BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **17**, No 3
Sofia • 2017 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.1515/cait-2017-0030

Particle Swarm Optimization Based on Smoothing Approach for Solving a Class of Bi-Level Multiobjective Programming Problem

Qingping He, Yibing Lv

School of Information and Mathematics, Yangtze University, Jingzhou 434023, China
E-mails: Heqp_2017@163.com lvyibing_2001@163.com

Abstract**:** *As a metaheuristic, Particle Swarm Optimization* (*PSO*) *has been used to solve the Bi-level Multiobjective Programming Problem* (*BMPP*). *However, in the existing solving approach based on PSO for the BMPP, the upper level and the lower level problem are solved interactively by PSO. In this paper, we present a different solving approach based on PSO for the BMPP. Firstly, we replace the lower level problem of the BMPP with Kuhn-Tucker optimality conditions and adopt the perturbed Fischer-Burmeister function to smooth the complementary conditions. After that, we adopt PSO approach to solve the smoothed multiobjective programming problem. Numerical results show that our solving approach can obtain the Pareto optimal front of the BMPP efficiently.*

Keywords**:** *Bi-level multiobjective programming problem, scalarization method, optimality conditions, smoothing method, particle swarm optimization.*

1. Introduction

Bi-level programming problem is a nested optimization problem, which is characterized by the existence of two optimization problems in which the constraint region of the upper level optimization problem (the leader) contains another optimization problem called the lower level optimization problem (the follower)[1]. As a powerful tool to describe the hierarchy existing in the real life problem, bi-level programming has been applied wildly in such areas as transportation systems [2], inverse optimization value problem [3], transportation network design [4], etc. Meanwhile, the wide application of the bi-level programming drives more and more researchers to devote to this promising field. In fact, in the last two decades, many papers have been published both from the theoretical and computational points of view (see the monographs of D e m p e [1] and the reviews by D e m p e and

Zemkoho [5], Sinha, Malo and Deb [6], and Colson, Marcotte and Savard [7]).

Bi-level Multiobjective Programming Problem (BMPP), where the leader, the follower, or the both have multiple conflicting objectives, has successively drawn the researchers' attention both from the theoretical and computational points of view. Here, we can roughly classify the feasible algorithms for the BMPP into two types. The first one is the traditional numerical approach and the second one is the evolutionary approach.

Focusing on the traditional numerical approach for the linear BMPP in which both the leader and the follower have multiple objectives, Nishizaki and Sakawa [8] give the three optimal solution definitions. They propose a feasible algorithm based on K-th best algorithm for the linear bi-level single objective programming problem. For the given lower level objective weights, Lv [9] replaces the lower level optimization problem with its optimality conditions and constructs a multiobjective penalized optimization problem. Then, an exact penalized function approach is proposed and some numerical results are reported; subsequently, following the outline in [9] and regarding the lower level objective weights as the upper level decision variables, Lv and Wan [10] propose another exact penalty function approach. For the linear BMPP, where the leader has single objective and the follower has multiple objectives, Ankhili and Mansouri [11] take the margin function of the lower level problem as the penalty term, and construct the corresponding penalized problem. Then, an exact penalty function algorithm is proposed for the above BMPP. For the linear BMPP, where the leader has multiple objectives and the follower has single objective, Calvete and Gale [12] prove the existence of the weakly efficient solution and the efficient solution under the condition that the constraint region is not empty, then present the frameworks of some feasible algorithms. However, no numerical results are reported. In additions, for a class of BMPP, where the lower level is a convex vector-programming problem, Lv and Wan [13] adopt the method of replacing the lower level problem with its optimality conditions, after that smooth the complementary conditions with some smoothing function. Then, a smoothing approach is proposed. It deserves pointing out that the main drawback of the traditional numerical approach for the BMPP is that only someone (weakly) efficient solution can be obtained. A recent study by Eichfelder [14] suggests a refinement-based strategy in which the algorithm starts with a uniformly distributed set of points on upper level variable, and some (weakly) efficient solutions can be obtained.

Evolutionary approach, which can obtain the (weakly) efficient solutions for the multiobjective programming problem efficiently, has been widely applied to solve BMPP. Deb and Sinha [15] as well as Sinha and Deb [16] discuss BMPP based on evolutionary multiobjective optimization principles. Based on the above studies, Deb and Sinha [17] propose a viable and hybrid evolutionary-local-search based algorithm, and present challenging test problems. Sinha [18] presents a progressively interactive evolutionary multiobjective optimization method for bi-level multiobjective programming problem.

It is noted that as a metaheuristic, Particle Swarm Optimization (PSO) [19] has proved to be a competitive algorithm for optimization problems compared with other algorithms such as Genetic Algorithm (GA) and Simulating Algorithm (SA). It can converge to the optimal solution rapidly. Z h a n g et al. [20] use some improved PSO algorithms to solve BMPP. Subsequently, Z h a n g et al. [21] propose a hybrid particle swarm optimization with crossover operator for some high dimensional BMPP. Then, some numerical results are presented to illustrate the superiority of the improved PSO. It is noted that in the above references on solving bi-level multiobjective programming problem using PSO approach, the model of the upper and lower level solving their own problems interactively is adopted.

In this paper, based on our previous work [13], we will adopt a different tack from the existing PSO approach for the BMPP. Our strategy can be outlined as follows. For a class of BMPP, where the lower level is a convex vector optimization problem, we replace the lower level problem with its optimality conditions. After that, we smooth the complementary conditions with some smoothing function and obtain the corresponding smoothed multiobjective programming problem. Then, a particle swarm optimization approach is proposed to solve the smoothed multiobjective programming problem and the approximate efficient solutions for the BMPP are obtained.

