JCEC
THE JOURNAL OF ENGINEERING AND EXACT SCIENCES

OPEN ACCESS

The Journal of Engineering and Exact Sciences – jCEC, Vol. 10 N. 09 (2024)
journal homepage: https://periodicos.ufv.br/jcec
eISSN: 2527-1075
ISSN: 2446-9416

A novel approach for solving bi-level mono-objective and multi-objective programming problems using evolutionary algorithms

Article Info:
Article history: Received 2024-09-09 / Accepted 2024-12-17 / Available online 2024-12-17
doi: 10.18540/jcecvl10iss9pp20661

Wafa Bouguern
ORCID: https://orcid.org/0000-0002-5565-3581
Mathematical Analysis and Applications Laboratory, Department of Mathematics, University Mohamed El Bachir El Ibrahimi of Bordj Bou Arreridj, El Anasser 34030, Algeria
E-mail: wafa.bouguern@univ-bba.dz
Smail Addoune
ORCID: https://orcid.org/0000-0001-9158-9874
Mathematical Analysis and Applications Laboratory, Department of Mathematics, University Mohamed El Bachir El Ibrahimi of Bordj Bou Arreridj, El Anasser 34030, Algeria
E-mail: smail.addoune@univ-bba.dz
Hanene Debbiche
ORCID: https://orcid.org/ 0009-0008-1291-173X
Mathematical Analysis and Applications Laboratory, Department of Mathematics, University Mohamed El Bachir El Ibrahimi of Bordj Bou Arreridj, El Anasser 34030, Algeria
E-mail: hanene.debbiche@univ-bba.dz

Abstract
Bi-level programming problems (BLP) constitute an important class of non-convex optimization problems, which makes it challenging to find a global optimal solution. In this article, we propose an efficient technique to solve this category of problems. We reformulated the initial problem as a single-level optimization problem using the optimal value function of the lower-level problem. To solve the latter, we employed a technique based on α-dense curves to approximate the value function of the lower-level problem. Two evolutionary algorithms were then used to solve the reformulated problem. Furthermore, we extended our method to address multi-objective bi-level programming problems with a single objective at the upper level and multiple objectives at the lower level, known as a semi-vectorial bi-level programming problem. Several numerical experiments on nonlinear BLP show the outstanding efficiency of our approach.
Keywords: Nonlinear bi-level programming. Multi-objective optimization. Global optimization. α-dense curves. Evolutionary algorithms.

Nomenclature

For $x, u, v \in R^{n_2}$, $\|x\|$ represents the Euclidean norm of x. The notation $u \cdot v$ denotes the Hadamard product of the vectors u and v, i.e., $(u \cdot v)_i = u_i v_i$ for all $1 \leq i \leq n$, and e is the all-one vector, $u < v$ would mean $u_i < v_i$ for all $i = 1, \ldots, n$. The computation time is denoted by the symbol t, measured in seconds (s).

1. Introduction

Bi-level programming problems are a special type of hierarchical optimization problem. They are divided into two decision-making levels, namely, the upper and the lower levels, where the lower-level parametric optimization problem forming some of the constraints for the upper-level problem. Since a bi-level programming problem is typically non-convex, solving it is a challenging task. Over the past time, numerous approaches have been proposed in the literature for solving this problem, such as the Karush-Kuhn-Tucker approach (see, e.g., Dempe and Franke (2019) and references therein), penalty methods (Anandalingam and White (1990)), and branch and bound methods (Bard and Moore (1990)), etc.

In recent years, meta-heuristic algorithms have been increasingly popular for addressing BLP problems because of their beneficial characteristics. For example, Mathieu *et al.* (1994) have developed one of the initial evolutionary algorithms (EAs) intended for this objective. The Genetic Algorithm (GA) is the most prevalent version among these algorithms, with numerous adaptations examined in diverse research. Additional significant methods encompass the Bat Algorithm (BA) (Srivastava and Sahana (2019)), Differential Evolution (DE) (Angelo *et al*. (2014)), and Particle Swarm Optimisation (PSO) (Zhang *et al.* (2017)). The Grey Wolf Optimizer (GWO), a prominent meta-heuristic technique, was introduced by Mirjalili *et al.* (2014) and simulates the natural leadership structure and hunting tactics of grey wolves; it has been used in many scientific fields (see, e.g., Nouri *et all*. (2023)).

Certain studies have concentrated on the application of EAs to bi-level multi-objective optimization problems (BLMOP) (refer to, for instance, Ruuska and Miettinen (2012), Joao and Paulo (2014) and associated references), which can be divided into three types: multiple objectives at the upper level with a single objective at the lower level; a single objective at the upper level with multiple objectives at the lower level; and multiple objectives at both levels.

Addressing bi-level programming problems directly can be exceedingly difficult. As a result, certain academics have focused on converting BLPs into single-level optimization problems. There are two principal approaches for this reformulation. The initial technique entails substituting the lower-level problem with the KKT optimality conditions. This approach necessitates strong assumptions to establish that the optimal solution of the bi-level optimization problem aligns with that of the reformulated problem (see Dempe and Dutta (2012) for further details). The second method is referred to as lower-level value function reformulation (VFR), which was originally suggested for numerical applications (Outrata (1990)). The optimality conditions related to this strategy are examined in Ye and Zhu (1995). Significantly, VFR has demonstrated superior numerical performance compared to KKT reformulation, as indicated in Zemkoho and Zhou (2021).

Numerous studies advocate for a nested methodology in resolving BLP issues, wherein a classical technique manages the lower level and an EA solves the upper level (see, e.g., Zhao and Gu (2006), Jialin *et all.*, and Jie (2016)). This strategy may be computationally intensive and ineffective for large-scale issues. We suggest a more effective approach to mitigate these limitations, intending to decrease computing costs and expedite the solution process. In this work, we utilize the value function reformulation approach. We establish a technique that calculates an approximation of the lower-level value function, employing the implementation of α-dense curves within the feasible domain, as first introduced by Mora and Cherruault (1997). These curves have been effectively applied in various fields of applied mathematics (see, e.g., Mora and Mora-Porta (2005), Butz (1972)). Our methodology is based on evolutionary algorithms, which have proven effective in various applied scientific fields for solving large-scale nonlinear problems. Furthermore, we offer a concise examination of employing the same methodology to address semi-vectorial, bi-level programming problems.

The rest of this paper is organized as follows: Section 2 presents the bi-level programming problem formulation, where some definitions and properties are stated. In Section 3, evolutionary

algorithms are introduced. In Section 4, a novel proposed technique is presented. In Section 5, we use the same method to solve the semi-vectorial bi-level programming problem. In Section 6, the experimental results of the proposed algorithm are presented. Also, a comparison between two types of evolutionary algorithms is made. Finally, Section 7 provides a conclusion and future work.

