A Criterion Space Search Feasibility Pump Heuristic for Solving Maximum Multiplicative Programs

Ashim Khanal, Hadi Charkhgard

Department of Industrial and Management Systems Engineering, University of South Florida, Tampa, FL, USA

Abstract

We study a class of nonlinear optimization problems with diverse practical applications, particularly in cooperative game theory. These problems are referred to as Maximum Multiplicative Programs (MMPs), and can be conceived as instances of “Optimization Over the Frontier” in multi-objective optimization. To solve MMPs, we introduce a feasibility pump-based heuristic that is specifically designed to search the criterion space of their multi-objective optimization counterparts. Through a computational study, we show the efficacy of the proposed method.

Keywords: Multiplicative program; Multi-objective optimization; Criterion space; Feasibility pump; Nash social welfare

1. Introduction

A Multi-Objective Mixed Integer Linear Program (MOMILP) can be stated as

$$\max\left\{f_1(\boldsymbol{x}),\dots,f_p(\boldsymbol{x}):\ \boldsymbol{x}\in\mathcal{X}\right\},\tag{1}$$

where $\mathcal{X}$ denotes the set of feasible solutions that is assumed to be bounded [10]. It is also assumed that $\mathcal{X}$ is characterized by only linear constraints and it can involve both integer and continuous decision variables. Additionally, in MOMILPs, $f_i(\boldsymbol{x})$ is a linear function for all $i\in\{1,\dots,p\}$. Since objectives are often conflicting in MOMILPs, there is often no solution that can optimize all objectives at the same. In such cases, computing and/or selecting a *Pareto-optimal* solution that can desirably balance the conflicting objectives is a non-trivial task; A Pareto-optimal solution is a solution that is impossible to improve the value of one of its objectives without making the values of other objectives worse. One classical approach for finding a desirable Pareto-optimal solution in the literature of multi-objective optimization is known as *optimization over the frontier* which seeks to optimize a separate objective function over the entire Pareto-optimal frontier for computing a desirable Pareto-optimal solution. This study focuses on an interesting class of optimization over the frontier problems known as Maximum Multiplicative Programs (MMPs). Given a reference point $\boldsymbol{r}\in\mathbb{R}^p$ in the criterion space, a MMP can be stated as

$$\max\left\{\prod_{i=1}^{p} y_i(\boldsymbol{x}):\ \boldsymbol{x}\in\mathcal{X},\ \boldsymbol{y}(\boldsymbol{x})\geq\boldsymbol{0}\right\},\tag{2}$$

where $y_i(\boldsymbol{x}) := f_i(\boldsymbol{x})-r_i$ and $r_i\in\mathbb{R}$ is parameter for all $i\in\{1,\dots,p\}$. As we show in Section 2, solving a MMP to optimality guarantees to return a Pareto-optimal solution. The underlying logic behind MMPs is that a desirable solution to a MOMILP is a Pareto-optimal solution $\boldsymbol{x}$ that its

Email address: `hcharkhgard@usf.edu` (Hadi Charkhgard)

image in the criterion space, i.e., $\boldsymbol{y}(\boldsymbol{x}) := \big(y_1(\boldsymbol{x}), \dots, y_p(\boldsymbol{x})\big)$, and the reference point $\boldsymbol{r} \in \mathbb{R}^p$ creates a box with the maximum hypervolume in the criterion space. In other words, the multiplicative objective function of a MMP basically measures the hypervolume for any feasible solution from the reference point in the criterion space.

MMPs are notably connected to Cooperative Game Theory, specifically in the realm of bargaining games. In these games, independent players aim to form a grand coalition that maximizes their individual payoffs [28]. The agreement among all players becomes crucial in determining a fair distribution of payoffs within the grand coalition. Addressing this challenge, Nash proposed the use of MMPs, also referred to as *Nash Social Welfare Optimization*, as a means to find a fair solution for cooperative bargaining games [20, 21]. By leveraging MMPs, players can seek an equitable outcome that considers the interests of all involved parties in the cooperative game. This approach brings a sense of fairness and promotes cooperative decision-making in situations where multiple players strive to maximize their collective gains.

The application of MMPs in game theory extends beyond bargaining games to encompass other domains. MMPs play a significant role in computing market equilibrium in linear Fisher markets or Kelly capacity allocation markets [7, 8, 11, 19, 30]. Moreover, MMPs find utility beyond game theory and are applicable in diverse fields such as conservation planning, system reliability, and maximum likelihood estimation problems [2, 8, 9, 12]. Conservation planning, for instance, involves the selection of protected sites within a specific geographic region to preserve biodiversity [22]. This problem can be formulated by employing binary decision variables and incorporating constraints, such as budget limitations, connectivity requirements, and compactness considerations [17, 29]. MMPs are commonly employed in biodiversity preservation, where $y_i(\boldsymbol{x})$ represents the survival probability of species $i \in \{1, \dots, p\}$ for a feasible solution $\boldsymbol{x} \in \mathcal{X}$ [6, 22, 31]. In the realm of maximum likelihood estimation, MMPs are shown to be useful for problems involving nested logit models, where binary decision variables represent option selection [13, 18]. Furthermore, for system reliability, MMPs can be utilized to maximize the dependability of series-parallel systems, where binary decision variables represent individual subsystems [9].

