A Compromise-based Particle Swarm Optimization Algorithm for Solving Bi-level Programming Problems with Fuzzy Parameters

Jialin Han, Yaoguang Hu
School of Mechanical Engineering,
Beijing Institute of Technology,
Beijing, China
e-mail: hjl@bit.edu.cn,
hyg@bit.edu.cn

Jialin Han, Guangquan Zhang, Jie Lu
Faculty of Engineering and Information Technology,
University of Technology Sydney,
Sydney, Australia
e-mail: hjl@bit.edu.cn,
{guangquan.zhang, jie.lu}@uts.edu.au

***Abstract*—Bi-level programming has arisen to handle decentralized decision-making problems that feature interactive decision entities distributed throughout a bi-level hierarchy. Fuzzy parameters often appear in such a problem in applications and this is called a fuzzy bi-level programming problem. Since the existing approaches lack universality in solving such problems, this study aims to develop a particle swarm optimization (PSO) algorithm to solve fuzzy bi-level programming problems in the linear and nonlinear versions. In this paper, we first present a general fuzzy bi-level programming problem and discuss related theoretical properties based on a fuzzy number ranking method commonly used. A PSO algorithm is then developed to solve the fuzzy bi-level programming problem based on different compromised selections by decision entities on the feasible degree for constraint conditions under fuzziness. Lastly, an illustrative numerical example and two benchmark examples are adopted to state the effectiveness of the compromise-based PSO algorithm.**

Keywords-bi-level programming; fuzzy number; swarm computing; particle swarm optimization; compromise

I. INTRODUCTION

Bi-level programming (also known as bi-level decision-making), motivated by Stackelberg game theory [1], seeks to address comprises among interactive decision entities that are distributed throughout a bi-level hierarchical organization. Decision entities at the upper level and the lower level are respectively termed the leader and the follower, and make their individual decisions in sequence with the aim of optimizing their respective objectives. This decision process means that the leader has priority in making its own decision and the follower reacts after and in full knowledge of the leader's decision. However, the leader's decision is implicitly affected by the follower's reaction. Since this category of hierarchical decision-making process often appears in many decentralized management problems in the real world, bi-level programming has motivated a number of research on solution approaches [2-4] and applications [5-7].

An important issue in modeling a bi-level programming problem is that parameters involved are sometimes obtained through experiments or experts' understanding of the nature of the parameters, which are often imprecisely or ambiguously known to the experts who establish the model; clearly, these parameters cannot be described by precise values [7]. With this observation, it would be certainly more appropriate to interpret the experts' understanding of such parameters as fuzzy numerical data that can be represented by means of fuzzy sets theory. A bi-level programming problem in which the parameters are described by fuzzy values, often characterized by fuzzy numbers, is called a fuzzy bi-level programming problem [8].

Since fuzzy bi-level programming can be used to handle many decentralized decision-making problems in the real world, it has motivated numerous research on solution approaches [9]. Zhang and Lu [10] proposed a general fuzzy linear bi-level programming problem and developed an approximation Kuhn-Tucker approach to solve this problem. They also presented an approximation *K*th-Best algorithm to solve the fuzzy linear bi-level decision problem [8]. Gao, et al. [11] proposed a programmable λ-cut approximate algorithm to solve a λ-cut set based fuzzy goal bi-level programming problem. Budnitzki [12] used the selection function approach and a modified version of *K*th-Best algorithm to solve a fuzzy linear bi-level programming problem. Also, fuzzy bi-level programming with multiple objectives has attracted numerous studies. Zhang, et al. [13] developed an approximation branch-and-bound algorithm to solve a fuzzy linear bi-level multi-objective programming problem. Gao, et al. [14] proposed a λ-cut and goal-programming-based algorithm to solve fuzzy linear bi-level multi-objective programming problems. Sakawa, et al. [15] proposed an interactive fuzzy programming approach to find a satisfactory solution to a fuzzy linear bi-level programming problem. They also extended the fuzzy approach to solve fuzzy linear bi-level fractional programming problems [16], fuzzy bi-level 0-1 programming problems [17] and fuzzy bi-level non-convex programming problems [18]. Based on fuzzy programming approaches proposed by Sakawa, Pramanik [19] adopted a fuzzy goal programming approach to solve fuzzy linear bi-level programming problems. However, these solution approaches are limited to handling some special fuzzy numbers and solving fuzzy programming problems in the linear version or in a special situation where all of the decision entities share the same constraint conditions. In particular, these interactive fuzzy approaches can only solve fuzzy bi-level programming problems in which decision entities from different levels prefer to

cooperate with one another, but the cooperation is inhibited in classical bi-level programming problems. Consequently, further investigation into solution approaches for solving fuzzy bi-level programming problems in much more generalized versions is necessary.

