Research Article

Solving Bilevel Multiobjective Programming Problem by Elite Quantum Behaved Particle Swarm Optimization

Tao Zhang,1, 2 Tiesong Hu,1 Jia-wei Chen,3 Zhongping Wan,3 and Xuning Guo1

1 State Key Laboratory of Water Resource and Hydropower Engineering Science, Wuhan University, Wuhan 430072, China
2 School of Information and Mathematics, Yangtze University, Jingzhou 434023, China
3 School of Mathematics and Statistics, Wuhan University, Wuhan 430072, China

Correspondence should be addressed to Jia-wei Chen, j.w.chen713@163.com

Received 25 August 2012; Accepted 18 October 2012

Copyright © 2012 Tao Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

An elite quantum behaved particle swarm optimization (EQPSO) algorithm is proposed, in which an elite strategy is exerted for the global best particle to prevent premature convergence of the swarm. The EQPSO algorithm is employed for solving bilevel multiobjective programming problem (BLMPP) in this study, which has never been reported in other literatures. Finally, we use eight different test problems to measure and evaluate the proposed algorithm, including low dimension and high dimension BLMPPs, as well as attempt to solve the BLMPPs whose theoretical Pareto optimal front is not known. The experimental results show that the proposed algorithm is a feasible and efficient method for solving BLMPPs.

1. Introduction

Bilevel programming problem (BLPP) arises in a wide variety of scientific and engineering applications including optimal control, process optimization, game-playing strategy development, transportation problem, and so on. Thus the BLPP has been developed and researched by many scholars. The reviews, monographs, and surveys on the BLPP can refer to [1–11]. Moreover, the evolutionary algorithms (EAs) have been employed to address BLPP in papers [12–16].

For the multiobjective characteristics widely existing in the BLPP, the bilevel multiobjective programming problem (BLMPP) has attracted many researchers to study
it. For example, Shi and Xia [17, 18], Abo-Sinna and Baky [19], Nishizaki and Sakawa [20], Zheng et al. [21] presented an interactive algorithm for BLMPP. Eichfelder [22] presented a method for solving nonlinear bilevel multiobjective optimization problems with coupled upper level constraints. Thereafter, Eichfelder [23] developed a numerical method for solving nonlinear nonconvex bilevel multiobjective optimization problems. In recent years, the metaheuristic has attracted considerable attention as an alternative method for BLMPP. For example, Deb and Sinha [24–26], as well as Sinha and Deb [27] discussed BLMPP based on evolutionary multiobjective optimization principles. Based on those studies, Deb and Sinha [28] proposed a viable and hybrid evolutionary-local-search based algorithm and presented challenging test problems. Sinha [29] presented a progressively interactive evolutionary multiobjective optimization method for BLMPP.

Particle swarm optimization (PSO) is a relatively novel heuristic algorithm inspired by the choreography of a bird flock, which has been found to be quite successful in a wide variety of optimization tasks [30]. Due to its high speed of convergence and relative simplicity, the PSO algorithm has been employed by many researchers for solving BLPPs. For example, Li et al. [31] proposed a hierarchical PSO for solving BLPP. Kuo and Huang [32] applied the PSO algorithm for solving bilevel linear programming problem. Gao et al. [33] presented a method to solve bilevel pricing problems in supply chains using PSO. However, it is worth noting that the papers mentioned above only for bilevel single objective problems and the BLMPP have seldom been studied using PSO so far. There are probably two reasons for this situation. One reason is that the added complexities associated with solving each level, and the other reason is that the global convergence of the PSO cannot be guaranteed [34].

In this paper, a global convergence guaranteed method called as EQPSO is proposed, in which an elite strategy is exerted for global best particle to prevent premature convergence of the swarm. The EQPSO is employed for solving the BLMPP in this study, which has not been reported in other literatures. For such problems, the proposed algorithm directly simulates the decision process of the bilevel programming, which is different from most traditional algorithms designed for specific versions or based on specific assumptions. The BLMPP is transformed to solve multiobjective optimization problems in the upper level and the lower level interactively by the EQPSO. And a set of approximate Pareto optimal solutions for BLMPP are obtained using the elite strategy. This interactive procedure is repeated until the accurate Pareto optimal solutions of the original problem are found. The rest of the paper is organized as follows. In Section 2, the problem formulation is provided. The proposed algorithm for solving bilevel multiobjective problem is presented in Section 3. In Section 4, some numerical examples are given to demonstrate the feasibility and efficiency of the proposed algorithm.