The existing literature on solving MMPs primarily focuses on exact solution methods. Numerous precise techniques have been developed for solving different subclasses of MMPs. One popular strategy for MMPs involving no integer variables (referred to as Linear MMPs) is to log-transform the objective function and solve the modified problem using a convex programming solver, such as IBM ILOG CPLEX, Gurobi, or FICO Xpress [16, 8]. Some authors have developed iterative linear-programming based solution methods for solving Linear MMPs [30, 8]. Alternatively, converting the problem into a Second-Order Cone Program (SOCP) using the approach proposed by Ben-Tal and Nemirovski [3] and solving it with a commercial solver is also an effective method for solving Linear MMPs [8]. For MMPs involving integer decision variables, converting the problems into mixed integer SOCPs and solving them with commercial solvers such as IBM ILOG CPLEX has shown to be an effective approach [24, 26]. Additionally several effective *criterion-space search methods* are also developed in recent years for solving MMPs with integer decision variables. Criterion space search methods are the method that search the objective space of Problem (1) to find an optimal solution of a MMP.

Although current exact solution methods have demonstrated success, they are limited in their ability to tackle large-scale MMPs. As a result, the literature lacks accessible and generic heuristic solution approaches that can produce high-quality solutions for MMPs. Consequently, the main contribution of this study is to present an efficient heuristic approach that effectively addresses this gap. Our approach customizes the well-known feasibility pump for this purpose [1, 4, 5, 14, 15]. The feasibility pump has emerged as a successful heuristic solution approach for tackling single-objective mixed integer linear programs. In recent years, customized variations of the feasibility pump heuristic have proven effective in generating the complete Pareto-optimal frontier for multi-objective mixed integer linear programs (MOMILPs) [23]. However, no prior attempts have been made to utilize this approach for optimization over the frontier in the context of multi-objective optimization. Therefore, this study is the first attempt (to the best of our knowledge) in leveraging the feasibility pump-based approach for a specific class of optimization over the frontier problems, i.e., MMPs. Our method operates within the criterion space and employs a cut-generating technique tailored to this space, enabling the generation of high-quality solutions. Through a computational study with 270 large randomly generated instances, we show that our approach can quickly generate near optimal solutions, i.e., about 1.5% optimality gap.

The remainder of this paper is organized as follows. Section 2 offers problem description. Section 3 provides a detailed description of the proposed heuristic. Section 4 offers a computational study. Finally, Section 5 presents some concluding remarks.

2. Problem Description

A MMP can be stated as

$$\begin{aligned} \max\ & \prod_{i=1}^{p} y_i \\ \text{s.t.}\ & \boldsymbol{y} = D\boldsymbol{x} + \boldsymbol{d} \\ & A\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x}, \boldsymbol{y} \geq \boldsymbol{0}, \quad \boldsymbol{x} \in \mathbb{B}^{n_b} \times \mathbb{R}^{n_c}, \quad \boldsymbol{y} \in \mathbb{R}^{p}, \end{aligned} \tag{3}$$

where n_b and n_c represents the number of decision variables in binary and continuous forms, respectively. In this study, vectors are shown using bold fonts. Since MMP is assumed to be bounded, without loss of generality, we can assume that any general integer variable is transformed into a set of binary decision variables using standard transformation techniques. We also make a pragmatic assumption regarding the optimal objective value of Problem (3) stating that it is strictly positive, i.e., there exists a feasible solution with $\boldsymbol{y} > \boldsymbol{0}$. We employ the notation D to indicate a $p \times n$ matrix where $n := n_c + n_b$. Furthermore, the vector $\boldsymbol{d}$ is of p-dimension and A is an $m \times n$ matrix, while b is an m-dimensional vector. For the sake of notation simplicity, we partition the variable index set $\mathcal{N} := \{1, 2, ..., n\}$ into binary $\mathcal{N}_b$ and continuous $\mathcal{N}_c$ variable index sets.

As mentioned in the Introduction, an effective method to solve an MMP is to transform it into a mixed integer SOCP problem using the technique proposed by Ben-Tal and Nemirovski [3], and then solving the transformed problem using a commercial solver such as CPLEX. The underlying idea behind the transformation comes from the observation that an equivalent problem to the MMP can be constructed by adding a new non-negative variable, denoted by γ, and a constraint known as the "geometric-mean constraint" to the problem as follows

$$\max\left\{\gamma :\ 0 \leq \gamma \leq \left(\prod_{i=1}^{p} y_i\right)^{\frac{1}{p}},\ \boldsymbol{y} \in \mathcal{Y}\right\}.$$

Upon solving the equivalent problem to optimality, the objective value of the solution raised to the power of p, i.e., $\bar{\gamma}^p$, represents the optimal objective value for the corresponding MMP. It is important to note that the geometric mean constraint mentioned earlier may not necessarily be a second-order cone constraint. However, any constraint in the form of $\{u, v, w \geq 0 : u \leq \sqrt{vw}\}$ is a second-order cone constraint because it is equivalent to $\{u, v, w \geq 0 : \sqrt{u^2 + (\frac{v-w}{2})^2} \leq \frac{v+w}{2}\}$. To convert the geometric-mean constraint formulation into a mixed integer SOCP, Ben-Tal and Nemirovski [3] demonstrate a straightforward method involving the introduction of additional sets of constraints and variables. Specifically, if k is the smallest integer value such that $2^k \geq p$, the resulting equivalent mixed integer SOCP is

$$\begin{aligned}
&\max \gamma \\
&\text{s.t. } 0 \leq \gamma \leq \sqrt{\tau_1^{k-1}\tau_2^{k-1}} \\
&\quad 0 \leq \tau_j^l \leq \sqrt{\tau_{2j-1}^{l-1}\tau_{2j}^{l-1}} && \text{for } j = 1, \ldots, 2^{k-l} \text{ and } l = 1, \ldots, k-1 \\
&\quad 0 \leq \tau_j^0 = y_j && \text{for } j = 1, \ldots, p \\
&\quad 0 \leq \tau_j^0 = \gamma && \text{for } j = p+1, \ldots, 2^k \\
&\quad \boldsymbol{y} \in \mathcal{Y}.
\end{aligned}$$