2. Problem Formulation

In this paper, we examine a class of nonlinear bi-level programming problems, where the lower-level problem is a parametric optimization problem with box constraints. This problem can be defined as follows:

$$(P)\quad \begin{cases} \underset{x,y}{Minimize}\ F(x,y) \\ \quad s.t. \\ a \le x \le b, \\ g(x,y) \le 0, \\ y \in \Psi(x), \end{cases}$$

where $\Psi(x)$ is the set of optimal solutions to the lower-level problem

$$(P_x)\quad \begin{cases} \underset{y}{Minimize}\ f(x,y) \\ \quad s.t. \\ c \le y \le d \end{cases}$$

There are two classes of variables in this problem: the upper-level variables $x \in R^{n_1}$ and the lower-level variables $y \in R^{n_2}$. The functions $F, f : R^{n_1} \times R^{n_2} \to R$ are, respectively the upper and lower objective functions, $g : R^{n_1} \times R^{n_2} \to R^m$ represents the upper-level constraints, $a, b \in R^{n_1}, c, d \in R^{n_2}$ with n_1, n_2 and m are integers. The current research indicates that the following definitions are essential for studying (P).

1. The feasible set of the lower level problem (P_x) for every fixed x:

   $S = \{y \in R^{n_2} \mid c \le y \le d\}$.

2. The constraint region of (P):

   $\hat{S} = \{(x,y) \in R^{n_1} \times R^{n_2} \mid a \le x \le b, c \le y \le d, g(x,y) \le 0\}$.

3. The rational reaction set of the lower level, for every fixed x, is:

   $R(x) = argmin\{f(x,y), y \in S\}$.

4. The inducible region of (P):

   $IR = \{(x,y) \in R^{n_1} \times R^{n_2} \mid (x,y) \in \hat{S}, y \in R(x)\}$.

The definitions of feasible and optimal solutions for (P) are given as follows:

Definition 1 A point (x, y) is feasible of (P) if $(x, y) \in$ IR.

Definition 2 A point (x^*, y^*) is an optimal solution of (P) if $(x^*, y^*) \in$ IR, and $F(x^*, y^*) \le F(x, y)$ for all $(x, y) \in$ IR.

3. Evolutionary Algorithms

Evolutionary algorithms are stochastic, population-based direct search techniques that emulate natural evolution. This section provides a succinct summary of two metaheuristic techniques, Grey Wolf Optimizer (GWO) and Particle Swarm Optimization (PSO), along with an explanation of their functional mechanics.

3.1 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO), proposed by Kennedy and Eberhart (1995), simulates the behavior of fish and birds in groups. The methodology employs mathematical equations that specify the positions and velocities of particles, allowing them to traverse the solution space effectively. A particle i is designated by x_i, representing its position, while its velocity is indicated by v_i. The following is the updating rule for both position and velocity:

$$v_i^{k+1} = w\, v_i^k + c_1 rand_1 \left(P_i^{best} - x_i^k\right) + c_2 rand_2 \left(P_{gbest} - x_i^k\right) \quad (1)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (2)$$

where k indicates the current iteration, $rand_1$ and $rand_2$ are two random numbers uniformly distributed in [0,1], c_1 and c_2 are the acceleration coefficients, the factor w is the inertia weight; P_i^{best} is the best solution previously found for the i^{th} particle; and P_{gbest} is the best solution previously found in the swarm (global best solution).

3.1 Grey Wolf Optimizer (GWO)

Grey Wolf Optimization is a population-based metaheuristic algorithm that draws inspiration from nature. Presented by Mirjalili *et al*. (2014), it is based on how grey wolves behave. The GWO algorithm uses a mathematical model of the grey wolf social hierarchy. The optimization method concentrates on three principal solutions: the best solution, denoted as α; the second-best solution, denoted as β; and the third solution, denoted as σ. The following equations are suggested:

$$A = 2a^{(k)} \cdot rand_1 - a^{(k)} \quad (3)$$

$$C = 2rand_2 \quad (4)$$

$$D = |C \cdot X_p^k - X^k| \quad (5)$$

$$X^{k+1} = X_p^k - A \cdot D \quad (6)$$

where $a^{(k)} = \lambda_k e$ with the scalar λ_k defined by the following formula:

$$\lambda_k = 2 - \frac{2k}{max} \quad (7)$$

where max is the maximum number of iterations, and k is the current iteration. The vectors A and C represent acceleration coefficients, the prey's location vector is indicated as X_p; the grey wolf's position vector is denoted by X. Additionally, $rand_1$ and $rand_2$ are two random vectors in the interval [0,1].

Since the prey's location is unknown, we will replace it with the three best solutions:X_α, X_β, and X_σ. So the wolves use these equations to update their positions.

$$D_\alpha = |C_1 \cdot X_\alpha - X|,\ D_\beta = \left|C_2 \cdot X_\beta - X\right|,\ D_\sigma = |C_3 \cdot X_\sigma - X|,$$

$$X_1 = X_\alpha - A_1 \cdot D_\alpha,\quad X_2 = X_\beta - A_2 \cdot D_\beta,\quad X_3 = X_\sigma - A_3 \cdot D_\sigma,$$

$$X_{k+1} = \frac{X_1 + X_2 + X_3}{3} \tag{8}$$

Here, $|X|$ denotes the vector of absolute values of each component of X.

4. The New Proposed Technique

The first step in this section is to convert (P) into a single-level optimization problem. Using the optimal value function reformulation (VFR), we get the following equivalent problem:

$$(P) \quad \begin{cases} \underset{x,y}{Minimize} \ F(x,y) \\ \quad s.t. \\ a \ \leq x \leq b, \\ c \ \leq y \leq d, \\ g(x,y) \leq 0, \\ f(x,y) - V(x) \ \leq 0, \end{cases}$$

where the optimal value function is defined by

$$V(x) = \min_{y} \{f(x,y) \mid c \ \leq \ y \ \leq \ d\}.$$

The problem (P) is intricate and requires significant effort to solve due to its non-convexity and the presence of the non-differentiable function $V(x)$. The presence of non-convexity hampers the resolution of optimization problems, and the involvement of typically non-differentiable functions further exacerbates the issue. Addressing such issues frequently necessitates specialized methodologies and a comprehensive analysis of the problem's framework and attributes.

Many researchers have used the VFR approach to develop optimality conditions (see, e.g., Stephan *et all.* (2007), Jane (2005)). However, few recent studies have focused on numerical techniques (see, e.g., Lin *et all.* (2014), Xu and Ye (2014)).

In order to solve (P), we can use an exact penalty function approach and formulate it as follows:

$$(P_\mu) \quad \begin{cases} \underset{x,y}{Minimize} \ F(x,y) + \mu H(x,y) \\ \quad s.t. \\ a \ \leq x \leq b, \\ c \ \leq y \leq d, \end{cases}$$

where

$$H(x,y) = max \ \{f(x,y) - V(x), \ \ 0\} + \sum_{i=1}^{m} max \ \{0, \ \ g_i(x,y)\}.$$

Our principal study focuses on resolving the penalty problem (P_μ) through the application of EAs. A multitude of scholars have employed EAs to address bi-level programming challenges. Numerous studies solve the lower-level problem for each particle in the population using classical methods, thereafter addressing the upper-level problem with EAs (see, e.g., Zhao and Gu (2006), Jialin *et al.* (2016)). This approach is often computationally demanding.