Since bi-level programming problems are strongly NP-hard and the existing solution approaches lack universality in solving such problems, intelligent heuristic algorithms may be used to generate an alternative for solving such problems. Particle swarm optimization (PSO) is a population-based heuristic algorithm first proposed by Kennedy and Eberhart [20], which is inspired by the social behavior of organisms such as fish schooling and bird flocking. As PSO requires only primitive mathematical operators, and is computationally inexpensive in terms of both memory requirements and speed [21], it has a good convergence performance and has been successfully applied in many fields such as multi-objective optimization [22], discrete optimization [23] and large-scale optimization [24]. In this study, we will try to develop a PSO algorithm to solve fuzzy bi-level programming problems involving linear and nonlinear versions.

The main contribution of this paper is the provision of a compromise-based PSO algorithm with the aim of solving much more generalized fuzzy bi-level programming problems involving the linear and nonlinear versions. This paper first presents a general fuzzy bi-level programming problem and discusses related theoretical properties based on a commonly used fuzzy number ranking method proposed by Jiménez [25]. It then develops a PSO algorithm to solve the proposed fuzzy bi-level programming problem based on the compromise between the leader and the follower on the satisfaction of constraint conditions under fuzziness. Lastly, numerical examples are used to illustrate the effectiveness of the proposed PSO algorithm.

II. Preliminaries

In this section, we present related notations and definitions that are used in the subsequent sections.

Definition 1 : [26] The membership function of a fuzzy number $\tilde{a}$ can be described in the following manner:

$$r=\mu_{\tilde{a}}(x)=\begin{cases}0 & x\in(-\infty,a_1],\\ f_a(x) & x\in[a_1,a_2],\\ 1 & x\in[a_2,a_3],\\ g_a(x) & x\in[a_3,a_4],\\ 0 & x\in[a_4,\infty),\end{cases}$$

where the function f_a and g_a are called the left and the right side of $\tilde{a}$, and f_a is an increasing and g_a is a decreasing function. The *r*-cuts are closed and bounded intervals and can be represented by $a_r=[f_a^{-1}(r),g_a^{-1}(r)]$.

Definition 2: [26] The *expected interval* and the *expected value* of a continuous fuzzy number $\tilde{a}$ are respectively defined as $EI(\tilde{a})$ and $EV(\tilde{a})$:

$$EI(\tilde{a})=[E_1^a,E_2^a]=\left[\int_0^1 f_a^{-1}(r)dr,\int_0^1 g_a^{-1}(r)dr\right],$$

$$EV(\tilde{a})=\frac{E_1^a+E_2^a}{2}.$$

Definition 3: [27] $EI(\lambda\tilde{a}+\gamma\tilde{b})=\lambda EI(\tilde{a})+\gamma EI(\tilde{b})$,

$$EV(\lambda\tilde{a}+\gamma\tilde{b})=\lambda EV(\tilde{a})+\gamma EV(\tilde{b}).$$

Definition 4: [28] For any pair of fuzzy numbers $\tilde{a}$ and $\tilde{b}$, the degree in which $\tilde{a}$ is bigger than $\tilde{b}$ is the following:

$$\mu_M(\tilde{a},\tilde{b})=\begin{cases}0 & E_2^a-E_1^b<0,\\ \dfrac{E_2^a-E_1^b}{E_2^a-E_1^b-(E_1^a-E_2^b)} & 0\in[E_1^a-E_2^b,E_2^a-E_1^b],\\ 1 & E_1^a-E_2^b>0,\end{cases}$$

where $[E_1^a,E_2^a]$ and $[E_1^b,E_2^b]$ are the expected intervals of $\tilde{a}$ and $\tilde{b}$. When $\mu_M(\tilde{a},\tilde{b})=0.5$, $\tilde{a}$ and $\tilde{b}$ are indifferent. $\mu_M(\tilde{a},\tilde{b})\ge\alpha$ means that $\tilde{a}$ is bigger than, or equal to $\tilde{b}$ at least in a degree α and that can be represented by $\tilde{a}\ge_\alpha\tilde{b}$.

III. The Fuzzy Bi-level Programming Problem and Related Theoretical Properties

In this section, we first present the fuzzy bi-level programming problem that is solved in this paper. Second, we discuss related theoretical properties based on the fuzzy number ranking method defined by Definition 3.

A. The Fuzzy Bi-level Programming Problem

The general fuzzy bi-level programming problem that is studied in this paper is defined as follows.