In this study, we utilize the above-mentioned mixed-integer SOCP reformulation to compute bounds and assess the quality of our proposed multi-objective optimization-based heuristic. Considering our approach operates in the criterion space, we define two sets:

$$\mathcal{X} := \{\boldsymbol{x} \in \mathbb{B}^{n_b} \times \mathbb{R}^{n_c} : A\boldsymbol{x} \leq \boldsymbol{b},\ \boldsymbol{x} \geq \boldsymbol{0}\}$$

and

$$\mathcal{Y} := \{\boldsymbol{y} \in \mathbb{R}^p : \exists\ \boldsymbol{x} \in \mathcal{X},\ \boldsymbol{y} = D\boldsymbol{x} + \boldsymbol{d},\ \boldsymbol{y} \geq \boldsymbol{0}\}.$$

These sets represent the feasible set in the decision and criterion spaces, respectively. In this context, $\boldsymbol{x} \in \mathcal{X}$ is referred to as a *feasible solution*, and $\boldsymbol{y} \in \mathcal{Y}$ is referred to as a *feasible point*, where $\boldsymbol{y}$ represents the image of $\boldsymbol{x}$ in the criterion space. Next, we present a formal definition and a proposition that are beneficial for comprehending our proposed methodology.

Definition 1. *A feasible solution $\boldsymbol{x} \in \mathcal{X}$ is called Pareto-optimal, if there is no other $\boldsymbol{x}' \in \mathcal{X}$ such that*

$$\begin{aligned}
&y_i \leq y_i' && \forall i \in \{1, 2, ..., p\} \\
&y_i < y_i' && \textit{for at least one } i \in \{1, 2, ..., p\},
\end{aligned}$$

where $\boldsymbol{y} := D\boldsymbol{x} + \boldsymbol{d}$ and $\boldsymbol{y}' := D\boldsymbol{x}' + \boldsymbol{d}$. If $\boldsymbol{x}$ is Pareto-optimal solution, then $\boldsymbol{y}$ is called a Pareto-optimal point.

Proposition 1. *An optimal solution of Problem (3), denoted by $\boldsymbol{x}^*$, is a Pareto-optimal solution and therefore its corresponding image in the criterion space, denoted by $\boldsymbol{y}^*$ where $\boldsymbol{y}^* := D\boldsymbol{x}^* + \boldsymbol{d}$, is a Pareto-optimal point.*

Proof. Let us assume that $\boldsymbol{x}^*$ is an optimal solution of Problem (3) but is not a Pareto-optimal solution. According to Definition 1, this implies the existence of a feasible solution denoted by $\boldsymbol{x} \in \mathcal{X}$ that dominates $\boldsymbol{x}^*$. In other words, the following conditions hold:

$$\begin{aligned} &y_i^* \le y_i && \forall i \in \{1, 2, \dots, p\} \\ &y_i^* < y_i && \text{for at least one } i \in \{1, 2, \dots, p\} \end{aligned}$$

where $\boldsymbol{y} := D\boldsymbol{x} + \boldsymbol{d}$. Additionally, based on the assumptions of Problem (3), we know that $\boldsymbol{y}^* > \boldsymbol{0}$. Therefore, it follows that $0 < \prod_{i=1}^{p} y_i^* < \prod_{i=1}^{p} y_i$. As a result, we can conclude that $\boldsymbol{x}^*$ cannot be an optimal solution, leading to a contradiction. □

The significance of Proposition 1 lies in its indication that the search for an optimal point in a MMP can be narrowed down to the set of Pareto-optimal points rather than considering all feasible points. This realization transforms a MMP into a special class of optimization over the frontier. We exploit this property to develop a generic criterion-space for solving MMPs.

Algorithm 1: Criterion Space Search Feasibility Pump Heuristic

```
Input: A feasible instance of Problem (3) represented by $\mathcal{Y}$ in the criterion space
1  $(\boldsymbol{x}^*, \boldsymbol{y}^*) \leftarrow (-, \boldsymbol{0})$
2  SearchDone← False
3  while $time$ < TimeLimit & SearchDone=False do
4      $(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}}) \leftarrow$ FeasibilityPump $\big(R(\mathcal{Y})\big)$
5      if $\widetilde{\boldsymbol{x}}$ is not integer feasible or $\widetilde{\boldsymbol{y}} \ngtr \boldsymbol{0}$ then
6          SearchDone← True
7      else
8          $\mathcal{Y} \leftarrow \mathcal{Y} \cap \Big\{\boldsymbol{y} \in \mathbb{R}^p : \sum_{i=1}^{p} \frac{y_i}{\widetilde{y}_i} > p\Big\}$
9          if $\prod_{i=1}^{p} \widetilde{y}_i > \prod_{i=1}^{p} y_i^*$ then
10             $(\boldsymbol{x}^*, \boldsymbol{y}^*) \leftarrow (\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}})$
11 return $(\boldsymbol{x}^*, \boldsymbol{y}^*)$
```

3. Proposed Algorithm

To simplify our discussion in this section, we use the notation $R(\mathcal{Y})$ to represent the set of all relaxed feasible solutions and their corresponding images in the criterion space. More specifically, we define $R(\mathcal{Y})$ as follows:

$$R(\mathcal{Y}) := \big\{(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{x} \in \mathcal{X}^R,\ \boldsymbol{y} \in \mathcal{Y},\ \boldsymbol{y} = D\boldsymbol{x} + \boldsymbol{d}\big\}$$