This paper is organized as follows. In the following Section 2, we firstly introduce the mathematical model and some basic definitions of the efficient solutions of the bi-level multiobjective programs. In addition, the smoothing method for the BMPP, which we obtain in our previous work [13], is introduced. A particle swarm optimization approach for the smoothed multiobjective programming problem is proposed in Section 3. In Section 4, we report some numerical results to illustrate the PSO approach. Finally, we conclude this paper with some remarks.

2. Bi-level multiobjective programming and smoothing method

The bi-level multiobjective programming problem considered in this paper can be formulated as:

$$\begin{aligned} &\min_{(x,y)} F(x,y), \\ &\text{s.t. } y \in S(x), \end{aligned} \tag{1}$$

where $S(x)$ denotes the efficient solution set of the following lower level optimization problem,

$$\begin{aligned} &(P_x)\ \min_{y} f(x,y), \\ &\text{s.t. } g(x,y) \le 0, \end{aligned} \tag{2}$$

and $x \in R^n$, $y \in R^m$, and $F: R^{n\times m} \to R^p$, $f: R^{n\times m} \to R^q$, $g: R^{n\times m} \to R^l$ are continuously differentiable functions.

To facilitate the discussion, we also introduce the following notations, which is presented in our previous work [13]. Let $S = \{(x,y): g(x,y) \le 0\}$ denote the

constraint region of problem (1), $S_y = \{x \in R^m : \exists y \in R^m, g(x,y) \le 0\}$ denote the projection of S onto the leader's decision space. In additions, let $I(x,y)$ denote the set of the active constraints, i.e.,

$$I(x,y) := \{i : g_i(x,y) = 0, i = 1,\ldots,l\}. \tag{3}$$

Definition 2.1. A point (x,y) is feasible to problem (1) if $(x,y) \in S$, and $y \in S(x)$.

Definition 2.2. A point (x^*, y^*) is a Pareto optimal solution to problem (1) if it is feasible and there exists no other feasible point (x,y), such that $F(x,y) \le F(x^*, y^*)$ and $F(x,y) \ne F(x^*, y^*)$.

Through this paper, we make the following assumptions:

(A_1) The constraint region S is nonempty and compact.

(A_2) For each given upper level variable x, the lower level problem (P_x) is a convex vector optimization problem, and the partial gradient $\nabla_y g_i(x,y)$, $i \in I(x,y)$, is linear independent.

Assumption (A_1) guarantees that the feasible region of problem (1) is nonempty, and assumption (A_2) can make us adopt some scalarization method to transform the lower level problem into the corresponding scalar optimization problem [13]. That is, based on assumption (A_2), we can transform problem (1) into the following bi-level multiobjective programming problem, where the lower level is a scalar optimization problem [13],

$$\begin{aligned} &\min_{(x,y,\lambda)} F(x,y), \\ &\text{s.t.} \sum_{i=1}^{q} \lambda_i = 1, \\ &\lambda \ge 0, \\ &\min_{y} \langle \lambda, f(x,y) \rangle, \\ &\text{s.t. } g(x,y) \le 0. \end{aligned} \tag{4}$$

Let $\gamma(x,\lambda)$ denote the optimal solution set of the lower level problem in problem (2). On the relationships between the Pareto optimal solutions of problem (2) and that of the original problem (1), we have the following result.

Theorem 2.1 [13]. Let $(\bar{x}, \bar{y})$ be a Pareto optimal solution of problem (1). Then for all $\bar{\lambda} \in \Omega = \left\{ \lambda : \lambda \in R_+^q, \sum_{i=1}^{q} \lambda_i = 1 \right\}$, the point $(\bar{x}, \bar{y}, \bar{\lambda})$ is a Pareto optimal solution of problem (2). Conversely, if the point $(\bar{x}, \bar{y}, \bar{\lambda})$ is a Pareto optimal solution of problem (2), then $(\bar{x}, \bar{y})$ is a Pareto optimal solution of problem (1).

Going one step further to the problem (2), we replace the lower level scalar optimization problem with its Kuhn-Tucker optimality conditions [13], and problem (2) can be reduced as the following multiobjective programming problem with complementary conditions:

$$
\begin{aligned}
&\min_{(x,y,\lambda,u)} F(x,y),\\
&\text{s.t.} \sum_{i=1}^{q}\lambda_i = 1,\\
&\nabla_y \langle \lambda, f(x,y)\rangle + \sum_{j=1}^{l} u_j \nabla_y g_j(x,y) = 0,\\
&u^{\mathrm{T}} g(x,y) = 0,\\
&g(x,y) \le 0,\\
&\lambda \ge 0,\ u \ge 0,
\end{aligned}
\tag{5}
$$

where the term $u \in R^l$ is the Lagrangian multiplier.