2. Problem Formulation

Let $x \in \mathbb{R}^{n_1}$, $y \in \mathbb{R}^{n_2}$, $F : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{m_1}$, $f : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{m_2}$, $G : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^p$, $g : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^l$. The general model of the BLMPP can be written as follows:

$$
\min_x F(x, y) \\
\text{s.t. } G(x, y) \geq 0,
$$
\[
\begin{align*}
\min_y f(x,y) \\
\text{s.t. } g(x,y) &\geq 0,
\end{align*}
\]

(2.1)

where \(F(x,y)\) and \(f(x,y)\) are the upper level and the lower level objective functions, respectively. \(G(x,y)\) and \(g(x,y)\) denote the upper level and the lower level constraints, respectively.

Let \(S = \{(x, y) \mid G(x, y) \geq 0, g(x, y) \geq 0\}, X = \{x \mid \exists y, G(x, y) \geq 0, g(x, y) \geq 0\}, S(x) = \{y \mid g(x, y) \geq 0\}\), and for the fixed \(x \in X\), let \(\overline{S}(X)\) denote the weak efficiency set of solutions to the lower level problem, the feasible solution set of problem (2.1) is denoted as: IR = \{(x, y) \mid (x, y) \in S, y \in \overline{S}(X)\}.

**Definition 2.1.** For a fixed \(x \in X\), if \(y\) is a Pareto optimal solution to the lower level problem, then \((x, y)\) is a feasible solution to the problem (2.1).

**Definition 2.2.** If \((x^*, y^*)\) is a feasible solution to the problem (2.1), and there are no \((x, y) \in \text{IR}\), such that \(F(x, y) \prec F(x^*, y^*)\), then \((x^*, y^*)\) is a Pareto optimal solution to the problem (2.1), where “\(<\)” denotes Pareto preference.

For problem (2.1), it is noted that a solution \((x^*, y^*)\) is feasible for the upper level problem if and only if \(y^*\) is an optimal solution for the lower level problem with \(x = x^*\). In practice, we often make the approximate Pareto optimal solutions of the lower level problem as the optimal response feedback to the upper level problem, and this point of view is accepted usually. Based on this fact, the EQPSO algorithm may have a great potential for solving BLMPP. On the other hand, unlike the traditional point-by-point approach mentioned in Section 1, the EQPSO algorithm uses a group of points in its operation, thus the EQPSO can be developed as a new way for solving BLMPP. We next present the algorithm based on the EQPSO is presented for (2.1).

### 3. The Algorithm

#### 3.1. The EQPSO

The quantum behaved particle swarm optimization (QPSO) is the integration of PSO and quantum computing theory developed by [35–38]. Compared with PSO, it needs no velocity vectors for particles and has fewer parameters to adjust. Moreover, its global convergence can be guaranteed [39]. Due to its global convergence and relative simplicity, it has been found to be quite successful in a wide variety of optimization tasks. For example, a wide range of continuous optimization problems [40–45] are solved by QPSO and the experiment results show that the QPSO works better than standard PSO. Some improved QPSO algorithms can refer to [46–48]. In this paper, the EQPSO algorithm is proposed, in which an elite strategy is exerted for global best particle to prevent premature convergence of the swarm, and it makes the proposed algorithm has good performance for solving the high dimension BLMPPS. The EQPSO has the same design principle with the QPSO except for the global optimal particle selection criterion, so the global convergence proof of the
EQPSO can refer to [39]. In the EQPSO, the particles move according to the following iterative equation:

\[
\begin{align*}
  z_{t+1}^t &= p^t - \alpha^t (m_{\text{Best}}^t - z_t^t) \cdot I_u \left( \frac{1}{u} \right) \quad \text{if } k \geq 0.5, \\
  z_{t+1}^t &= p^t + \alpha^t (m_{\text{Best}}^t - z_t^t) \cdot I_u \left( \frac{1}{u} \right) \quad \text{if } k < 0.5,
\end{align*}
\]

(3.1)

where

\[
\begin{align*}
  p^t &= \varphi^t p_{p_{\text{Best}}^t}^t + (1 - \varphi^t) p_{g_{\text{Best}}^t}^t, \\
  m_{\text{Best}}^t &= \frac{1}{N} \sum_{i=1}^{N} p_{p_{\text{Best}}^t}^i, \\
  \alpha^t &= m - (m - n) \cdot \frac{t}{T}, \\
  p_{g_{\text{Best}}^t} &= \text{rand}(A_t),
\end{align*}
\]