Here, $\mathcal{X}^R$ represents the LP-relaxation of $\mathcal{X}$, where binary variables are relaxed and allowed to take values between 0 and 1. Our proposed approach, outlined in Algorithm 1, aims to generate a high-quality solution for the MMP problem. The algorithm receives a feasible MMP and iteratively applies two main operations. The first operation, a special case of the classical feasibility pump (Line 4), finds feasible solutions, while the second operation, a criterion space-based cut-generating mechanism (Line 8), enhances solution quality. Throughout the algorithm, the best solution and point found, denoted as $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, are continuously updated if a better feasible solution and point in terms of the multiplicative objective function are discovered (Lines 9-10). Upon termination, the algorithm reports $(\boldsymbol{x}^*, \boldsymbol{y}^*)$. The termination condition occurs either when the time limit is reached (Line 3) or when the feasibility pump operation fails to find an integer feasible solution with a strictly positive image in the criterion space (Lines 5-6). If $\widetilde{\boldsymbol{x}} \notin \mathbb{B}^{n_b} \times \mathbb{R}^{n_c}$, the feasibility pump operation has failed to generate an integer feasible solution, making it unlikely to find any other solutions using this approach. Similarly, if $\widetilde{\boldsymbol{y}} \not> \boldsymbol{0}$, the feasibility pump operation has failed to generate a strictly positive feasible point, rendering the cut-generating mechanism unusable. Next, we explain both feasibility pump and criterion space-based cut-generating operations in more details.

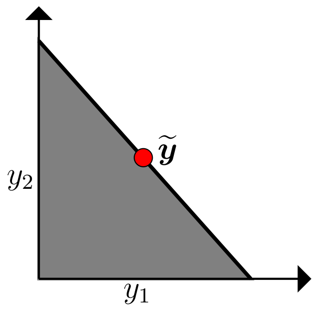

Figure 1: $\widetilde{y}$ maximizes $y_1 \times y_2$ among all points in the triangle

3.1. Cut-generating Operation

We begin by introducing the cut-generating operation, which draws inspiration from the research conducted by Charkhgard et al. [8]. The fundamental concept behind the cut stems from the observation that, given any strictly positive feasible point $\widetilde{\boldsymbol{y}} \in \mathcal{Y}$, it is feasible to create a right-angled triangle in the criterion space. This triangle's corner points consist of the origin and one point on each axis. Within this triangle, $\widetilde{\boldsymbol{y}}$ attains the maximum value for the multiplicative objective function among all points present. Let's consider Figure 1 for illustration purposes, assuming $p = 2$ and $\widetilde{\boldsymbol{y}}$ as a strictly positive feasible point. The assertion is that we can generate the depicted triangle in the figure, which maximizes $y_1 \times y_2$ among all points within the triangle. To construct such a triangle, it suffices to determine its hypotenuse. According to Proposition 2 explained subsequently, when $p = 2$, the equation for the hypotenuse can be shown as $\sum_{i=1}^{2} \frac{y_i}{\widetilde{y}_i} = 2$. Consequently, the right-angled triangle is defined as $\{\boldsymbol{y} \in \mathbb{R}^2 : \boldsymbol{y} \geq \boldsymbol{0},\ \sum_{i=1}^{p} \frac{y_i}{\widetilde{y}_i}\}$.

Proposition 2. *Let $\widetilde{\boldsymbol{y}}$ be a strictly positive point. It is the unique optimal point of*

$$\max\Big\{\prod_{i=1}^{p} y_i :\ \boldsymbol{y} \geq \boldsymbol{0},\ \sum_{i=1}^{p} \frac{y_i}{\widetilde{y}_i} \leq p\Big\}.$$

PROOF. Observe that

$$\max\Big\{\prod_{i=1}^{p} y_i :\ \boldsymbol{y} \geq \boldsymbol{0},\ \sum_{i=1}^{p} \frac{y_i}{\widetilde{y}_i} \leq p\Big\} \equiv \max\Big\{\sum_{i=1}^{p} \log(y_i) :\ \boldsymbol{y} \geq \boldsymbol{0},\ \sum_{i=1}^{p} \frac{y_i}{\widetilde{y}_i} \leq p\Big\}$$

$$\equiv \min\left\{ -\sum_{i=1}^{p} \log(y_i) : \boldsymbol{y} \geq \boldsymbol{0},\ \sum_{i=1}^{p} \frac{y_i}{\widetilde{y}_i} \leq p \right\}$$

The equivalent optimization problem is a convex optimization problem. We relax the problem by dropping the non-negativity constraint:

$$\min\left\{ -\sum_{i=1}^{p} \log(y_i) : \sum_{i=1}^{p} \frac{y_i}{\widetilde{y}_i} - p \leq 0 \right\}.$$

The KKT optimality conditions of the relaxed problem are as follows,

$$\begin{aligned} \frac{1}{y_i} - \frac{\lambda}{\widetilde{y}_i} &= 0 && \forall i \in \{1,\ldots,p\} \\ \lambda\Big(\sum_{i=1}^{p} \frac{y_i}{\widetilde{y}_i} - p\Big) &= 0 \\ \sum_{i=1}^{p} \frac{y_i}{\widetilde{y}_i} - p &\leq 0 \\ \lambda &\geq 0 \end{aligned}$$

where λ is the dual variable associated with constraint $\sum_{i=1}^{p} \frac{y_i}{\widetilde{y}_i} - p \leq 0$. It is evident that the KKT conditions are satisfied by setting $\boldsymbol{y} = \widetilde{\boldsymbol{y}}$ and $\lambda = 1$. This implies that the strictly positive $\widetilde{\boldsymbol{y}}$ is optimal for both relaxed and not-relaxed problems. $\square$