To facilitate the depiction, we also adopt the following notations [13]. Let $H_1 : R^{n+m+q+l} \to R^m$, $H_2 : R^{n+m+q+l} \to R^l$, $H_3 : R^{n+m+q+l} \to R^l$, $H_4 : R^{n+m+q+l} \to R$, $H_5 : R^{n+m+q+l} \to R^q$, which is defined as:

$$
\begin{aligned}
&H_1(w) := H_1(x,y,\lambda,u) := \nabla_y \langle \lambda, f\rangle + \sum_{j=1}^{l} u_j \nabla_y g_j(x,y),\\
&H_2(w) := H_2(x,y,\lambda,u) := g(x,y),\\
&H_3(w) := H_3(x,y,\lambda,u) := u,\\
&H_4(w) := H_4(x,y,\lambda,u) := \sum_{i=1}^{q}\lambda_i - 1,\\
&H_5(w) := H_5(x,y,\lambda,u) := \lambda.
\end{aligned}
\tag{6}
$$

Based on the above notations (4), problem (3) can be rewritten as:

$$
\begin{aligned}
&\min F(w),\\
&\text{s.t. } H_1(w) = 0,\\
&\langle H_2(w), H_3(w)\rangle = 0,\\
&H_2(w) \le 0,\\
&H_3(w) \ge 0,\\
&H_4(w) = 0,\\
&H_5(w) \ge 0.
\end{aligned}
\tag{7}
$$

Let w^* be a feasible point to problem (5), we also define the following index set:

$$
\begin{aligned}
\alpha = \alpha(w^*) &= \{i : H_{2i}(w^*) < 0,\ H_{3i}(w^*) = 0,\ i = 1,\ldots,l\},\\
\beta = \beta(w^*) &= \{i : H_{2i}(w^*) = H_{3i}(w^*) = 0,\ i = 1,\ldots,l\},\\
\gamma = \gamma(w^*) &= \{i : H_{2i}(w^*) = 0,\ H_{3i}(w^*) > 0,\ i = 1,\ldots,l\},\\
\zeta = \zeta(w^*) &= \{i : H_{5i}(w^*) = 0,\ i = 1,\ldots,q\}.
\end{aligned} \tag{8}
$$

In the following context, the following assumption is satisfied.

(A_3) Let w^* is feasible to problem (5), the gradient vectors $\nabla H_{1i}(i = 1,\ldots,m)$, $\nabla H_{2i}(i \in \beta \cup \gamma)$, $\nabla H_{3i}(i \in \alpha \cup \beta)$, $\nabla H_4, \nabla H_{5i}(i \in \zeta)$ are linearly independent.

It is known that problem (5) is a nonsmooth multiobjective problem. In addition, in our previous work [13], to overcome the nonsmooth of problem (5), we adopt the following so-called perturbed Fischer-Burmeister function $\phi : R^2 \times R_+ \to R$, which has the formulation

$$
\phi(a, b, \varepsilon) = a + b - \sqrt{a^2 + b^2 + \varepsilon} \tag{9}
$$

to smooth problem (5). Moreover, the perturbed Fischer-Burmeister function $\phi(a, b, \varepsilon)$ has the property $\phi(a, b, \varepsilon) = 0 \Leftrightarrow a > 0,\ b > 0,\ ab = \frac{\varepsilon}{2}$.

Based on the perturbed Fischer-Burmeister function (7), we can smooth the complementary conditions $\langle H_2(w), H_3(w) \rangle = 0$ in problem (5) as follows:

$$
H_{3i}(w) - H_{2i}(w) - \sqrt{H_{2i}^2(w) + H_{3i}^2(w) + \varepsilon} = 0,\ i = 1,\ldots,l. \tag{10}
$$

Then for every $\varepsilon > 0$ we can obtain the following smooth multiobjective programming problem:

$$
\begin{aligned}
&\min\ F(w),\\
&\text{s.t. } H_1(w) = 0,\\
&H_{3i}(w) - H_{2i}(w) - \sqrt{H_{2i}^2(w) + H_{3i}^2(w) + \varepsilon} = 0,\ i = 1,\ldots,l,\\
&H_4(w) = 0,\\
&H_5(w) \geq 0.
\end{aligned} \tag{11}
$$

On the relationships between the Pareto optimal solutions of the smoothed problem (9) and that of problem (5), we have the following result in our previous work [13].

Theorem 2.2 [13]. Let w^ε be a Pareto optimal solution of problem (9) and suppose that $w^\varepsilon \to \bar{w}$ as $\varepsilon \to 0$, and the assumption (A_3) holds at $\bar{w}$. Then w^ε is a KKT point of problem (9) with multiplier $(u^\varepsilon, v^\varepsilon, \mu^\varepsilon, \nu^\varepsilon)$; Moreover, if $(u^\varepsilon, v^\varepsilon, \mu^\varepsilon, \nu^\varepsilon) \to (u', v', \mu', \nu')$ as $\varepsilon \to 0$, then $(\bar{x}, \bar{y})$ is a Pareto optimal solution to problem (1).

Based on Theorem 2.2, the nonsmooth multiobjective programming problem (5) can be progressively approximated by a sequence of smooth multiobjective programming problem (9). That means we can obtain the approximate Pareto optimal solution of problem (5) by solving problem (9). In fact, in our previous work [13], we adopt the numerical approach, such as ε-constraint approach, to solve problem (9) and obtain some Pareto optimal solution of the original problem (1). Here, we consider adopting PSO approach to solve problem (9), and obtain the Pareto optimal front of problem (1).

3. Particle swarm optimization for problem (9)

3.1. Overview of the PSO algorithm

Particle Swarm Optimization (PSO) is a kind of evolutionary algorithm, which has been widely studied in recent years. It is first proposed by Kennedy and Eberhart [19] in 1995 and successfully applied to the optimization problem, and then it has been effectively extended to some other aspects.