We suggest a viable and inexpensive technique that converts the two-level problem into a single-level problem, as previously described. EA cannot be directly implemented because of the

characteristics of the function $V(x)$; therefore, we must focus on the optimal value function $V(x)$. The basic idea for determining $V(x)$ is to substitute it with an appropriate approximation, as described below:

$$V(x) = \min_{y}\{\ f(x,y) \mid c \le y \le d\ \}$$
$$\approx min\{\ f(x,y_1), f(x,y_2), \dots, f(x,y_p)\ \mid c \le y_j \le d,\ \ j = 1, \dots, p\ \}.$$

where $\{y_1, y_2, \dots\ y_p\}$ are p points of the set $[c,d]$. We aim to cover the set $[c,d]$ by distributing the points $\{y_1, y_2, \dots, y_p\}$ across it. To do this, we will convert the multidimensional lower-level problem into a one-dimensional problem by employing a parametric α-dense curve (see Mora and Cherruault (1997)). The primary characteristic of these curves is their ability to represent n variables with just one variable. Let us recall the definition of an α-dense curve.

Definition 3 Let $\alpha > 0$, a subset $D \subset S$ is said to be α-dense in S if, for every $y \in S$, there exists a point $\bar{y} \in D$ such that $\|y - \bar{y}\| \le \alpha$.

Definition 4 A curve $\gamma : [0,\beta] \to S$, where $\beta > 0$, is said to be α-dense in S if $\gamma([0,\beta])$ is α-dense in S, i.e., for every $y \in S$ there exists $t \in [0,\beta]$ such that $\|y - \gamma(t)\| \le \alpha$, where $\gamma(t) = (\gamma_1(t), \gamma_2(t), \dots, \gamma_{n_2}(t))$.

Theorem 1 (Ziadi and Bencherif-Madani (2023)) Let $\gamma(t) = \left(\gamma_1(t), \gamma_2(t), \dots, \gamma_{n_2}(t)\right) : [0,\beta] \to [c,d]$ be a continuous parametrized curve. Additionally, let $\theta_1, \theta_2, \dots, \theta_{n_{2-1}}$, α be strictly positive numbers satisfying the following conditions:

1. γ_{n_2} is surjective.
2. For any $i = 1, \dots, n_{2-1}$, γ_i reaches its bounds c_i and d_i within every closed interval of length θ_i
3. For any $i = 1, \dots, n_2 - 1$ and any interval $I \subseteq [0,\beta]$, we have

$$\mu(I) < \theta_i \implies \mu\left(\gamma_{i+1}(I)\right) < \frac{\alpha}{\sqrt{n_2 - 1}},$$

where $\mu(\cdot)$ is the Lebesgue measure. Given these conditions, the conclusion is that the curve γ is α-dense in $[c,d]$.

The unidimensional problem, which depends on the single variable t:

$$\min_{t \in [0,\beta]} f^*(x,t)$$

where $f^*(x,t) = f\left(x, \gamma(t)\right)$ represents an approximation of the multidimensional problem (P_x). In this formulation, the objective function $f^*(x,t)$ is an approximation of the objective function $f(x,y)$ in the problem (P_x).

The density curve that we will use in our work is given by the following formula (taken from Ziadi and Bencherif-Madani (2023)). Let $\alpha > 0$ be a given number; consider the function $\gamma : \left[0, \frac{\pi}{\theta_{n_2}}\right] \to [c,d]$ such that

$$\gamma_1(t) = \frac{c_1 + d_1}{2} - \frac{c_1 - d_1}{2}\cos\theta_1\, t,$$
$$\gamma_2(t) = \frac{c_2 + d_2}{2} - \frac{c_2 - d_2}{2}\cos\theta_2\, t,$$
$$\vdots$$
$$\gamma_{n_2}(t) = \frac{c_{n_2} + d_{n_2}}{2} - \frac{c_{n_2} - d_{n_2}}{2}\cos\theta_{n_2}\, t,$$

where $\theta_1, \theta_{2,}\ldots, \theta_{n_2}$ are the parameters specified by:

$$\theta_1 = 1,$$
$$\theta_2 = \frac{\alpha}{\pi(|c_2| + |d_2|)},$$
$$\theta_3 = \frac{\alpha^2}{\pi^2(|c_2| + |d_2|)(|c_3| + |d_3|)}$$
$$\vdots$$
$$\theta_{n_2} = \frac{\alpha^{n_2-1}}{\pi^{n_2-1}(|c_2| + |d_2|)(|c_3| + |d_3|)\cdots(|c_{n_2}| + |d_{n_2}|)}.$$

By Theorem 1, the parametrized curve $\gamma(t) = (\gamma_1(t), \ldots, \gamma_{n_2}(t))$ is α-dense in $[c, d]$.
Figure 1 illustrates the densification of the square $[-1,1] \times [-1,1]$ by the support of the provided curve for $\alpha = 0.3$ and $\alpha = 0.1$. It is evident that the curve more effectively covers the designated area when α is smaller.

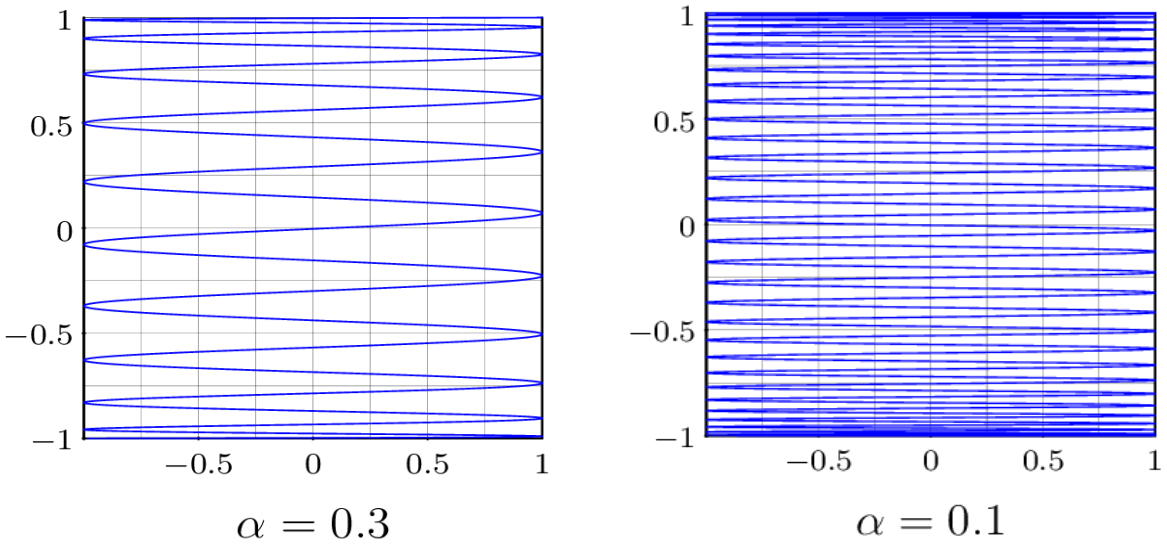


Figure 1 – the densification of the squar $[-1, 1]^2$ by the curve γ.