Proposition 2 is generic and holds for any arbitrary value of p. Specifically, it states that when working with right-angled triangles in higher dimensions, the hypotenuse can be stated as $\sum_{i=1}^{p} \frac{y_i}{\widetilde{y}_i} = p$. Thus, if we aim to discover a feasible point with a higher value for the multiplicative objective function compared to the one corresponding to $\widetilde{\boldsymbol{y}}$, denoted as $\sum_{i=1}^{p} \widetilde{y}_i$, it is necessary to search beyond the boundaries of the triangle. In simpler terms, the condition Thus, if we aim to discover a feasible point with a higher value for the multiplicative objective function compared to the one corresponding to $\widetilde{\boldsymbol{y}}$, denoted as $\sum_{i=1}^{p} \widetilde{y}_i$, it is necessary to search beyond the boundaries of the triangle. In simpler terms, the condition

$$\sum_{i=1}^{p} \frac{y_i}{\widetilde{y}_i} > p$$

needs to be included as a cut in the criterion space. This precise step is carried out in Line 8 of Algorithm 1.

3.2. Feasibility Pump Operation

Our proposed feasibility pump operation resembles the classical approach used for single-objective integer linear programs. If the problem is feasible, the method operates on two solutions: $\widetilde{\boldsymbol{x}}$ and $\widetilde{\boldsymbol{x}}^I$. The former is a feasible solution for the LP-relaxation, $\widetilde{\boldsymbol{x}} \in \mathcal{X}^R$, that may not be an integer solution, i.e., $\widetilde{\boldsymbol{x}}^I \notin \mathbb{B}^{n_b} \times \mathbb{R}^n_c$, while the latter is an integer solution, i.e., $\widetilde{\boldsymbol{x}}^I \in \mathbb{B}^{n_b} \times \mathbb{R}^n_c$, that

Algorithm 2: FeasibilityPump $\big(R(\mathcal{Y})\big)$

```
1  $(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}}) = \arg\max\big\{\sum_{i=1}^{p} y_i : (\boldsymbol{x}, \boldsymbol{y}) \in R(\mathcal{Y})\big\}$
2  if $\widetilde{\boldsymbol{x}} = null$ then
3      Terminate $\leftarrow True$
4  else
5      Terminate $\leftarrow False$
6      $\widetilde{\boldsymbol{x}}^I \leftarrow$ Round($\widetilde{\boldsymbol{x}}$)
7  while $Terminate = False$ & $time <$ SearchTimeLimit do
8      if $\widetilde{x}_j \in \mathbb{B}$ for all $j \in \mathcal{N}_b$ then
9          Terminate $\leftarrow True$
10     else
11         $(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}}) \leftarrow \arg\min\big\{\sum_{\{j\in\mathcal{N}_b:\ \widetilde{x}^I_j=0\}} x_j + \sum_{\{j\in\mathcal{N}_b:\ \widetilde{x}^I_j=1\}}(1-x_j) : (\boldsymbol{x}, \boldsymbol{y}) \in R(\mathcal{Y})\big\}$
12         if $\exists\ j \in \mathcal{N}_b$ such that Round($\widetilde{x}_j$) $\neq \widetilde{x}^I_j$ then
13             $\widetilde{\boldsymbol{x}}^I \leftarrow$ Round($\widetilde{\boldsymbol{x}}$)
14         else
15             $\widetilde{\boldsymbol{x}}^I \leftarrow$ Flip($\widetilde{\boldsymbol{x}}^I$)
16 return $(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}})$
```

may not be feasible for the LP-relaxation, i.e., $\widetilde{\boldsymbol{x}}^I \notin \mathcal{X}^R$. Through iterative updates, the feasibility pumping method aims to minimize the distance between $\widetilde{\boldsymbol{x}}$ and $\widetilde{\boldsymbol{x}}^I$, ultimately seeking an integer feasible solution. Algorithm 2 shows the details of the proposed feasibility pump approach.

The algorithm initiates by solving $\max\big\{\sum_{i=1}^{p} y_i : (\boldsymbol{x}, \boldsymbol{y}) \in R(\mathcal{Y})\big\}$, aiming to obtain the initial solution $\widetilde{\boldsymbol{x}}$ and its corresponding criterion space representation $\widetilde{\boldsymbol{y}}$ (Line 1). This optimization problem exhibits linearity and can be efficiently solved, as it has a relaxed feasible set compared to the MMP and employs a summation objective function instead of multiplication. Notably, based on the principles of multi-objective optimization, the summation objective function always yields a Pareto-optimal solution for any given feasible set, which aligns with the nature of the multiplicative objective function as well. Moreover, a beneficial relationship exists between any MMP and its counterpart in summation-objective optimization, enabling the computation of a dual/upper bound for the corresponding MMP. For instance, $\big(\frac{\sum_{i=1}^{p}\widetilde{y}_i}{p}\big)^p$ serves as a dual/upper bound for its corresponding MMP, i.e., $\max\big\{\prod_{i=1}^{p} y_i : (\boldsymbol{x}, \boldsymbol{y}) \in R(\mathcal{Y})\big\}$. It is noteworthy that $\big(\frac{\sum_{i=1}^{p}\widetilde{y}_i}{p}\big)^p$ holds intuitive meaning in Cooperative Game Theory, signifying that in an ideal scenario, each player's share/utility should precisely match the average of the maximum attainable total benefits. This claim can be proven using the well-known "inequality of arithmetic and geometric means" [25].