In PSO, the population is referred as a swarm and the individuals are called particles. Like other evolutionary algorithms, PSO performs searches using a population of individuals that are updated from iteration to iteration. To find the optimal or approximately optimal solution, each particle changes its searching direction according to two factors, its own best previous experience and the best experience of the entire swarm in the past. In the process of searching optimal solution, each particle changes its velocity and position according to the following formulas:

$$v_{t+1} = w \cdot v_t + c_1 \cdot r_1 (P_t - x_t) + c_2 \cdot r_2 (P_g - x_t), \quad x_{t+1} = x_t + v_{t+1}, \tag{12}$$

where r_1 and r_2 are random numbers in (0, 1); w is the inertia weight; c_1, c_2 are acceleration factors, which are the positive constant parameters; P_t is the personal best position for a particle; P_g is the global best position in the entire swarm. If the search space is D-dimensional, the velocity and position are represented as $v_t = (v_{t1}, v_{t2}, v_{t3} \cdots, v_{tD})$, $x_t = (x_{t1}, x_{t2}, x_{t3} \cdots, x_{tD})$, and the same with $P_t = (P_{t1}, P_{t2}, P_{t3} \cdots, P_{tD})$, $P_g = (P_{g1}, P_{g2}, P_{g3} \cdots, P_{gD})$.

3.2. The algorithm for solving multiobjective problem (9)

As mentioned above, we can obtain the approximate Pareto optimal solution of the original problem (1) by solving the smoothed multiobjective programming problem (9). In addition, in the following contents, we use PSO to solve problem (9).

Definition 3.1. Let w_d^{next} and w_d^{last} be the next and the last adjacent particles of the *d*-th particle w_d, respectively. The crowding distance of the *d*-th individual particle w_d is definied as

$$\sigma_d = \sum_{m=1}^{p} \frac{F_m(w_d^{\text{next}}) - F_m(w_d^{\text{last}})}{F_m^{\max} - F_m^{\min}}, \tag{13}$$

where *p* is the number of the objectives in problem (9); $F_m(w_d^{\text{next}})$, $F_m(w_d^{\text{last}})$ and $F_m(w_d)$ are respectively the fitness values of *m*-th objective in problem (9); $F_m^{\max}$ and $F_m^{\min}$ are the maximum and minimum values of the *m*-th objective, respectively.

The crowding distance is introduced as the index to judge the distance between the particle and the adjacent particle, and it reflects the congestion degree of no dominated solutions. In the population, the larger the crowding distance, the sparser and more uniform.

Definition 3.2. The infeasibility threshold η is defined as

$$\eta = \begin{cases} \eta_0 \times (1 - 5t / 4N), & t \le 0.8N, \\ 0, & t > 0.8N, \end{cases} \tag{14}$$

where η_0 is an initial value allowed by constraint violation degree, t is the current evolution generation, and N is the maximal evolution generation.

From the above formula (11), we can know that the infeasibility threshold ε decreases with the increase of the evolution generation.

Definition 3.3. The degree of the individual particle violating the constraints in problem (9) is defined as

$$C(w_d) = \sum_{i=1}^{q} \max(H_{5i}(w_d), 0) + \sum_{j=1}^{m} \max(\left|H_{1j}(w_d)\right| - \delta, 0) +$$

$$+ \sum_{\tau=1}^{l} \max(\left|H_{3\tau}(w_d) - H_{2\tau}(w_d) - \sqrt{H_{2\tau}^2(w_d) + H_{3\tau}^2(w_d) + \varepsilon}\right| - \delta, 0) + \max(\left|H_4(w_d)\right| - \delta, 0).$$

where δ is the tolerance of the equation constraints, which reflects the degree of the strictness on the equation constraints.

We firstly outline the PSO approach for the multiobjective programming problem. In the feasible solution space, we uniformly and randomly initialize the particle swarms and select the no dominated solution particles consisting of the elite set. After that by the methods of congestion degree choosing (the congestion degree can make the particles distribution more sparse) and the dynamic ε infeasibility dominating the constraints, we remove the no dominated particles in the elite set. Then, the objectives can be approximated.

The details of the proposed PSO algorithm for problem (9) can be described as follows.

Proposed PSO Algorithm

Step 1. Give the particle population size *M* (including the position x and the velocity v) and the maximal evolution generation *N*. Select the initial infeasibility threshold η_0, and let $t = 0$.

Step 2. Update each particle in the particle group:

Step 2.1. Archive the no dominated solutions of the particle swarm in the external elite set and calculate the congestion distance and the degree of the individual particle violating the constraints $C(w)$ on each nondominated solution in the external elite set. The distances are made in descending order. Then randomly select one particle as the global optimal position P_g from the archived elite set.

Step 2.2. Update the velocity and position of the particle by the Formula (10). If the position of a particle exceeds the preset boundary, the position of the particle is equal to its boundary value and its velocity is multiplied by "–1" to search the particle in the opposite direction.

Step 3. Update the external elitist set: Compare the updated nondominated solutions of the particle swarm with the nondominated solutions in the external elites, and decide whether the nondominated solutions in the particle swarm should be archived in the external elite set. If the solution in the particle swarm satisfies the domination relation, it needs to judge whether the external elitist set is full: if it is not full, the nondominated solution is archived directly; otherwise, the following steps are adopted:

Step 3.1. Archive all nondominated solutions of the external elite set in descending order according to the congestion distance.

Step 3.2. Randomly pick a particle in the *M* particles of the sorted set and replace it with the particle that needs to be archived.

Step 4. Update the local optimal position of the particle:

Step 4.1. Update the global optimum position if the position of the particle updated dominates its historical optimal position.

Step 4.2. If the updated particle position does not dominate its historical optimum position, according to 50% chance to retain its best position in history. When the degree of all the particles in the nondominated set violating the constraints $C(w)$ is zero, the algorithm terminates and we get the approximate Pareto optimal solutions w^* and the values $F(w^*)$. Otherwise, go to Step 5.