(3.2)

where the \( z \) denotes the particle’s position. \( m_{\text{Best}} \) denotes the mean best position of all the particles’ best positions. The \( k, u, \) and \( \varphi \) are random numbers distributed uniformly on \((0, 1)\), respectively. \( \alpha(t) \) is the expansion-contraction coefficient. In general, \( m = 1, n = 0.5, t \) is the current iteration number, and \( T \) is the maximum number of iterations. The \( p_{p_{\text{Best}}} \) and \( p_{g_{\text{Best}}} \) are the particle’s personal best position and the global best position, respectively. \( A_t \) is the elite set which is introduced in following parts (see Algorithm: Step 3).

### 3.2. The Algorithm for Solving BLMPP

The process of the proposed algorithm for solving BLMPP is an interactive coevolutionary process. We first initialize population and then solve multiobjective optimization problems in the upper level and the lower level interactively using the EQPSO. For one time of iteration, a set of approximate Pareto optimal solutions for problem 1 is obtained by the elite strategy which was adopted in Deb et al. [49]. This interactive procedure is repeated until the accurate Pareto optimal solutions of problem (2.1) are found. The details of the proposed algorithm are given as follows.

**Algorithm**

**Step 1. Initializing.**

**Step 1.1.** Initialize the population \( P_0 \) with \( N_u \) particles which is composed by \( n_s = N_u / N_l \) subswarms of size \( N_l \) each. The particle’s position of the \( k \)-th \((k = 1, 2, \ldots, n_s)\) subswarm is presented as: \( z_j = (x_j, y_j) \) \((j = 1, 2, \ldots, n_l)\) and \( z_j \) is sampled randomly in the feasible space.

**Step 1.2.** Initialize the external loop counter \( t := 0 \).
Step 2. For the $k$th subswarm, $(k = 1, 2, \ldots, n_s)$, each particle is assigned a nondomination rank $ND_l$ and a crowding value $CD_l$ in $f$ space. Then, all resulting subswarms are combined into one population which is named as the $P_l$. Afterwards, each particle is assigned a nondomination rank $ND_u$ and a crowding value $CD_u$ in $F$ space.

Step 3. The nondomination particles assigned both $ND_u = 1$ and $ND_l = 1$ from $P_l$ are saved in the elite set $A_l$.

Step 4. For the $k$th subswarm, $(k = 1, 2, \ldots, n_s)$, update the lower level decision variables.

Step 4.1. Initialize the lower level loop counter $t_l := 0$.

Step 4.2. Update the $j$th ($j = 1, 2, \ldots, N_l$) particle’s position with the fixed $x_j$ according to (3.1) and (3.2).

Step 4.3. $t_l := t_l + 1$.

Step 4.4. If $t_l \geq T_l$, go to Step 4.5. Otherwise, go to Step 4.2

Step 4.5. Each particle of the $i$th subswarm is reassigned a nondomination rank $ND_l$ and a crowding value $CD_l$ in $f$ space. Then, all resulting subswarms are combined into one population which is renamed as the $Q_l$. Afterwards, each particle is reassigned a nondomination rank $ND_u$ and a crowding value $CD_u$ in $F$ space.

Step 5. Combined population $P_l$ and $Q_l$ to form $R_l$. The combined population $R_l$ is reassigned a nondomination rank $ND_u$, and the particles within an identical nondomination rank are assigned a crowding distance value $CD_u$ in the $F$ space.

Step 6. Choose half particles from $R_l$. The particles of rank $ND_u = 1$ are considered first. From the particles of rank $ND_u = 1$, the particles with $ND_l = 1$ are noted one by one in the order of reducing crowding distance $CD_u$ for each such particle the corresponding subswarm from its source population (either $P_l$ or $Q_l$) is copied in an intermediate population $S_l$. If a subswarm is already copied in $S_l$ and a future particle from the same subswarm is found to have $ND_u = ND_l = 1$, the subswarm is not copied again. When all particles of $ND_u = 1$ are considered, a similar consideration is continued with $ND_u = 2$ and so on till exactly $n_s$ subswarms are copied in $S_l$.

Step 7. Update the elite set $A_l$. The nondomination particles assigned both $ND_u = 1$ and $ND_l = 1$ from $S_l$ are saved in the elite set $A_l$.

Step 8. Update the upper level decision variables in $S_l$.

Step 8.1. Initiate the upper level loop counter $t_u := 0$.