After establishing our comprehension of α-dense curves, we now focus on their application in populating the set S. By choosing a minimal value for α, we guarantee a high density, thereby covering a significant portion of the set S. Our objective is to uniformly distribute the points $\{y_1, y_2, \ldots, y_p\}$ along this curve, where p denotes the number of points in the set $[c, d]$, as seen in Figure 2. The number of points p in the distribution along the curve can be modified as required. It is clear that augmenting p results in a superior approximation of the set S.

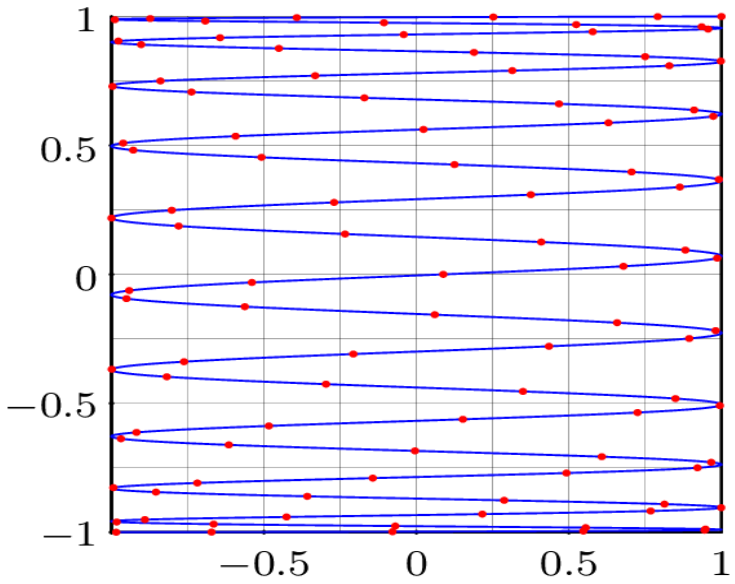


Figure 2 – The distribution points of the curve for 100 points with $\alpha = 0.3$.

Consequently, we can resolve the following problem:

$$(P'_\mu) \begin{cases} \underset{x,y}{Minimize}\ G(x,y) = F(x,y) + \mu H_{app}(x,y) \\ s.t. \\ a \le x \le b, \\ c \le y \le d, \end{cases}$$

where

$$H_{app}(x,y) = max\ \{f(x,y) - V_{app}(x),\ 0\} + \sum_{i=1}^{m} max\ \{0,\ g_i(x,y)\}$$

and

$$V_{app}(x) = \min_{y} \left\{ f\left(x, \gamma(t_j)\right) : j = 1, \dots, p \right\}$$

where $\{t_1, t_2, \dots, t_p\}$ represents a subdivision of the interval $\left[0, \frac{\pi}{\theta_{n_2}}\right]$. The steps in this subdivision should be small, meaning that for a small $\varepsilon > 0$,

$$\max_{1 \le i \le p-1} \{t_{i+1} - t_i\} < \varepsilon.$$

We can also use a uniform subdivision by setting $t_i = i\,h$, where $h = \frac{\pi}{p\,\theta_{n_2}}$ for $i = 0, \dots, p$. In this case, the step size h decreases as p increases.

Below, we provide two algorithms (BL-GWO, BL-PSO) based on PSO and GWO to address the final approximate problem.

Algorithm 1 BL-GWO Algorithm

Step 0. Configure the parameters: population size n, maximum number of iterations max, penalty parameter $\mu > 0$, density $\alpha > 0$, and number of distribution points p.

Step 1. Initialization

1. Randomly initialize the grey wolf population $P = \{(x_i, y_i): i = 1, \dots, n\}$
2. Generate the distribution points $z_i = \gamma(t_i),\ i = 1, \dots, p$, and set

$$V_{app}(x) = \min_{1 \le i \le p} \{f(x, z_i)\}.$$

3. Set $H(x,y) = \max\{f(x,y) - V_{app}(x), 0\} + \sum_{i=1}^{m} \max\{0,\ g_i(x,y)\}$.
4. Set $G(x,y) = F(x,y) + \mu\, H(x,y)$.

To begin, calculate the fitness values $G(x_i, y_i)$ of each individual of the population and choose the three best solutions: X_α the best solution, X_β is the second-best solution, X_σ is the third-best solution.

Step 2. Treatment

1. **For** $i = 1\ to\ max$ **do**
2. **For** each $X \in P$ **do**
    i. Update $a^{(k)}, A$, and C using formulas (7), (3), (4).
    ii. Update the position according to equation (8).
    iii. Calculate its fitness value.
3. **End For**
4. Update the solutions X_α, X_β, and X_σ and set $i := i + 1$.
5. **End For**
6. return X_α.

Algorithm 2 BL- PSO Algorithm

Step 0. Configure the parameters: population size n, maximum number of iterations max, penalty parameter $\mu > 0$, density $\alpha > 0$, number of distribution points p, acceleration coefficients $c_1$and c_2, and inertia weight w.

Step 1. Initialization

1. Randomly initialize the particle population $P = \{(x_i, y_i): i = 1, \dots, n\}$ and their velocity equals zero for all components.
2. Generate the distribution points $z_i = \gamma(t_i),\ i = 1, \dots, p$, and set

$$V_{app}(x) = \min_{1 \le i \le p} \{f(x, z_i)\}.$$

3. Set $H(x,y) = \max\{f(x,y) - V_{app}(x), 0\} + \sum_{i=1}^{m} \max\{0,\ g_i(x,y)\}$.
4. Set $G(x,y) = F(x,y) + \mu\, H(x,y)$.
5. Determine the global best solution (P_{gbes}).

Step 2. Treatment

1. **For** $i = 1\ to\ max$ **do**
2. **For** each $X \in P$ **do**
    i. Update the position and velocity using equations (1), (2).
    ii. Calculate its fitness value. $G(x_i, y_i)$.
    iii. Update the personal best position P_{best}.
    iv. Update the global best position P_{gbest}.
3. **End For**
4. Set $i := i + 1$.
5. **End For**
6. return P_{gbest}.

---

5. Semivectorial Bi-level Programming Problem

This section begins with preliminary knowledge of the multi-objective optimization problems discussed in this paper. It then provides a detailed description of the application of our algorithm (BL-PSO) in solving semi-vectorial bi-level programming problems.

A general multi-objective programming problem (MOP) can be formulated as

$$(MOP)\ \begin{cases} \underset{x}{Minimize}\ f(x) := \left(f_1(x), \dots, f_q(x)\right) \\ s.t \\ x \in X. \end{cases}$$

where X is a non-empty subset of R^n, and $f: X \to R^q$ (with $q \geq 2$), the space R^n containing the set X of admissible points is called the decision space. The set $Y = f(X)$, called the set of admissible vectors, is in the criterion space R^q.

Definition 5 A point $\bar{x} \in X$ is Pareto optimal (or efficient) if there is no $x \in X$ that is better than it. This means that there is no point $x \in X$ such that $f_i(x) \leq f_i(\bar{x})$ for all $i = 1, \dots, q$ and $f_k(x) < f_k(\bar{x})$ for at least one k.