Step 5. If $t \geq N$ we get the approximate Pareto optimal solutions w^* and the values $F(w^*)$. Otherwise, set $t = t + 1$, and go to Step 2.

4. Numerical results

The Generational Distance (GD) is an indicator to estimate the average distance between the obtained optimal Pareto solutions and the theoretical optimal solutions. It can be defined as

$$\mathrm{GD} = \frac{\sum_{i=1}^{N} d_i}{N},$$

where N is the number of the obtained optimal solutions and d_i is the Euclidean distance in objective space between each of the founded optimal solutions and the nearest member of the theoretical optimal solution set. It is obvious that $\mathrm{GD} = 0$ means the all the founded optimal solutions belong to the theoretical optimal solution set.

SPacing (SP) is an indicator to describe the distribution uniformity of the obtained Pareto solutions. It can be defined as

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

where $d_i = \min_j (\left|F_1^i(x,y) - F_1^j(x,y)\right| + \left|F_2^i(x,y) - F_2^j(x,y)\right|, (i, j = 1, 2, \cdots, n)$; $\bar{d}$ is the mean of all d_i; d_m^{E} is the Euclidean distance between the extreme solutions in the 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; n is the number of the obtained solutions. It is obvious that $\mathrm{SP} = 0$ means that all the obtained Pareto solutions distribute evenly in objective space.

To verify the feasibility and effectiveness of the above PSO approach for the bi-level multiobjective programming problem, we give some numerical results. The perturbed Fischer-Burmeister function parameter $\varepsilon = 0.01$ and the PSO parameters are set as follows: $w = 0.7$ and $c_1 = c_2 = 1.5$. We make the programs and use a personal computer (CPU: Intel Pentium 2.26 GHZ, RAM:1G) to execute the programs.

Example 1. This example is taken from [22], where $x \in R$, $y \in R$. The particle population size and the maximal evolution generation for the above example are respectively set as $M = 100, N = 50$,

$$\begin{aligned}
&\max_x \ (-2x - y,\ x^2 - y)^{\mathrm{T}}, \\
&\text{s.t. } 0 \le x \le 15, \\
&\max_y \ (x + y^2,\ 2x - y)^{\mathrm{T}}, \\
&\text{s.t. } -x + y \le 0, \\
&2x - y \ge 4, \\
&x + 2y \le 33, \\
&y \ge 0.
\end{aligned} \tag{15}$$

For Example 1, in [22] the authors discuss two decision mechanisms: semi-cooperation decision mechanism and pure-independence, two solutions $(x, y) = (15, 0)$, and $(x, y) = (15, 8)$ are founded. Fig. 1 shows the two solutions and the Pareto optimal front by our PSO for Example 1. It can be seem that the solution

(15, 8) is not in the theoretical optimal solution set and the optimal solutions we found can integrally describe the Pareto optimal front.

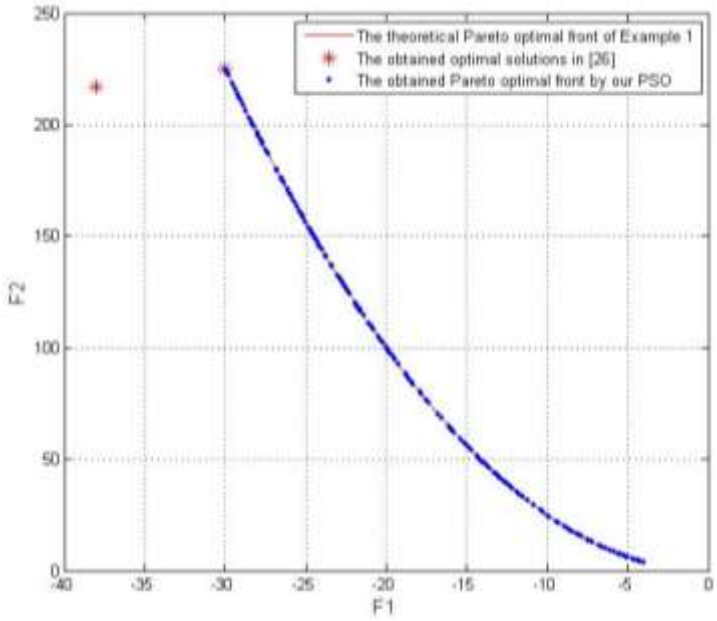

Fig. 1. The obtained Pareto optimal front of Example 1

Example 2. This example is taken from [23], where $x \in R,\ y \in R,$ and we set the particle population size and the maximal evolution generation as $M = 100,\ N = 50$.

$$
\begin{aligned}
&\max_{x}\ (-x+4y,-2x+y)^{\mathrm{T}},\\
&\text{s.t. } 0 \le x \le 3,\\
&\max_{y}\ (-y,-2y),\\
&\text{s.t. } -x-y \le -3,\\
&\quad -x+2y \ge 4,\\
&\quad 2x+y \le 12,\\
&\quad -3x+2y \le -4,\\
&\quad y \ge 0.
\end{aligned}
\tag{16}
$$

For Example 2, in [23], the authors use a fuzzy interactive method and five optimal solutions with different satisfactoriness of the upper level decision maker are founded. Fig. 2 shows the Pareto optimal front by our PSO in Example 3. It can be seem that the optimal solutions we found can integrally describe the Pareto optimal front, and the optimal solutions in [23] are included.