Step 8.2. Update the $i$th ($i = 1, 2, \ldots, N_u$) particle’s position with the fixed $y_i$ according to (3.1) and (3.2).

Step 8.3. $t_u := t_u + 1$.

Step 8.4. If $t_u \geq T_u$, go to Step 8.5. Otherwise, go to Step 8.2.

Step 8.5. Every member is then assigned a nondomination rank $ND_u$ and a crowding distance value $CD_u$ in $F$ space.
Step 9. $t := t + 1$.

Step 10. If $t \geq T$, output the elite set $A_t$. Otherwise, go to Step 2.

In Steps 4 and 8, the global best position is chosen at random from the elite set $A_t$. The criterion of personal best position choice is that, if the current position is dominated by the previous position, then the previous position is kept; otherwise, the current position replaces the previous one; if neither of them is dominated by the other, then we select one of them randomly. A relatively simple scheme is used to handle constraints. Whenever two individuals are compared, their constraints are checked. If both are feasible, nondomination sorting technology is directly applied to decide which one is selected. If one is feasible and the other is infeasible, the feasible dominates. If both are infeasible, then the one with the lowest amount of constraint violation dominates the other. Notations used in the proposed algorithm are detailed in Table 1.

4. Numerical Experiment

In this section, three examples will be considered to illustrate the feasibility of the proposed algorithm for problem (2.1). In order to evaluate the closeness between the obtained Pareto optimal front and the theoretical Pareto optimal front, as well as the diversity of the obtained Pareto optimal solutions along the theoretical Pareto optimal front, we adopted the following evaluation metrics.

4.1. Performance Evaluation Metrics

(a) Generational Distance (GD): this metric used by Deb [50] is employed in this paper as a way of evaluating the closeness between the obtained Pareto optimal front and the theoretical Pareto optimal front. The GD metric denotes the average distance between the obtained Pareto optimal front and the theoretical Pareto optimal front:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n},$$

where $n$ is the number of the obtained Pareto optimal solutions by the proposed algorithm and $d_i$ is the Euclidean distance between each obtained Pareto optimal solution and the nearest member of the theoretical Pareto optimal set.

(b) Spacing (SP): this metric is used to evaluate the diversity of the obtained Pareto optimal solutions by comparing the uniform distribution and the deviation of solutions as described by Deb [50]:

$$SP = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{n} (\bar{d} - d_i)^2}{\sum_{m=1}^{M} d_m^e + nd},$$

where $d_i = \min_j(|F_i^1(x, y) - F_j^1(x, y)| + |F_i^2(x, y) - F_j^2(x, y)|)$, $i, j = 1, 2, \ldots, n$, $\bar{d}$ is the mean of all $d_i$, $d_m^e$ is the Euclidean distance between the extreme solutions in obtained Pareto optimal solution set and the theoretical Pareto optimal solution set on the $m$th objective, $M$ is the
number of the upper level objective function, and \( n \) is the number of the obtained solutions by the proposed algorithm.

All results presented in this paper have been obtained on a personal computer (CPU: AMD Phenon II X6 1055T 2.80 GHz; RAM: 3.25 GB) using a C# implementation of the proposed algorithm.

### 4.2. Numerical Examples

#### 4.2.1. Low Dimension BLMPPs

**Example 4.1.** Example 4.1 is taken from [22]. Here \( x \in \mathbb{R}^1, y \in \mathbb{R}^2 \). In this example, the population size and iteration times are set as follows: \( N_u = 200, T_u = 200, N_l = 40, T_l = 40, \) and \( T = 40 \):

\[
\begin{align*}
\min_x F(x, y) &= (y_1 - x, y_2) \\
\text{s.t.} \quad G_1(y) &= 1 + y_1 + y_2 \geq 0 \\
\min_y f(x, y) &= (y_1, y_2) \\
\text{s.t.} \quad g_1(x, y) &= x^2 - y_1^2 - y_2^2 \geq 0, \\
&\quad -1 \leq y_1, y_2 \leq 1, \quad 0 \leq x \leq 1.
\end{align*}
\]

(4.3)

Figure 1 shows the obtained Pareto front of this example by the proposed algorithm. From Figure 1, it can be seen that the obtained Pareto front is very close to the theoretical
Pareto optimal front, and the average distance between the obtained Pareto optimal front and the theoretical Pareto optimal front is 0.00024, that is, GD = 0.00024 (see Table 2). Moreover, the lower SP value (SP = 0.0042, see Table 2) shows that the proposed algorithm is able to obtain a good distribution of solutions on the entire range of the theoretical Pareto optimal front. Figure 2 shows the obtained solutions of this example, which follow the relationship, that is, $y_1 = -1 - y_2$, $y_2 = -1/2 \pm (1/4)\sqrt{8x^2 - 4}$ and $x \in (1/\sqrt{2}, 1)$. It is also obvious that all obtained solutions are close to being on the upper level constraint $G(x)$ boundary ($1 + y_1 + y_2 = 0$).

