optimization by bilinear fractional programming. In: Journal of the Operations Research Society of Japan, Vol. 32, No. 2, 1989.
9. Liu, X.J., Umegaki, T., Yamamoto, Y.: Heuristic methods for linear multiplicative programming. In: Journal of Global Optimization, 4(15), pp. 433-447, 1999.
10. Mahmoodian, M., Charkhgard, H., Zhang, Y.: Multi-objective optimization-based algorithms for solving mixed integer linear minimum multiplicative programs. In: Computers & Operations Research, Vol. 128, 2021.
11. Benson HP, Boger GM.: Multiplicative programming problems: analysis and efficient point search heuristic. In: Journal of Optimization Theory and Application, 94(2), pp. 487–510, 1997.
12. Shao, L., Ehrgott, M.: Primal and dual multi-objective linear programming algorithms for linear multiplicative programmes. In: A Journal of Mathematical Programming and Operations Research, Vol. 65(2), 2016.
13. Samadi, F., Mirzazadeh, A., Pedram, M.M.: Fuzzy pricing, marketing and service planning in a fuzzy inventory model: a geometric programming approach. In: Appl Math Model 37, pp. 6683–6694, 2013.
14. Henderson, J.M., Quandt, R.E.: Microeconomic theory. In: McGraw-Hill, New York, 1961.
15. Alves, M.J., Costa, J.P.: Graphical exploration of the weight space in three-objective mixed integer linear programs. In: European Journal of Operational Research, Vol. 248, pp. 72-83, 2016.

APPENDIX

Appendix 1

[Young's inequality for products] *Given* $x, y > 0, \alpha, \beta > 0$ *and* $\alpha + \beta = 1$. *Then* $\alpha x + \beta y \geq x^{\alpha} y^{\beta}$.

Proof. Since the natural logarithmic function is concave, using Jensen's inequality, we have

$$log(\alpha x + \beta y) \geq \alpha log(x) + \beta log(y) = log(x^{\alpha} y^{\beta}),$$

Thus, $\alpha x + \beta y \geq x^{\alpha} y^{\beta}$ and the equality holds if and only if $x = y$. □