Definition 6 A point $x^* \in X$ is weakly efficient if there is no point $x \in X$ such that $f(x) < f(x^*)$.

One of the most commonly employed methods for solving (MOP) is the weighted sum scalarization method. This approach facilitates the identification of efficient solutions for (MOP) by solving the following single-objective optimization problem:

$$(WS_\lambda)\ \left\{ \underset{x \in X}{Minimize} \sum_{i=1}^{q} \lambda_i f_i(x), \right.$$

where λ represents the scalar weights, with $\lambda \in \Sigma_q$, and

$$\Sigma_q = \left\{ \lambda \in R^q : \lambda \geq 0, \sum_{i=1}^{q} \lambda_i = 1 \right\}$$

The following propositions, as presented in Ehrgott (2005), illustrate the relationship between the multi-objective optimization problem (MOP) and the weighted sum (WS_λ) problem.

Proposition 1 (Ehrgott (2005)) Let x^* be an optimal solution of the weighted sum problem (WS_λ). The following statements hold.

- If $\lambda \in \Sigma_q$, then x^* is weakly efficient.
- If $\lambda \in \Sigma_q^+$, then x^* is efficient, where $\Sigma_q^+ = \{\lambda \in R^q : \lambda > 0,\ \sum_{i=1}^q \lambda_i = 1\}$.

Proposition 2 (Ehrgott (2005)) Let f_k, for $k = 1, \dots, q$, be convex functions, and X be a convex set, then x^* is weakly efficient if and only if there exists $\lambda \in \Sigma_q$, such that x^* is an optimal solution of (WS_λ).

Now, we examine a class of semi-vectorial bi-level programming problems in which the lower-level problem is a parametric multi-objective optimization problem with box constraints. The problem can be formulated as follows:

$$(BMP)\ \begin{cases} \underset{x,y}{\text{Minimize}}\ F(x,y) \\ s.t. \\ a \le x \le b, \\ g(x,y) \le 0, \\ y \in \Psi_{we}(x), \end{cases}$$

where $\Psi_{we}(x)$ is the set of weakly efficient solutions of the lower level problem

$$(MP_x)\ \begin{cases} \underset{y}{\text{Minimize}}\ f(x,y) := \big(f_1(x,y), f_2(x,y), \dots, f_q(x,y)\big) \\ s.t. \\ c \le y \le d, \end{cases}$$

where $F : R^{n_1} \times R^{n_2} \to R,\ g : R^{n_1} \times R^{n_2} \to R^m,\ f_i : R^{n_1} \times R^{n_2} \to R,\ i = 1, \dots, q.$

$\Psi_{we}(x) = \{y \in S : \nexists\ \bar{y} \in S,\ such\ that\ f(x,\bar{y}) < f(x,y)\}.$

To rewrite problem (BMP) as a single-level problem, we assume that $f_i(x, y)$, for $i = 1, \dots, q$, are convex in y when x is fixed. By applying the weighted sum scalarization to the lower-level problem, (BMP) can be reformulated as follows:

$$(BMP_s)\ \begin{cases} \underset{x,y,\lambda}{Minimize}\ F(x,y) \\ s.t. \\ a \le x \le b, \\ 0 \le \lambda \le 1,\ \sum_{i=1}^{q} \lambda_i = 1, \\ g(x,y) \le 0, \\ y \in S(x,\lambda), \end{cases}$$

where $S(x,\lambda) = \underset{c \le y \le d}{argmin} \sum_{i=1}^q \lambda_i f_i(x,y).$

This conversion has been utilized by various researchers (see, e.g., Gupta and Ong (2015), Li and L.Zhang (2021)). The relationship between the scalarized problem (BMP_s) and the original bi-level multiobjective problem (BMP) has been examined in Dempe and Zemkoho (2013) and Dempe and Mehlitz (2019). Results concerning local optimality are discussed in Dempe and Zemkoho (2013) and corrected in Dempe and Mehlitz (2019). It has been established that the problems (BMP) and (BMP_s) are equivalent in terms of global optimal solutions (see Dempe and Mehlitz (2019)).

Using the optimal value function reformulation and an exact penalty function approach we get the following problem:

$$(BL_\mu)\begin{cases} \underset{x,y,\lambda}{Minimize}\ F(x,y)+\mu H(x,y,\lambda) \\ s.t. \\ a \le x \le b, \\ c \le y \le d, \\ 0 \le \lambda_i \le 1, i=1,\dots,q, \end{cases}$$

where

$$H(x,y,\lambda) = max\left\{\sum_{i=1}^{q}\lambda_i f_i(x,y) - V(x), 0\right\} + \sum_{i=1}^{m} max\{0,\ g_i(x,y)\} + \left|\sum_{i=1}^{q}\lambda_i - 1\right|$$

and the optimal value function is defined by

$$V(x,\lambda) = \min_{y}\left\{\sum_{i=1}^{q}\lambda_i f_i(x,y) \mid c \le y \le d\right\}.$$

Finally, we apply our proposed algorithm, BL-PSO, to solve (BL_μ).

6. Computational Tests

In this section, we implement our algorithm on several nonlinear bi-level programming problems to illustrate the efficacy of the suggested method. Furthermore, we compare the solutions obtained by the proposed algorithm with those found in related references. We evaluate the computational efficiency of BL-GWO and BL-PSO in terms of computational time and solution quality. The parameters for BL-PSO are established as follows: acceleration coefficients $c_1 = c_2 = 1.2$, inertia weight $w = 0.8$, population size $n = 1000$, and $\alpha = 0.01$. We implement the algorithm using Julia 1.10.1.

6.1 Test Problems

Problem 1. (Mitsos and Barton (2006))

$$\begin{cases} \underset{x,y}{Minimize}\ \left(x+\frac{1}{2}\right)^2 + \frac{1}{2}y^2 \\ s.t. \\ -1 \le x \le 1, \\ y \in \underset{-1\le y\le 1}{argmin}\ \frac{1}{2}xy^2 + \frac{1}{4}y^4 \end{cases}$$

*Problem 2. (*Mitsos and Barton (2006))

$$\begin{cases} \underset{x,y}{Minimize}\ y \\ s.t. \\ -1 \le x \le 1, \\ y \in \underset{-0.8\le y\le 1}{argmin}\ x\left(16y^4 + 2y^3 - 8y^2 - \frac{3}{2}y + \frac{1}{2}\right) \end{cases}$$

*Problem 3. (*Oduguwa and Roy (2002))

$$\begin{cases} \underset{x,y}{Minimize}\ (x-1)^2+(y-1)^2 \\ s.t. \\ x \geq 0, \\ y \in \underset{y\geq 0}{argmin}\ \frac{1}{2}y^2+500y-50xy \end{cases}$$

*Problem 4.(*Mitsos and Barton(2006)).

$$\begin{cases} \underset{x,y}{Minimize}\ x^2-y \\ s.t. \\ 0 \leq x \leq 1, \\ y \in \underset{0\leq y\leq 3}{argmin}((y-1-0.1x)^2-0.5-0.5x)^2 \end{cases}$$