Example 4.2. Example 4.2 is taken from [51]. Here $x \in R^1$, $y \in R^2$. In this example, the population size and iteration times are set as follows: $N_u = 200$, $T_u = 50$, $N_l = 40$, $T_l = 20$, and $T = 40$:

$$
\begin{align*}
\min_x & \quad F(x, y) = \left( x^2 + (y_1 - 1)^2 + y_2^2, (x - 1)^2 + (y_1 - 1)^2 + y_2^2 \right) \\
\min_y & \quad f(x, y) = \left( y_1^2 + y_2^2, (y_1 - x)^2 + y_2^2 \right) \\
& \quad -1 \leq x, \quad y_1, y_2 \leq 2.
\end{align*}
$$

Figure 3 shows the obtained Pareto optimal front of this example by the proposed algorithm. From Figure 3, it is obvious that the obtained Pareto optimal front is very close to the theoretical Pareto optimal front, the average distance between the obtained Pareto optimal front and the theoretical Pareto optimal front is 0.00003 (see Table 2). On the other hand, the obtained Pareto optimal solutions can distribute uniformly on entire range of theoretical Pareto optimal front base on the fact that the SP value is lower ($SP = 0.00169$, see Table 2).
The obtained Pareto optimal solutions
The theoretical Pareto optimal solutions

**Figure 2:** The obtained solutions of Example 4.1.

The obtained Pareto optimal front
The theoretical Pareto optimal front

**Figure 3:** The obtained Pareto optimal front of Example 4.2.

**Figure 4** shows the obtained Pareto optimal solutions, they follow the relationship, that is, $x = y_1$, $y_1 \in [0.5, 1]$ and $y_2 = 0$. 
4.2.2. High Dimension BLMPPs

Example 4.3. Example 4.3 is taken from [28]. Here $x \in \mathbb{R}^{10}$, $y \in \mathbb{R}^{10}$. In this example, the population size and iteration times are set as follows: $N_u = 400$, $T_u = 50$, $N_l = 40$, $T_l = 20$, and $T = 60:

$$\min_x F(x, y) = \left( 1 + r - \cos(\alpha \pi x_1) \right) + \sum_{j=2}^{K} \left( x_j - \frac{j - 1}{2} \right)^2 + \tau \sum_{i=2}^{K} (y_i - x_i)^2 - \gamma \cos \left( \frac{\pi x_1}{2y_1} \right),$$

$$+ \tau \sum_{j=2}^{K} (y_j - x_j)^2 - \gamma \sin \left( \frac{\pi x_1}{2y_1} \right)$$

$$\min_y f(x, y) = \left( y_1^2 + \sum_{i=2}^{K} (y_i - x_i)^2 + \frac{10}{1 - \cos \left( \frac{\pi}{K} (y_i - x_i) \right)} \right),$$

$$\sum_{i=1}^{K} (y_i - x_i)^2 + \sum_{i=2}^{K} 10 \left| 1 - \sin \left( \frac{\pi}{K} (y_i - x_i) \right) \right|$$

s.t.  

$-K \leq y_i \leq K, \quad (i = 1, 2, \ldots, K);$

$1 \leq x_i \leq 4, \quad -K \leq x_j \leq K, \quad (j = 2, 3, \ldots, K).$

$\alpha = 1, \quad r = 0.1, \quad \tau = 1, \quad \gamma = 1, \quad K = 10.$

(4.5)
Example 4.4. Example 4.4 is taken from [28]. Here $x \in \mathbb{R}^{10}$, $y \in \mathbb{R}^{10}$. In this example, the population size and iteration times are set as follows: $N_u = 400$, $T_u = 50$, $N_t = 40$, $T_t = 20$, and $T = 80$.