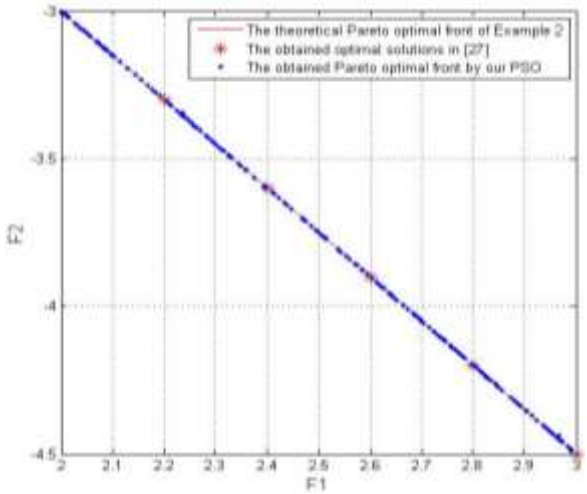

Fig. 2. The obtained Pareto optimal front of Example 2

Example 3. This example is taken from [24], where $x \in R$, $y \in R$, and we set the particle population size and the maximal evolution generation as $M = 200$, $N = 50$,

$$\min_x (-x - y, x^2 + (y-10)^2)^{\mathrm{T}},$$
$$\text{s.t. } 0 \le x \le 15,$$
$$\min_y (y^2, y(x-30))^{\mathrm{T}}, \tag{17}$$
$$\text{s.t. } y - x \le 0,$$
$$0 \le y \le 15.$$

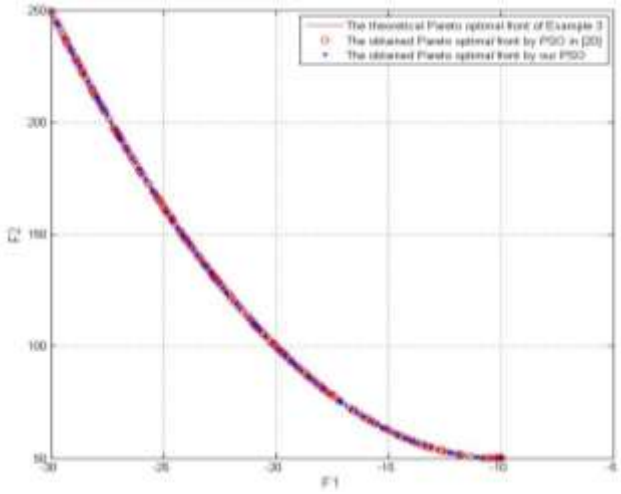

Fig. 3. The obtained Pareto optimal front of Example 3

Fig. 3 shows the Pareto optimal front of Example 3. In [20], the authors have also presented a PSO method for BMPP, in which the BMPP is transformed to solve many multiobjective optimization problems in the upper level and the lower level interactively with a predefined count.

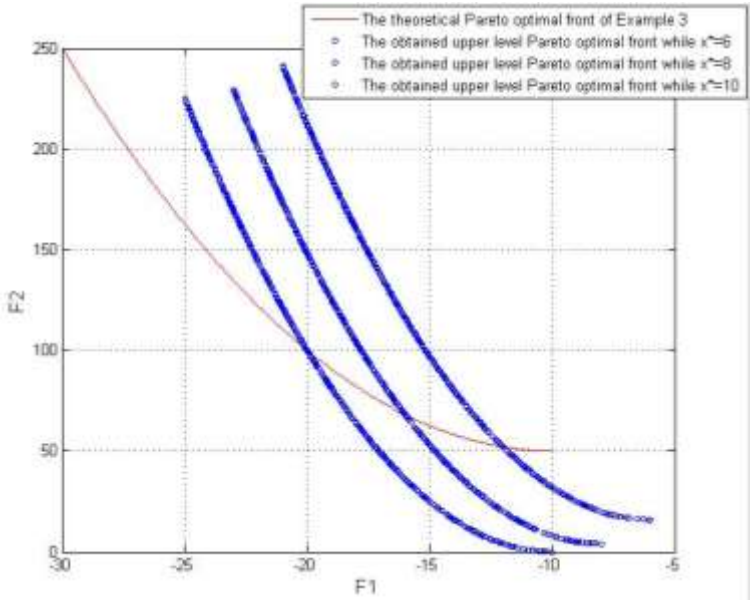

Fig. 4. The theoretical Pareto optimal front of Example 3 and the obtained upper level Pareto optimal fronts while x^* varies

Fig. 4 shows the theoretical Pareto optimal front of Example 3 and three obtained upper level Pareto optimal fronts with different upper decision variable x^*. It can be seen that each of the three obtained upper level Pareto optimal fronts has only one intersection with the theoretical Pareto optimal front of Example 3. This PSO method has to solve a series of multiobjective optimization problems in the upper level to find all theoretical optimal solutions.

Example 4. This example is taken from [25], where $x \in R, y \in R$ and we set the particle population size and the maximal evolution generation as $M = 200, N = 50,$

$$\begin{aligned} &\min_x (x^2, (y-10)^2)^{\mathrm{T}}, \\ &\text{s.t. } -x+y \le 0, \\ &0 \le x \le 15, \\ &\min_y ((x+2y-30)^2, x+0.5y^2)^{\mathrm{T}}, \\ &\text{s.t. } 0 \le y \le 15. \end{aligned} \tag{18}$$

Fig. 5 shows the obtained Pareto front by the PSO method in [20] and our PSO for Example 4, it can be seen that both the two obtained Pareto optimal fronts are very close to the theoretical Pareto optimal front.