Problem 5. (Mitsos and Barton (2006))

$$\begin{cases} \underset{x,y}{Minimize}\ x_1y_1+x_2y_2^2+x_1x_2y_3^3 \\ s.t. \\ -1 \leq x \leq 1, \\ 0.1-x_1^2 \leq 0, \\ 1.5-y_1^2-y_2^2-y_3^2 \leq 0, \\ -2.5+y_1^2+y_2^2+y_3^2 \leq 0, \\ y \in \underset{-1\leq y\leq 1}{argmin}\ x_1y_1^2+x_2y_2^2+(x_1-x_2)y_3^2 \end{cases}$$

*Problem 6. (*Mitsos and Barton (2006))

$$\begin{cases} \underset{x,y}{Minimize}\ (x-3)^2+(y-2)^2 \\ s.t. \\ 0 \leq x \leq 8, \\ -2x+y \leq 1, \\ x-2y \leq -2, \\ x+2y \leq 14, \\ y \in \underset{0\leq y\leq 10}{argmin}\ (y-5)^2 \end{cases}$$

*Problem 7. (*Shimizu and Aiyoshi (1981))

$$\begin{cases} \underset{x,y}{Minimize}\ (x_1-30)^2+(x_2-20)^2-20y_1+20y_2 \\ s.t. \\ -x_1-2x_2+30 \leq 0, \\ x_1+x_2-25 \leq 0, \\ x_2-15 \leq 0, \\ y \in \underset{0\leq y\leq 10}{argmin}(x_1-y_1)^2+(x_2-y_2)^2 \end{cases}$$

*Problem 8. (*Ma and Wang (2020))

$$\begin{cases} \underset{x,y}{\text{Minimize}} \sum_{i=1}^{10}(|x_i-1|+|y_i|) \\ s.t. \\ y \in \underset{-\pi\leq y\leq \pi}{argmin} \exp\left(\left[1+\sum_{i=1}^{10}(y_i^2/4000)-\prod_{i=1}^{10}\cos\left(y_i/\sqrt{i}\right)\right]\sum_{i=1}^{10}x_i^2\right) \end{cases}$$

*Problem 9. (*Dempe and Mehlitz (2019))

$$\begin{cases} \underset{x,y}{Minimize}\ y-x \\ s.t. \\ 0 \leq x \leq 1, \\ y \in \underset{0\leq y\leq 1}{argmin}\ (xy, 1-y) \end{cases}$$

*Problem 10. (*João *et all* (2015))

$$\begin{cases} \underset{x,y}{Minimize}\ (y_1-1)^2+y_2^2+x^2 \\ s.t. \\ -1 \leq x \leq 2, \\ y \in \underset{-1\leq y\leq 2}{argmin}\ (y_1^2+y_2^2,\ (y_1-x)^2+y_2^2) \end{cases}$$

6.2 Results

The experiments for each example are repeated 20 runs, ensuring that the best solution from each run is selected as the respective global optimum.

Table 1 specifies the parameters for p, max, and μ. Table 2 compares the solutions obtained with the proposed algorithm against those found in relevant references, showing the best solutions (x^*, y^*) and the corresponding value F^* achieved at the upper level. Table 3 presents a

comparative study between the two evolutionary algorithms, BL-PSO and BL-GWO. Table 4 compares our algorithm with the one proposed by Zhao and Gu (2006).

Table 1 – Parameters of μ, p, and max for solving problems $1-10$.

Problems	μ	p	max
p_1	10^6	1000	100
p_2	100	1000	100
$p_3 - p_6$	10^6	1000	100
p_7	100	6000	100
p_8	100	1000	200
p_9	100	1000	100
p_{10}	100	1000	200

Table 2 – Comparison between the best solutions found by our proposed algorithm and the results obtained in the corresponding references.

	BL-PSO		Ref.	
Problems.	(x^*, y^*)	F^*	(x^*, y^*)	F^*
P_1	$(-0.25, \pm 0.5)$	0.1868	$(-0.25, \pm 0.5)$	0.1875
P_2	$(4.68 \times 10^{-11}, -0.8)$	-0.8	$(0, -0.8)$	-0.8
P_3	$(1, 0)$	1	$(10.04, 0.1429)$	82.44
P_4	$(0.21, 1.80)$	-1.75	$(0.21, 1.79)$	-1.75
P_5	$(-1, -1, 1, \pm 1, -0.7071)$	-2.35	$(-1, -1, 1, 1, -0.707)$	-2.35
P_6	$(3, 4.99)$	8.95	$(3, 5)$	9
P_7	$(20, 4.99, 10, 4.82)$	221.511	$(20, 5, 9.77, 4.95)$	228.7
P_8	$(0.996, 0.999, 1.000, 0.999, 0.999, 1.000, 0.999, 0.999, 1.000, 1.000 -0.001, 0.005, -7.395 \times 10^{-5}, -0.0013, 0.000, 0.000, -0.000, 0.002, 0.001, -0.000)$	0.018	$(1.00, 1.00, 1.00, 1.00, 1.00, 0.999, 1.00, 0.999, 1.00, 1.00, 3.56 \times 10^{-6}, -2.11 \times 10^{-7}, 7.38 \times 10^{-7}, 5.02 \times 10^{-7}, -5.38 \times 10^{-7}, -1.26 \times 10^{-6}, -9.99 \times 10^{-7}, -2.30 \times 10^{-6}, -9.08 \times 10^{-8}, 1.7 \times 10^{-6})$	3.26×10^{-3}

Comments. First, we mention that Problems $1, 2, 4, 5$, and 6 are taken from the reference Mitsos and Barton (2006), which provide theoretical solutions to these problems. When we compared these theoretical solutions with our obtained results, we found that our algorithm consistently produced perfect solutions. Notably, for Problem 5, our algorithm revealed the presence of two global solutions, whereas only one solution was indicated in Mitsos and Barton (2006). The algorithm referenced in Oduguwa and Roy (2002) fails to attain the global optimum for Problem

3, while our algorithm successfully achieves it. For Problems 7 and 8, our algorithm yields highly satisfactory results. Table 3 provides an overview of the comparison results between BL-GWO and BL-PSO. The results show that both algorithms perform similarly in terms of solution quality, but BL-GWO takes more time than BL-PSO.

Table 3 – Performance comparison between BL-PSO and BL-GWO.