$$\min_x F(x, y) = v_1(x_1) + \sum_{j=2}^{K} y_j^2 + 10\left(1 - \cos\left(\frac{\pi}{K} y_1\right)\right) + \tau \sum_{i=2}^{K} \left(y_i - x_i\right)^2 - r \cos\left(\frac{\pi x_1}{2y_1}\right),$$

$$v_2(x_1) + \sum_{j=2}^{K} y_j^2 + 10\left(1 - \cos\left(\frac{\pi}{K} y_1\right)\right)$$

$$+ \tau \sum_{i=2}^{K} \left(y_i - x_i\right)^2 - r \sin\left(\frac{\pi x_1}{2y_1}\right),$$

$$\min_y f(x, y) = \left(y_1^2 + \sum_{i=2}^{K} (y_i - x_i)^2, \sum_{i=1}^{K} i(y_i - x_i)^2\right)$$

s.t. $-K \leq y_i \leq K$, $x_i \geq 0$, $y_i \geq 0$, $x_i \geq 0$, $y_i \geq 0$,

$$0.001 \leq x_1 \leq 4, \quad -K \leq x_j \leq K, \quad (j = 2, 3, \ldots, K).$$

$$a = 1, \quad r = 0.25, \quad \tau = 1, \quad \gamma = 4, \quad K = 10,$$

(4.6)

where

$$v_1(x_1) = \begin{cases} 
\cos(0.2\pi)x_1 + \sin(0.2\pi)\sqrt{[0.02\sin(5\pi x_1)]}, & \text{for } 0 \leq x_1 \leq 1, \\
x_1 - (1 - \sin(0.2\pi)), & \text{for } x_1 > 1,
\end{cases}$$

(4.7)

$$v_2(x_1) = \begin{cases} 
-\sin(0.2\pi)x_1 + \cos(0.2\pi)\sqrt{[0.02\sin(5\pi x_1)]}, & \text{for } 0 \leq x_1 \leq 1, \\
0.1(x_1 - 1) - \sin(0.2\pi), & \text{for } x_1 > 1.
\end{cases}$$

This problem is more difficult compared to the previous problems (Examples 4.1 and 4.2) because the lower level problem of this example has multimodalities, thereby making the lower level problem difficult in finding the upper level Pareto optimal front. From Figure 5, it can be seen that the obtained Pareto front is very close to the theoretical Pareto optimal front, and the average distance between the obtained Pareto optimal front and the theoretical Pareto optimal front is 0.00027, that is, $GD = 0.00027$ (see Table 2). Moreover, the lower SP value ($SP = 0.00127$, see Table 2) shows that the proposed algorithm is able to obtain a good distribution of solutions on the entire range of the theoretical Pareto optimal front. Furthermore, two obtained lower level Pareto optimal fronts are given when $x_1 = 2$ and $x_1 = 2.5$.

Figure 6 shows the obtained Pareto front of Example 4.4 by the proposed algorithm. The upper level problem has multimodalities, thereby causing an algorithm difficulty in finding the upper level Pareto optimal front. From Figure 6, it can be seen that the obtained Pareto front is very close to the theoretical Pareto optimal front, and the average distance between the obtained Pareto optimal front and the theoretical Pareto optimal front is 0.00036, that is, $GD = 0.00036$ (see Table 2). Moreover, the lower SP value ($SP = 0.00235$, see Table 2)
The obtained Pareto optimal solutions
The theoretical Pareto optimal solutions

Figure 5: The obtained Pareto front of Example 4.3.

The theoretical Pareto optimal front of lower level problem
The obtained upper level Pareto optimal front

Figure 6: The obtained Pareto front of Example 4.4.
Example 4.5. Example 4.5 is taken from [28]. Here $x \in R^{10}$, $y \in R^5$. In this example, the population size and iteration times are set as follows: $N_u = 400$, $T_u = 50$, $N_l = 40$, $T_l = 20$, and $T = 60$:

$$
\min_x F(x \cdot y) = \left( x_1 + \sum_{j=3}^{K} \left( x_j - \frac{j}{2} \right)^2 + \tau \sum_{i=3}^{K} (y_i - x_i)^2 - \cos \left( 4 \tan^{-1} \left( \frac{x_2 - y_2}{x_1 - y_1} \right) \right) \right),
$$

$$
x_2 + \sum_{j=3}^{K} \left( x_j - \frac{j}{2} \right)^2 + \tau \sum_{i=3}^{K} (y_i - x_i)^2 - \cos \left( 4 \tan^{-1} \left( \frac{x_2 - y_2}{x_1 - y_1} \right) \right)
$$

s.t. $G(x) = x_2 - \left( 1 - x_1^2 \right)^2 \geq 0$

$$
\min_y f(x, y) = \left( y_1 + \sum_{i=3}^{K} (y_i - x_i)^2, \ y_2 + \sum_{i=3}^{K} (y_i - x_i)^2 \right)
$$