To highlight the advantage of our PSO, a series of contrast numerical experiments between our PSO and the PSO method in [20] are conducted. In order to eliminate the difference of each experiment, we executed both the two algorithms for 50 times for each example. Table 1 shows the comparison of results between the two algorithms considering the two indicators previously described and the execution time (ET, in seconds). Considering the generation distance and the spacing, it can be seen that our PSO performs better than the PSO method in [20] except for example 2, in which the performance of our PSO is slightly worse. However, our PSO is much more efficient with respect to the execution time.

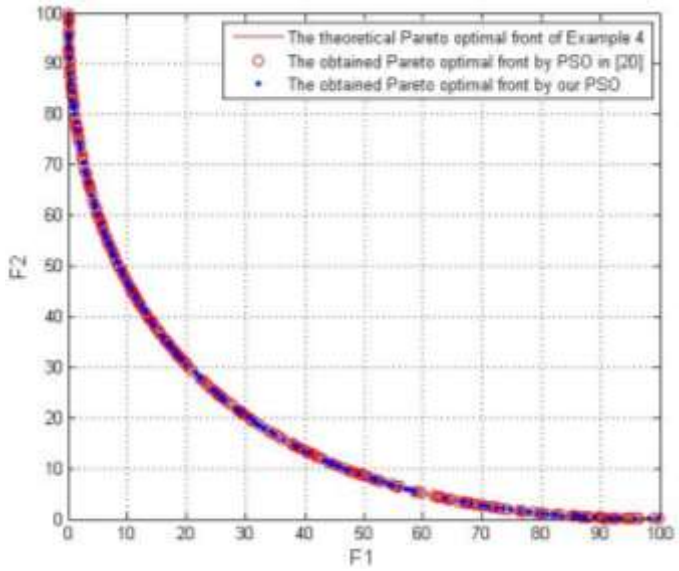

Fig. 5. The obtained Pareto optimal front of Example 4

Table 1. Results of the General Distance (GD), SPacing (SP) and Execution Time (ET) for the four examples

Example		GD		SP		ET(s)	
		PSO in [20]	Our PSO	PSO in [20]	Our PSO	PSO in [20]	Our PSO
Example 1	Max	3.27×10^{-4}	4.27×10^{-4}	1.48×10^{-1}	2.89×10^{-1}	1089.26	62.65
	Min	2.69×10^{-4}	9.87×10^{-5}	6.13×10^{-2}	4.78×10^{-2}	676.54	49.14
	Average	3.12×10^{-4}	2.07×10^{-4}	7.78×10^{-2}	6.74×10^{-2}	745.26	55.47
Example 2	Max	4.23×10^{-4}	1.04×10^{-2}	2.11×10^{-2}	6.49×10^{-2}	1447.03	117.78
	Min	8.71×10^{-5}	4.49×10^{-3}	7.46×10^{-3}	3.32×10^{-2}	864.54	87.54
	Average	3.47×10^{-4}	5.56×10^{-3}	1.55×10^{-2}	4.15×10^{-2}	1078.30	91.13
Example 3	Max	2.04×10^{-4}	1.44×10^{-4}	1.22×10^{-3}	3.77×10^{-3}	454.44	57.54
	Min	6.77×10^{-5}	6.41×10^{-5}	4.37×10^{-3}	9.89×10^{-4}	273.97	39.56
	Average	9.43×10^{-5}	8.55×10^{-5}	2.02×10^{-3}	1.97×10^{-3}	329.44	51.77
Example 4	Max	7.59×10^{-5}	5.44×10^{-5}	2.17×10^{-3}	4.33×10^{-3}	459.63	62.23
	Min	3.81×10^{-5}	2.26×10^{-5}	9.49×10^{-4}	9.49×10^{-4}	212.26	40.48
	Average	5.01×10^{-5}	4.89×10^{-5}	1.57×10^{-3}	1.35×10^{-3}	287.91	47.71

5. Conclusions

In this paper, we propose a different solving approach for the BMPP based on PSO. The main difference of the solving approach proposed here comparing to that in the existing references is that we only need to solve the corresponding smoothed multiobjective programming problem to obtain the Pareto optimal front of the BMPP. The numerical results prove that our PSO is a high efficiency and accurate method.

It is noted that in our solving approach based on PSO for the BMPP, we just adopt the original PSO algorithm. In recent years, the PSO algorithm is applied in many fields such as social network [26], neural networks optimization [27], reservoir operation [28], etc. The researchers have proposed some modified PSO algorithms [29] and combinatorial PSO algorithms [30] for the optimization problem. Our future work will be to lead these modified PSO approaches into our solving approach.