In the proposed feasibility pump approach, the algorithm first checks if it is possible to find a possibly fractional solution $(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}})$ by solving the summation optimization problem. If the problem is infeasible and returns 'null', the search is immediately terminated (Lines 2-3). Otherwise, the algorithm proceeds by rounding $\widetilde{\boldsymbol{x}}$ to obtain an initial integer but possibly infeasible solution $\widetilde{\boldsymbol{x}}^I$ (Lines 4-6). The rounding operation is only applied to binary variables and not continuous variables. Next, the algorithm iteratively minimizes the distance between $\widetilde{\boldsymbol{x}}$ and $\widetilde{\boldsymbol{x}}^I$ in order to approach an integer feasible solution (Lines 7-15). The algorithm terminates if an integer feasible solution is found, indicated by $\widetilde{x}_j \in \mathbb{B}$ for all $j \in \mathcal{N}_b$ (Lines 8-9), or if the time limit is reached (Line 7). During each iteration, a new $\widetilde{\boldsymbol{x}}$ and its corresponding image $\widetilde{\boldsymbol{y}}$ are computed by solving an optimization problem that seeks to minimize the sum of variables that differ from the current $\widetilde{\boldsymbol{x}}^I$ in terms of binary values. This is achieved by solving the following optimization problem (Line 11):

$$\min\Big\{ \sum_{\{j\in\mathcal{N}_b:\ \widetilde{x}^I_j=0\}} x_j + \sum_{\{j\in\mathcal{N}_b:\ \widetilde{x}^I_j=1\}} (1-x_j) : (\boldsymbol{x},\boldsymbol{y}) \in R(\mathcal{Y})\Big\}.$$

If rounding the new $\widetilde{\boldsymbol{x}}$ results in a solution different from the current $\widetilde{\boldsymbol{x}}^I$, rounding is applied to generate a new $\widetilde{\boldsymbol{x}}^I$ (Lines 12-13). Otherwise, some components of the current $\widetilde{\boldsymbol{x}}^I$ are flipped according to the classical flipping procedure described in Fischetti et al. [14]. A random selection is made between $0.5v$ and $1.5v$ binary variables, and their values are flipped in $\widetilde{\boldsymbol{x}}^I$, i.e., if the selected component's value is zero, it is changed to one, and vice versa (Lines 12-13). It is important to highlight that the parameter v is defined by the user. In this research, we fine-tuned and established its default value as 10, aligning with the value employed in Fischetti et al. [14]. Similarly, the parameter SearchTimeLimit, also user-defined, underwent tuning in our computational analysis, leading to its configuration as $\frac{\log(m+n)}{4}$.

4. Computational study

In this section, we conduct a computational study using IBM CPLEX 22.1. Note that CPLEX is used both in our proposed algorithm when solving linear programs and also for solving mixed-integer SOCP reformulation of MMPs for showing the quality of the solutions found by our proposed algorithm. We use Python for implementing our proposed algorithm and running all experiments in this section. All our computational experiments are conducted on a Dell PowerEdge R360 with two Intel Xeon E5-2650 2.2 GHz 12-Core Processors (30MB), 128GB RAM, the RedHat Enterprise Linux 7.0 operating system, and using a single thread. Also, a time limit of 1200 seconds is imposed for solving each instance for all algorithms. All our instances and our Python source codes can be found in `https://github.com/Ashim-Khanal/MIMMPs`.

In this computational study, a total of 270 instances are employed. To generate challenging instances, we adopt a similar procedure as detailed in existing literature (e.g., [27]). We fix the values of $n_c = 0.2n$ and $n_b = 0.8n$. The instances are categorized into three main classes, distinguished by their p values from the set $\{3, 4, 5\}$. Each of these classes encompasses nine subclasses, which further differentiate instances based on the dimensions of the matrix $A_{m\times n}$. Within each subclass, we generate 10 instances at random. Our parameter settings involve considering different values of n from the set $\{2000, 3000, 4000\}$, where m is defined as $m = \alpha n$, with α taking on values from the set $\{2, 2.5, 3\}$. As a result, our smallest subclass corresponds to 2000×4000, with $n = 2000$ x-variables and $m = 4000$ constraints tied to x-variables. Conversely, our largest subclass corresponds to 4000×12000, with $n = 4000$ x-variables and $m = 12000$ constraints associated with x-variables.

We establish a sparsity level of 50% for matrix A, denoted as $s_A := 0.5$. Entries of matrix A corresponding to 'binary' variables are randomly selected from the discrete uniform distribution $[1, 30]$, while entries associated with 'continuous' variables are drawn from the discrete uniform distribution $[-10, 30]$. For vector $\boldsymbol{b}$, its components are chosen randomly from the discrete uniform distribution $[ns_A, 10ns_A]$, where ns_A signifies the anticipated count of non-zero elements in each row of matrix A. Matrix D is also characterized by a sparsity of 50%, with entries in row $i \in \{1, \dots, p\}$

being randomly sampled from the discrete uniform distribution $[-10i, 10i]$. It is worth noting that each row corresponds to a linear function that defines a y-variable.

To guarantee the complexity of the instances, we introduce conflict into the y-variables. Specifically, when deciding to assign a non-zero value to an entry located at row $i \in \{1, \ldots, p\}$ and column $j \in \mathcal{N}$ of matrix D, we analyze the historical distribution of positive and negative values in column j across previously generated rows of D. If the count of positives (negatives) surpasses the count of negatives (positives), we ensure that the entry's value in row $i \in \{1, \ldots, p\}$ and column $j \in \mathcal{N}$ becomes negative (positive). In instances of a tie, no limitations on the sign are imposed. Lastly, to ensure non-negativity in each objective function, we stipulate that the elements of vector $\boldsymbol{d}$ lie within the range $[|L_i|, 100|L_i| + 10]$, where L_i is the sum of absolute values of negative coefficients in row i. We randomly draw components for $\boldsymbol{d}$ from the mentioned discrete uniform distribution.