s.t. $g_1(x, y) = (y_1 - x_1)^2 + (y_2 - x_2)^2 \leq r^2$

$$
R(x_1) = (0.1 + 0.15|\sin(2\pi(x_1 - 0.1))|),
$$

$-K \leq y_i \leq K, \ (i = 1, 2, \ldots, K);$  

$0 \leq x_1 \leq K, \ -K \leq x_j \leq K, \ (j = 2, 3, \ldots, K).$

$\tau = 1, \ r = 0.2, \ K = 10.$

Example 4.6. Example 4.6 is taken from [28]. Here $x \in R^4$, $y \in R^5$. In this example, the population size and iteration times are set as follows: $N_u = 400$, $T_u = 50$, $N_l = 40$, $T_l = 20$, and $T = 40$:

$$
\min_x F(x, y) = \left( (1 - y_1) \left( 1 + \sum_{j=2}^{K} y_j^2 \right) x_1, \ y_1 \left( 1 + \sum_{j=2}^{K} y_j^2 \right) x_1 \right)
$$

s.t. $G_1(x, y) = -(1 - y_1)x_1 - \frac{1}{2}x_1y_1 \leq 1$
The obtained upper level Pareto optimal front

The theoretical Pareto optimal front of lower level problem

The obtained upper level Pareto optimal front

Figure 7: The obtained Pareto front of Example 4.5.

min \ y \ f(x,y) = \left( 1 - y_1 \right) \left( 1 + \sum_{j=K+1}^{K+L} y_j^2 \right) x_1, \ y_1 \left( 1 + \sum_{j=K+1}^{K+L} y_j^2 \right) x_1 \\

s.t. \ 1 \leq x_1 \leq 2, \ -1 \leq y_1 \leq 1, \ \\
\ - (K + L) \leq y_j \leq (K + L), \ (j = 2, 3, \ldots, K + L), \ K = 5, \ L = 4.

(4.9)

Figure 7 shows the obtained Pareto front of Example 4.5 by the proposed algorithm. From Figure 7, it can be seen that the obtained Pareto front is very close to the theoretical Pareto optimal front, and the average distance between the obtained Pareto optimal front and the theoretical Pareto optimal front is 0.00058, that is, GD = 0.00058 (see Table 2). Moreover, the lower SP value (SP = 0.00364, see Table 2) shows that the proposed algorithm is able to obtain a good distribution of solutions on the entire range of the theoretical Pareto optimal front. Furthermore, all obtained lower level Pareto optimal fronts are given. It is also obvious that the Pareto optimal fronts for both the lower and upper level lie on constraint boundaries and every lower level Pareto optimal front has an unequal contribution to the upper level Pareto optimal front.

Figure 8 shows the obtained Pareto front of Example 4.6 by the proposed algorithm. From Figure 8, it can be seen that the obtained Pareto front is very close to the theoretical Pareto optimal front, and the average distance between the obtained Pareto optimal front and the theoretical Pareto optimal front is 0.00039, that is, GD = 0.00039 (see Table 2). Moreover, the lower SP value (SP = 0.00168, see Table 2) shows that the proposed algorithm is able to obtain a good distribution of solutions on the entire range of the theoretical Pareto optimal front. Furthermore, three obtained lower level Pareto optimal fronts are given when $y_1 = 1$,
The obtained upper level Pareto optimal front

The theoretical Pareto optimal front

The theoretical lower level Pareto optimal front when $y_1 = 1.5$

The theoretical lower level Pareto optimal front when $y_1 = 2$

Figure 8: The obtained Pareto front of Example 4.6.

$y_1 = 1.5$ and $y_1 = 2$. It can be seen that only one Pareto optimal point from each participating lower level problem qualifies to be on the upper level Pareto optimal front.