References

1. D e m p e, S. Foundation of Bi-Level Programming. London, Kluwer Academic Publishers, 2002.
2. S t o i l o v a, K., T. S t o i l o v, V. I v a n o v. Practical Bi-Level Optimization as a Tool for Implementation of Intelligent Transportation Systems. – Cybernetics and Information Technologies, Vol. **17**, 2017, No 2, pp. 97-105.
3. L v, Y., Z. C h e n, Z. W a n. A Penalty Function Method Based on Bi-Level Programming for Solving Inverse Optimal Value Problems. – Applied Mathematics Letters, Vol. **23**, 2010, pp. 170-175.
4. Y a n g, H., M. G. H. B e l l. Transport Bi-Level Programming Problems: Recent Methodological Advances. – Transportation Research, Vol. **35**, 2001, pp. 1-4.
5. D e m p e, S., A. B. Z e m k o h o. The Bi-Level Programming Problem: Reformulations, Constraint Qualifications and Optimality Conditions. – Mathematical Programming, Vol. **138**, 2013, No 1-2, pp. 447-473.
6. S i n h a, A., P. M a l o, K. D e b. A Review on Bi-Level Optimization: From Classical to Evolutionary Approaches and Applications. – IEEE Transactions on Evolutionary Computation, PP(99), 2017, pp. 1-1.
7. C o l s o n, B., P. M a r c o t t e, G. S a v a r d. An Overview of Bi-Level Optimization. – Annals of Operations Research, Vol. **153**, 2007, pp. 235-256.
8. N i s h i z a k i, I., M. S a k a w a. Stackelberg Solutions to Multiobjective Two-Level Linear Programming Problem. – Journal of Optimization Theory and Applications, Vol. **103**,1999, No 1, pp. 161-182.
9. L v, Y. An Exact Penalty Function Approach for Solving the Linear Bi-Level Multiobjective Programming Problem. – Filomat, Vol. **29**, 2015, No 4, pp. 773-779.
10. L v, Y., Z. W a n. Solving Linear Bi-Level Multiobjective Programming Problem via Exact Penalty Function Approach. – Journal of Inequalities and Applications, Vol. **2015**, 2015, 258.
11. A n k h i l i, Z., A. M a n s o u r i. An Exact Penalty on Bi-Level Programs with Linear Vector Optimization Lower Level. – European Journal of Operational Research, Vol. **197**, 2009, No 1, pp. 36-41.
12. C a l v e t e, H., C. G a l e. Linear Bi-Level Programs with Multi Objectives at the Upper Level. – Journal of Computational and Applied Mathematics, Vol. **234**, 2010, pp. 950-959.
13. L v, Y., Z. W a n. A Smoothing Method for Solving Bi-Level Multiobjective Programming Problems. – Journal of the Operations Research Society of China, Vol. **2**, 2014, No 4, pp. 511-525.
14. E i c h f e l d e r, G. Multiobjective Bi-Level Optimization. – Mathematical Programming, Vol. **124**, 2010, pp. 419-449.

15. D e b, K., A. S i n h a. Solving Bi-Level Multiobjective Optimization Problems Using Evolutionary Algorithms. – In: Lecture Notes in Computer Science, Evolutionary Multi-Criterion Optimization, Vol. **5467**, 2009, pp. 110-124.
16. S i n h a, A., K. D e b. Towards Understanding Evolutionary Bi-Level Multiobjective Optimization Algorithm. Technical Report KanGAL, Report No 2008006, 2008.
17. D e b, K., A. S i n h a. An Evolutionary Approach for Bi-Level Multiobjective Problems. – Communications in Computer and Information Science, Vol. **35**, 2009, pp. 17-24.
18. S i n h a, A. Bi-Level Multi-Objective Optimization Problem Solving Using Progressively Interactive EMO. – In: Lecture Notes in Computer Science, Evolutionary Multi-Criterion Optimization, Vol. **6576**, 2011, pp. 269-284.
19. K e n n e d y, J., R. E b e r h a r t. Particle Swarm Optimization. Perth, Aust: IEEE, Piscataway, NJ, USA, 1995.
20. Z h a n g, T., T. S. H u, Y. Z h e n g, X. N. G u o. An Improved Particle Swarm Optimization for Solving Bi-Level Multiobjective Programming Problem. – Journal of Applied Mathematics, Vol. **2012**, 2012.
21. Z h a n g, T., T. S. H u, Y. Z h e n g, X. N. G u o. Solving Bi-Level Multiobjective Programming Problem by Elite Quantum Behaved Particle Swarm Optimization. – Abstract and Applied Analysis, Vol. **2012**, 2012, ID 102482.
22. S h e n g, H., W. Z h o n g, N. X u. A Multiobjective Analysis of the Bi-Level Decision Making Problem and its Decision Method. – Journal of Systems Engineering, Vol. **11**, 1996, No 4, pp. 1-6.
23. Z h e n g, Y., Z. W a n. A Fuzzy Interactive Method for a Class of Bi-Level Multiobjective Programming Problem. – Expert Systems with Applications, Vol. **38**, 2011, No 8, pp. 10384-10388.
24. L i n, D., Y. C h o u, M. L i. Multiobjective Evolutionary Algorithm for Multi-Objective Bi-Level Programming Problem. – Journal of Systems Engineering, Vol. **22**, 2007, No 2, pp. 181-184.
25. T e n g, C., L. L i, H. L i. A Class of Genetic Algorithms on Bi-Level Multiobjective Decision Making Problem. – Journal of Systems Science and Systems Engineering, Vol. **9**, 2000, No 3, pp. 290-293
26. L i, Z. H., L. L. H e, H. Z h a n g. A Novel Social Network Structural Balance Based on the Particle Swarm Optimization Algorithm. – Cybernetics and Information Technologies, Vol. **15**, 2015, No 2, pp. 23-35.
27. L i, H. A Teaching Quality Evaluation Model Based on a Wavelet Neural Network Improved by Particle Swarm Optimization. – Cybernetics and Information Technologies, Vol. **14**, 2014, No 3, pp. 110-120.
28. S a b e r c h e n a r i, K., H. A b g h a r i, H. T a b r i. Application of PSO Algorithm in Short-Term Optimization of Reservoir Operation. – Environmental Monitoring and Assessment, Vol. **188**, 2016, No 12, p. 667.
29. D o n g, Y., C. S. W u, H. M. G u o. Particle Swarm Optimization Algorithm with Adaptive Chaos Perturbation. – Cybernetics and Information Technologies, Vol. **15**, 2015, No 6, pp. 70-80.
30. S u, Y. X., R. C h i. Multi-Objective Particle Swarm-Differential Evolution Algorithm. – Neural Computing and Applications, Vol. **28**, 2017, No 2, pp. 407-418.