	BL-GWO			BL-PSO		
Problems.	(x^*, y^*)	F^*	t/s	(x^*, y^*)	F^*	t/s
$\boldsymbol{P_1}$	$(-0.25, 0.5)$	0.186	6.8	$(-0.25, 0.5)$	0.186	4.8
$\boldsymbol{P_2}$	$(4.0833 \times 10^{-20}, -0.8)$	-0.8	8.6	$(4.688 \times 10^{-11}, -0.8)$	-0.8	5.4
$\boldsymbol{P_3}$	$(1, 0)$	1	4.3	$(1, 0)$	1	3.2
$\boldsymbol{P_4}$	$(0.21, 1.80)$	-1.75	5.0	$(0.21, 1.80)$	-1.75	3.2
$\boldsymbol{P_5}$	$(-1, -1, 1, -1, -0.7070)$	-2.353	4.7	$(-1, -1, 1 -1, -0.7071)$	-2.353	3.2
$\boldsymbol{P_6}$	$(3, 4.99)$	8.952	3.4	$(3, 4.99)$	8.952	2.4
$\boldsymbol{P_7}$	$(19.99, 5.001, 10, 4.82)$	221.54	25.5	$(20, 4.99, 10, 4.82)$	221.51	16.5
$\boldsymbol{P_8}$	$(0.999, 0.999, 0.999, 1.000, 1.000, 1.000, 0.999, 0.999, 1.000, 0.999$ $0.007, 0.002, 0.003, -0.012, -0.035, 0.001, 0.008, -0.012$ $-0.003, 0.001)$	0.094	307.9	$(0.996,0.999, 1.000,0.999, 0.999,1.000, 0.999,0.999, 1.000,1.000$ $-0.001,0.005, -7.39 \times 10^{-5}, -0.0013, 0.000,0.000,$ $-0.000,0.002, 0.001, -0.000)$	0.018	235.0

Comparison between our algorithm and the algorithm proposed by Zhao and Gu (2006)

This part compares the performance of our algorithm with that of the algorithm proposed by Zhao and Gu (2006). We selected Problems 3, 6, and 7 for comparison, using parameters $n = 40$, $max = 100$, $p = 1000$, and $\alpha = 0.01$. Table 4 presents the results.

Table 4 – Comparing our algorithm with the one proposed by Zhao and Gu (2006).

	BL-PSO		\cite{Zhao2006}	
Problems.	$(\boldsymbol{x}^*, \boldsymbol{y}^*)$	$\boldsymbol{t/s}$	$(\boldsymbol{x}^*, \boldsymbol{y}^*)$	$\boldsymbol{t/s}$
$\boldsymbol{P_3}$	$(1.0, 0.0)$	0.15	$(1.0, -9.974 \times 10^{-9})$	50.19
$\boldsymbol{P_6}$	$(3.000, 4.992)$	0.10	$(3.0, 4.9999)$	46.27
$\boldsymbol{P_7}$	$(19.997, 5.001, 10.0, 4.758)$	0.40	$(19.837, 5.109, 10.00, 5.109)$	83.87

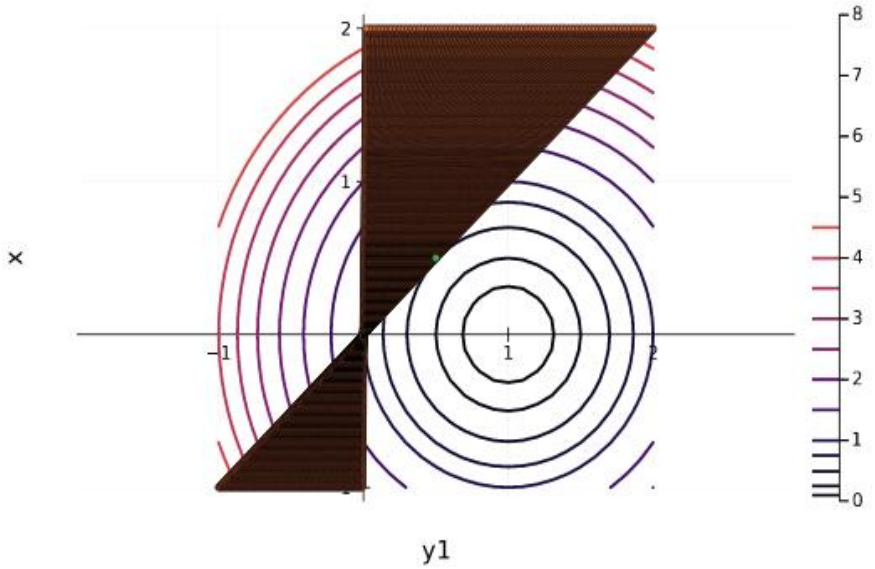
Although our approach relies on approximating the value function of the lower level, resulting in approximate solutions, it is significantly faster and more efficient in computational time compared to the algorithm proposed by Zhao and Gu (2006).

Results of the two examples 9, 10: In Problem 9, the set of weakly efficient solutions to the lower-level problem is the interval [0,1] for all $x \in [0,1]$. Hence, $(x^*, y^*) = (1, 0)$ represents the unique optimal solution. When we apply our BL-PSO algorithm, we also find $(x^*, y^*) = (1, 0)$ with $\lambda = (1, 0)$.

For the Problem 10, we apply the weighted sum approach to the lower-level problem

$$\begin{cases} \underset{y}{\text{Minimize}}\ (y_1^2 + y_2^2, (y_1 - x)^2 + y_2^2) \\ s.t. \\ -1 \leq y \leq 2, \end{cases}$$

Using different values of λ_i, (where $i = 1,2$), with the constraint that $\lambda_1 + \lambda_2 = 1$, we obtain the set of efficient points shown in Figure 3. Since y_2 is zero in all efficient solutions, we plot the graph using the y_1 and x axes.



The theoretical optimal solution is the point $(x^*, y^*) = (0.5, 0.5, 0)$, When we apply our algorithm BL-PSO, we find: $(x^*, y^*) = (0.48, 0.54, 5.46 \times 10^{-5})$, with $\lambda = (0, 1)$.

Based on Examples 9 and 10, we conclude that our technique can be extended to semi-vectorial bi-level problems. However, this extension poses certain challenges that require further investigation and resolution.

6. Conclusion and future work

In this paper, we have proposed an effective method for solving bi-level programming problems using evolutionary algorithms. We demonstrated the effectiveness of our algorithm by applying it to several problems. Our approach is computationally efficient; unlike studies that solve the lower-level problem using classical methods and apply evolutionary algorithms to the upper-level problem (Zhao and Gu (2006), Jialin *et al*. (2016)), our method reduces the overall computational effort. We also compared the performance of two evolutionary algorithms: PSO and GWO. Additionally, we extended our approach to semi-vectorial bi-level programming problems, achieving satisfactory results for small-scale problems. In future work, we aim to develop an algorithm capable of handling large-scale semi-vectorial bi-level programming problems while also incorporating additional constraints into the lower-level problem.