4.2.3. The BLMPPs with Unknown Theoretical Pareto Optimal Fronts

Example 4.7. Example 4.7 is taken from [52], in which the theoretical Pareto optimal front is not given. Here $x \in \mathbb{R}^2$, $y \in \mathbb{R}^3$. In this example, the population size and iteration times are set as follows: $N_u = 100$, $T_u = 50$, $N_l = 20$, $T_l = 10$, and $T = 40$:

$$
\max_x F(x, y) = (x_1 + 9x_2 + 10y_1 + y_2 + 3x_3, 9x_1 + 2x_2 + 2y_1 + 7y_2 + 4x_3)
$$

s.t. $G_1(x, y) = 3x_1 + 9x_2 + 9y_1 + 5y_2 + 3y_3 \leq 1039$

$G_2(x, y) = -4x_1 - x_2 + 3y_1 - 3y_2 + 2y_3 \leq 94$

$$
\min_y f(x, y) = (4x_1 + 6x_2 + 7y_1 + 4y_2 + 8y_3, 6x_1 + 4x_2 + 8y_1 + 7y_2 + 4y_3)
$$

s.t. $g_1(x, y) = 3x_1 - 9x_2 - 9y_1 - 4y_2 \leq 61$

$g_2(x, y) = 5x_1 + 9x_2 + 10y_1 - y_2 - 2y_3 \leq 924$

$g_3(x, y) = 3x_1 - 3x_2 + y_2 + 5y_3 \leq 420$

$x_1, x_2, y_1, y_2, y_3 \geq 0.$ (4.10)
Example 4.8. Example 4.8 is taken from [23]. Here $x \in \mathbb{R}^1$, $y \in \mathbb{R}^2$. In this example, the population size and iteration times are set as follows: $N_u = 800$, $T_u = 50$, $N_l = 40$, $T_l = 20$, and $T = 40$:

$$
\min_x F(x, y) = \left( x + y_1^2 + y_2^2 + \sin^2(x_1 + y), \cos(y_2)(0.1 + y) \left( \exp \left( -\frac{y_1}{0.1 + y_2} \right) \right) \right)
$$

s.t. $G_1(x, y) = (x - 5)^2 - (y_1 - 0.5)^2 - (y_2 - 5)^2 \leq 16$

$$
\min_y f(x, y) = \left( \frac{(y_1 - 2)^2 + (y_2 - 2)^2}{4} + \frac{xy_2 + (5 - x)^2}{16} \right)
+ \sin \left( \frac{y_2}{10} \right), \frac{y_1^2 + (y_2 - 6)^2 - 2xy_1 - (5 - x)^2}{80}
$$

s.t. $g_1(x, y) = y_1^2 - y_2 \leq 0$

$g_2(x, y) = 5y_1^2 + y_2 - 10 - \leq 0$

$g_3(x, y) = y_2 - 5 - \frac{x}{6} \leq 0$

$0 \leq x \leq 10, \quad 0 \leq y_1, \quad y_2 \leq 10.$

Figure 9 shows the obtained Pareto optimal front of Example 4.7 by the proposed algorithm. Note that, Zhang et al. [52] only obtained a single optimal solution $x = (146.2955, 28.9394)$ and $y = (0, 0.679318, 0)$ which lies on the maximum of the $F_2$ using weighted sum method. In contrast, a set of Pareto optimal solutions are obtained by the proposed algorithm. However, the fact that the single optimal solution in [49] is included in the obtained Pareto optimal solutions illustrates the feasibility of proposed algorithm. Figure 10 shows the final archive solutions of the Example 4.8 by the proposed algorithm. For this problem, the exact Pareto optimal front is not known, but the obtained Pareto optimal front by the proposed algorithm is similar to that reported in the previous study [23].

5. Conclusion

In this paper, an EQPSO is presented, in which an elite strategy is exerted for global best particle to prevent the swarm from clustering, enabling the particle to escape the local optima. The EQPSO algorithm is employed for solving bilevel multiobjective programming problem (BLMPP) for the first time. In this study, some numerical examples are used to explore the feasibility and efficiency of the proposed algorithm. The experimental results indicate that the obtained Pareto front by the proposed algorithm is very close to the theoretical Pareto optimal front, and the solutions are also distributed uniformly on entire range of the theoretical Pareto optimal front. The proposed algorithm is simple and easy to implement, which provides another appealing method for further study on BLMPP.
The obtained Pareto optimal front by the proposed algorithm
The single optimal solution in [37]

Figure 9: The obtained front of Example 4.7.

Figure 10: The obtained front of Example 4.8.

Acknowledgments

The authors are indebted to the referees and the associate editor for their insightful and pertinent comments. This work is supported by the National Science Foundation of China (71171150, 71171151, 50979073, 61273179, 11201039, 20101304), the Academic Award for Excellent Ph.D. Candidates Funded by Wuhan University and the Fundamental Research Fund for the Central Universities (no. 201120102020004), and the Ph.D. short-time mobility program by Wuhan University.

References

Abstract and Applied Analysis


Submit your manuscripts at http://www.hindawi.com