Table 1: Performance comparison for Class $p = 3$

Subclass	**SOCP-Solver**		**Proposed Approach**			
			Without Cuts		**With Cuts**	
($m \times n$)	Time (sec.)	Gap (%)	Time (sec.)	Gap (%)	Time (sec.)	Gap (%)
4,000 × 2,000	398.8	0	4.5	5.1	14.2	0.8
5,000 × 2,000	756.2	0	5.6	6.2	19.1	1.2
6,000 × 2,000	839.4	0	23.6	4.0	41.9	1.7
6,000 × 3,000	825.0	0	9.8	6.0	32.8	1.6
7,500 × 3,000	1,095.1	40	11.8	5.0	35.5	1.5
9,000 × 3,000	1,200	100	113.5	4.2	186.4	1.4
8,000 × 4,000	1,200	90	18.2	7.0	61.4	1.6
10,000 × 4,000	1,200	100	22.3	6.3	71.0	1.5
12,000 × 4,000	1,200	100	206.4	5.2	279.5	2.9

Table 1-3 presents an in-depth performance comparison between our proposed method and the SOCP-Solver. The values in each row of these tables represent averages across 10 instances. Note that when the SOCP-Solver successfully solves an instance within the specified time limit, it provides an optimal solution and its optimal objective value. If not, it is capable of reporting an upper bound (i.e., dual bound) for the optimal objective value in addition to the best solution that it has found. In either scenario, the SOCP-Solver aids us in calculating the optimality gap for our proposed approach. Consequently, we can leverage the SOCP-Solver's findings to demonstrate the efficacy of our proposed method. To underscore the significance of the cuts in our approach, we present the outcomes of our method under two distinct settings: 'With Cuts' and 'Without Cuts'. The 'With Cuts' configuration corresponds to the complete implementation of Algorithm 1. On the other hand, the 'Without Cuts' configuration involves executing only one iteration of the main loop (Lines 3-10) within Algorithm 1.

The results in the tables reveal that the performance of the SOCP-Solver is primarily influenced by the number of decision variables, i.e., n. Notably, the parameter p has a minor impact on its performance. When dealing with a small number of decision variables, the SOCP-Solver effectively produces optimal solutions within the imposed time limit. However, as the number of decision variables increases, the SOCP-Solver's performance experiences a steep decline. It not only fails to yield optimal solutions within the time limit but also struggles to generate solutions of high quality, with an optimality gap approaching 100%. On the contrary, our proposed heuristic approach swiftly produces nearly optimal solutions, accomplishing this within a fraction of the allocated time limit. Comparatively, the 'Without Cuts' setting exhibits a speed improvement of up to four times, but at the expense of solution quality, which can be up to six times worse than those obtained under the 'With Cuts' setting. This contrast underscores the pivotal role that the generation of high-quality

Table 2: Performance comparison for Class $p = 4$

Subclass ($m \times n$)	**SOCP-Solver**		**Proposed Approach**			
			Without Cuts		**With Cuts**	
	Time (sec.)	Gap (%)	Time (sec.)	Gap (%)	Time (sec.)	Gap (%)
4,000 × 2,000	442.1	0	5.4	4.8	16	1.6
5,000 × 2,000	472.8	0	6.7	4.2	19.1	1.1
6,000 × 2,000	922.8	0	8.0	4.0	23.4	0.8
6,000 × 3,000	657.2	0	10.5	5.2	40.5	0.9
7,500 × 3,000	999.6	10	13.7	5.3	48	1.2
9,000 × 3,000	1,087.8	80	16.4	3.5	44.3	0.7
8,000 × 4,000	1,150.2	90	19.5	4.5	66.5	0.7
10,000 × 4,000	1,200	100	25	5.5	84.2	1.3
12,000 × 4,000	1,200	100	30.7	3.9	100.8	0.7

solutions through cuts plays in our approach. In a broader context, our observations underscore the capability of the proposed heuristic approach to consistently generate solutions with an average optimality gap of 1.5%, even for the largest values of n and p.

Table 3: Performance comparison for Class $p = 5$

Subclass ($m \times n$)	**SOCP-Solver**		**Proposed Approach**			
			Without Cuts		**With Cuts**	
	Time (sec.)	Gap (%)	Time (sec.)	Gap (%)	Time (sec.)	Gap (%)
4,000 × 2,000	408.2	0	5.5	4.3	16.6	1.3
5,000 × 2,000	682.6	0	6.7	4.3	19.9	1.4
6,000 × 2,000	865.9	0	8.0	3.3	22.0	0.7
6,000 × 3,000	857.2	0	11.2	3.8	35.2	0.6
7,500 × 3,000	1,160.1	40	14.2	4.5	44.0	1.1
9,000 × 3,000	1,200	100	17.5	4.0	35.2	1.0
8,000 × 4,000	1,118.5	80	20.3	3.9	57.0	1.0
10,000 × 4,000	1,200	100	26.9	5.2	80.2	1.8
12,000 × 4,000	1,200	100	32.5	3.7	83.7	0.8

5. Final remarks

In this study, we proposed the first custom-built feasibility pump-based heuristic approach for solving MMPs. A notable attribute of our proposed method lies in its generality which makes it applicable for solving various MMPs as long as mathematical formulations are provided by users. By transforming the non-linear objective function of the MMP into a linear form, our approach applies the problem-solving process within the criterion space of its multi-objective optimization counterpart. Through the integration of criterion-space-oriented cuts, the method iteratively navigates towards high-quality solutions. We showed efficacy of our approach through a computational study, demonstrating its rapid ability to yield high-quality solutions.

References

[1] Achterberg, T., Berthold, T., 2007. Improving the feasibility pump. Discrete Optimization 4 (1), 77–86.