References

Anandalingam, G., & White, D. (1990). A solution method for the linear static stackelberg problem using penalty functions. *IEEE*, 35, 1170-1173. doi: https://doi.org/10.1109/9.58565

Angelo, S., Krempser, E., & Barbosa, H. J. (2014). Differential evolution assisted by a surrogate model for bilevel programming problems. *2014 IEEE Congress on Evolutionary, Computation (CEC)*, 1784–1791. doi: https://doi.org/10.1109/CEC.2014.6900529

Bard, J. F., & Moore, J. T. (1990). A branch and bound algorithm for the bi-level programming problem. *SIAM Journal on Scientific and Statistical Computing*, 11, 128-290. doi: https://doi.org/10.1137/0911017

Butz, A. (1972). Solution of nonlinear equations with space filling curves. *Journal of Mathematical Analysis and Applications*, 37, 351-383. doi: https://doi.org/10.1016/0022-247X(72)90280-6

Dempe, S., & Dutta, J. (2012). Is bi-level programming a special case of a mathematical program with complementarity constraints?. *Mathematical Programming*, 131, 37-48. doi: https://doi.org/10.1007/s10107-010-0342-1

Dempe, S., & Franke, S. (2019). Solution of bi-level optimization problems using the kkt approach.*Optimization*, 68,1471-1489. doi: https://doi.org/10.1080/02331934.2019.1581192

Dempe, S., & Mehlitz, P.(2019). Semi-vectorial bi-level programming versus scalar bi-level programming. *Optimization*,157,657-679 .doi:https://doi.org/1 0.1080/023319 34.2019.1625 900

Dempe, S., & Zemkoho, A. (2013). New optimality conditions for the semi-vectorial bi-level optimization problem. *Journal of Optimization Theory and Applications*, 157, 54-74. doi: https://doi.org/10.1007/s10957-012-0161-z

Ehrgott, M. (2005). Multicriteria optimization. *Springer Science and Business Media*, 491 . doi: https://doi.org/10.1007/3-540-27659-9

Gupta, A., & Ong, Y. S. (2015). An evolutionary algorithm with adaptive scalarization for multi-objective bi-level programs. *2015 IEEE Congress on Evolutionary Computation (CEC)*, 1636-1642. https://doi:10.1109/CEC.2015.7257083

Jane, J. Y. (2005). Necessary and sufficient optimality conditions for mathematical programs with equilibrium constraints. *Journal of Mathematical Analysis and Applications*, 307, 350-369. doi: https://doi.org/10.1016/j.jmaa.2004.10.032

Jialin, H., Guangquan, Z., Yaoguang, H., & Jie, L. (2016). A solution to bi/tri-level programming problems using particle swarm optimization. *Information Sciences*, 370, 519-537. doi: https://doi.org/10.1016/j.ins.2016.08.022

Joao, A. M., & Paulo, C. J. (2014). An algorithm based on particle swarm optimization for multi-objective bi-level linear problems. *Applied Mathematics and Computation*, 247, 547-561. doi: https://doi.org/10.1016/j.amc.2014.09.013

João, A. M., Antunes, C. H., & Carrasqueira, P. (2015). A pso approach to semi-vectorial bi-level programming: pessimistic, optimistic and deceiving solutions. Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, 599-606. doi: https://doi.org/10.1145/2739480.2754644

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95-international conference on neural networks*, 4, 1942-1948. doi: https://doi.org/10.1109/ICNN.1995 .488968

Li, H., & L.Zhang. (2021). An efficient solution strategy for bi-level multi-objective optimization problems using multi-objective evolutionary algorithm. *Soft Computing*, 25, 8241-8261. doi: https://doi.org/10.1007/s00500-021-05750-0

Lin, G. H., Xu, M., & Ye, J. J. (2014). On solving simple bi-level programs with a non convex lower level program. *Mathematical Programming*, 144, 277-305. doi: https://doi.o rg/10.10 07/s10107-013-0633-4

Ma, L., & Wang, G. (2020). A solving algorithm for nonlinear bi-level programing problems based on human evolutionary model. Algorithms, 13, 260-272. doi: https://doi. org/10.3390/a13100260

Mathieu, R., Pittard, L., & Anandalingam, G. (1994). Genetic algorithm based approach to bi-level linear programming. *RAIRO-Operations Research*, 28, 1-21.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, 69, 46-61. doi: https://doi.org/10.1016/j.advengsoft.2013.12.007

Mitsos, A., & Barton, P. (2006). A test set for bi-level programs. Available at https://www.researchgate.net/publication/228455 291 .

Mora, G., & Cherruault, Y. (1997). Characterization and generation of _-dense curves. *Computers and Mathematics with Applications*, 33, 83-91. doi: https://doi.org/10.1016/S0898-1221(97)00067-9

Mora, G., & Mora-Porta, G. (2005). Dimensionality reducing multiple integrals by alpha-dense curves. *International Journal of Pure and Applied Mathematics*, 22, 103-114.

Nouri, A., Abdenacer, N., & Sahraoui, D. (2023). Accurate range-based distributed localization of wireless sensor nodes using grey wolf optimizer. *The Journal of Engineering and Exact Sciences*, 9, 15920–01e. doi: https://doi.org/10.18540/jcecvl9iss4pp15920-01e

Oduguwa, V., & Roy, R. (2002). Bi-level optimisation using genetic algorithm. *Proceedings 2002 IEEE International Conference on Artificial Intelligence Systems*, 322-327. doi: https://doi.org/10.1109/ICAIS.2002.1048121

Outrata, J. V. (1990). On the numerical solution of a class of stackelberg problems. *Zeitschrift für Operations Research*, 34, 255-277. doi: https://doi.org/10.1007/BF01416737

Ruuska, S., & Miettinen, K. (2012). Constructing evolutionary algorithms for bi-level multiobjective optimization. *2012 IEEE congress on evolutionary computation*, 1-7. doi: https://doi.org/10.1109/CEC.2012.6256156

Shimizu, K., & Aiyoshi, E. (1981). A new computational method for stackelberg and min-max problems by use of a penalty method. *IEEE Transactions on Automatic Control*, 26, 460-466. doi: https://doi.org/10.1109/TAC.1981.1102607

Srivastava, S., & Sahana, S. K. (2019). Application of bat algorithm for transport network design problem. *Applied Computational Intelligence and soft computing*, 2019, 9864090. doi: https://doi.org/10.1155/2019/9864090

Stephan, D., Joydeep, D., & Mordukhovich, B. (2007). New necessary optimality conditions in optimistic bi-level programming. *Optimization*, 56, 577-604. doi: https ://doi.org/ 10.108 0/ 02331930701617551

Xu, M., & Ye, J. J. (2014). A smoothing augmented lagrangian method for solving simple bi-level programs. *Computational Optimization and Applications*, 59, 353-377. doi: http s://do i.org/ 10.1007/s10589-013-9627-7

Ye, J. J., & Zhu, D. L. (1995). Optimality conditions for bi-level programming problems. *Optimization*, 33, 9-27. doi: https://doi.org/10.1080/02331939508844060

Zemkoho, A., & Zhou, S. (2021). Theoretical and numerical comparison of the karush–Kuhn tucker and value function reformulations in bi-level optimization. *Computational Optimization and Applications*, 78, 625-674. doi: https://doi.org/10.1007/s10589-020-00250-7

Zhang, T., Chen, Z., & Chen, J. (2017). A cooperative coevolution pso technique for complex bilevel programming problems and application to watershed water trading decision making problems. *Journal of Nonlinear Sciences and Applications(JNSA)*, 10. doi: https://doi.org/10 .224 36 /jns a .010.04.65

Zhao, Z., & Gu, X. (2006). Particle swarm optimization based algorithm for bi-level programming problems. *Sixth International Conference on Intelligent Systems Design and Applications*, 2, 951-956. doi: https://doi.org/10.1109/ISDA.2006.253740

Ziadi, R., & Bencherif-Madani, A. (2023). A mixed algorithm for smooth global optimization. *Journal of Mathematical Modeling*, 11, 207-228. doi: http s://do i.org/10. 22124/JM M.202 2.23133.2061