[2] Ardakan, M. A., Hamadani, A. Z., 2014. Reliability optimization of series–parallel systems with mixed redundancy strategy in subsystems. Reliability Engineering & System Safety 130, 132 – 139.

[3] Ben-Tal, A., Nemirovski, A., 2001. On polyhedral approximations of the second-order cone. Mathematics of Operations Research 26 (2), 193–205.

[4] Bertacco, L., Fischetti, M., Lodi, A., 2007. A feasibility pump heuristic for general mixed-integer problems. Discrete Optimization 4 (1), 63–76.

[5] Boland, N. L., Eberhard, A. C., Engineer, F. G., Fischetti, M., Savelsbergh, M. W. P., Tsoukalas, A., 2014. Boosting the feasibility pump. Mathematical Programming Computation 6 (3), 255–279.

[6] Calkin, D. E., Montgomery, C. A., Schumaker, N. H., Polasky, S., Arthur, J. L., Nalle, D. J., 2002. Developing a production possibility set of wildlife species persistence and timber harvest value. Canadian Journal of Forest Research 32 (8), 1329–1342.

[7] Chakrabarty, D., Devanur, N., Vazirani, V. V., 2006. New results on rationality and strongly polynomial time solvability in Eisenberg-Gale markets. In: Internet and Network Economics. Vol. 4286 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 239–250.

[8] Charkhgard, H., Savelsbergh, M., Talebian, M., 2018. A linear programming based algorithm to solve a class of optimization problems with a multi-linear objective function and affine constraints. Computers & Operations Research 89, 17 – 30.

[9] Coit, D. W., Jun 2001. Cold-standby redundancy optimization for nonrepairable systems. IIE Transactions 33 (6), 471–478.

[10] Dai, R., Charkhgard, H., 2018. A two-stage approach for bi-objective integer linear programming. Operations Research Letters 46 (1), 81–87.

[11] Eisenberg, E., Gale, D., 1959. Consensus of subjective probabilities: The pari-mutuel method. The Annals of Mathematical Statistics 30 (1), 165–168.

[12] Feizabadi, M., Jahromi, A. E., 2017. A new model for reliability optimization of series-parallel systems with non-homogeneous components. Reliability Engineering & System Safety 157, 101 – 112.

[13] Fernandez-Antolin, A., Lurkin, V., de Lapparent, M., Bierlaire, M., 2017. Discrete-continuous maximum likelihood for the estimation of nested logit models. 16th Swiss Transport Research Conference, Ascona, Switzerland, 17-19 May.

[14] Fischetti, M., Glover, F., Lodi, A., 2005. The feasibility pump. Mathematical Programming 104 (1), 91–104.

[15] Fischetti, M., Salvagnin, D., 2009. Feasibility pump 2.0. Mathematical Programming Computation 1 (2), 201–222.

[16] Grötschel, M., Lovasz, L., Schrijver, A., 1988. Geometric Algorithms and Combinatorial Optimization. Springer-Verlag, Berlin.

[17] Haider, Z., Charkhgard, H., Kwon, C., 2018. A robust optimization approach for solving problems in conservation planning. Ecological Modelling 368, 288 – 297.

[18] Hensher, D. A., 1986. Sequential and full information maximum likelihood estimation of a nested logit model. The Review of Economics and Statistics 68 (4), 657–667.

[19] Jain, K., Vazirani, V. V., 2007. Eisenberg-Gale markets: Algorithms and structural properties. In: Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing. STOC '07. ACM, New York, NY, USA, pp. 364–373.

[20] Nash, J. F., 1950. The bargaining problem. Econometrica 18, 155–162.

[21] Nash, J. F., 1953. Two-person cooperative games. Econometrica 21, 128–140.

[22] Nicholson, E., Possingham, H. P., 2006. Objectives for multiple-species conservation planning. Conservation Biology 20 (3), 871–881.

[23] Pal, A., Charkhgard, H., 2019. Fpbh: A feasibility pump based heuristic for multi-objective mixed integer linear programming. Computers & Operations Research 112, 104760.

[24] Saghand, P. G., Charkhgard, H., 2022. A criterion space search algorithm for mixed integer linear maximum multiplicative programs: a multiobjective optimization approach. International Transactions in Operational Research 29 (3), 1659–1687.

[25] Saghand, P. G., Charkhgard, H., 2022. Exact solution approaches for integer linear generalized maximum multiplicative programs through the lens of multi-objective optimization. Computers Operations Research 137, 105549.

[26] Saghand, P. G., Charkhgard, H., Kwon, C., 2019. A branch-and-bound algorithm for a class of mixed integer linear maximum multiplicative programs: A bi-objective optimization approach. Computers & Operations Research 101, 263 – 274.

[27] Saghand, P. G., Rigterink, F., Mahmoodian, V., Charkhgard, H., 2023. Solving multiplicative programs by binary-encoding the multiplication operation. Computers & Operations Research 159, 106340.

[28] Serrano, R., 2005. Fifty years of the Nash program 1953-2003. Investigaciones Economicas, 219–258.

[29] Sierra-Altamiranda, A., Charkhgard, H., Eaton, M., Martin, J., Yurek, S., Udell, B. J., 2020. Spatial conservation planning under uncertainty using modern portfolio theory and nash bargaining solution. Ecological Modelling 423, 109016.

[30] Vazirani, V. V., 2012. Rational convex programs and efficient algorithms for 2-player Nash and nonsymmetric bargaining games. SIAM J. discrete math 26(3), 896–918.

[31] Williams, P. H., Araújo, M. B., Jun 2002. Apples, oranges, and probabilities: Integrating multiple factors into biodiversity conservation with consistency. Environmental Modeling & Assessment 7 (2), 139–151.