Definition 5: For $x\in X\subset R^p$, $y\in Y\subset R^q$, a general fuzzy bi-level programming problem is defined as:

$$\begin{aligned}&\min_{x\in X}\ \tilde{F}(x,y)=\tilde{c}_1F(x,y) && \text{(Leader)}\\ &\text{s.t. } \tilde{G}(x,y)\le\tilde{b}_1,\\ &\quad\text{where, for each } x \text{ fixed, } y \text{ solves} && (1)\\ &\quad\min_{y\in Y}\ \tilde{f}(x,y)=\tilde{c}_2f(x,y) && \text{(Follower)}\\ &\quad\text{s.t. } \tilde{g}(x,y)\le\tilde{b}_2,\end{aligned}$$

where x and y are the decision variables of the leader and the follower respectively; $\tilde{c}_1\in F^n(R)$, $\tilde{c}_2\in F^m(R)$, $\tilde{b}_1\in F^s(R)$, $\tilde{b}_2\in F^t(R)$, $F(x,y):R^p\times R^q\to R^n$, $f(x,y):R^p\times R^q\to R^m$, $\tilde{F}(x,y):R^p\times R^q\to F(R)$, $\tilde{f}(x,y):R^p\times R^q\to F(R)$, $\tilde{G}(x,y):R^p\times R^q\to F^s(R)$, $\tilde{g}(x,y):R^p\times R^q\to F^t(R)$, $F(R)$ is the set of all finite fuzzy numbers.

To find an acceptable optimal solution to the fuzzy bi-level programming problem (1), relevant solution concepts are presented as follows:

Definition 6:

1) The constraint region of the fuzzy bi-level problem (1):
$S=\{(x,y)\in X\times Y:\tilde{G}(x,y)\le\tilde{b}_1,\tilde{g}(x,y)\le\tilde{b}_2\}$.

2) The feasible set of the follower for each fixed x:

$S(x) = \{y \in Y : \tilde{g}(x, y) \le \tilde{b}_2\}$.

3) The rational reaction set of the follower:

$P(x) = \{y \in Y : y \in \arg\min[\tilde{f}(x, y) : y \in S(x)]\}$.

4) The inducible region of the fuzzy bi-level problem (1):

$IR = \{(x, y) : (x, y) \in S, y \in P(x)\}$.

5) The optimal solution set of the fuzzy bi-level problem (1):

$OS = \{(x, y) : (x, y) \in \arg\min[\tilde{F}(x, y) : (x, y) \in IR]\}$.

It is clear from Definition 6 that the constraint domain associated with a bi-level programming problem is implicitly determined by two optimization problems which must be solved in a predetermined sequence from the leader to the follower [29].

B. Related Theoretical Properties

For the sake of developing an efficient algorithm to solve the fuzzy bi-level programming problem (1), we now turn our attention to related theoretical properties based on the solution concepts and the fuzzy number ranking method, which are above mentioned.

Definition 7: Given a decision vector (x, y), it is said to be feasible in a degree α (α -feasible) to the constraint region *S* if

$$\min\{\mu_M(\tilde{b}_1, \tilde{G}(x, y)), \mu_M(\tilde{b}_2, \tilde{g}(x, y))\} = \alpha . \quad (2)$$

In view of Definition 4, the previous expression (2) can be written as :

$\alpha E_2^G + (1-\alpha)E_1^G \le (1-\alpha)E_2^{b_1} + \alpha E_1^{b_1}$,

$\alpha E_2^g + (1-\alpha)E_1^g \le (1-\alpha)E_2^{b_2} + \alpha E_1^{b_2}$.

The α -feasible constraint region of the fuzzy bi-level programming problem (1) can be denoted by

$${}_\alpha S = \{(x, y) \in X \times Y : \alpha E_2^G + (1-\alpha)E_1^G \le (1-\alpha)E_2^{b_1} + \alpha E_1^{b_1},$$
$$\alpha E_2^g + (1-\alpha)E_1^g \le (1-\alpha)E_2^{b_2} + \alpha E_1^{b_2}\} .$$

By Definition 7, if $\alpha_1 < \alpha_2$, then ${}_{\alpha_1}S \supset {}_{\alpha_2}S$.

In line with Definition 7, we let $\min\{\mu_M(\tilde{b}_1, \tilde{G}(x, y)\} = \alpha_L$ and $\min\{\mu_M(\tilde{b}_2, \tilde{g}(x, y)\} = \alpha_F$, thus, $\alpha = \min\{\alpha_L, \alpha_F\}$. For the fixed *x* by the leader, *y* can be said to be α_F -feasible to the feasible set of the follower *S*(*x*) under $\min\{\mu_M(\tilde{b}_2, \tilde{g}(x, y)\} = \alpha_F$. Accordingly, the feasible set of the follower in relation to all α_F -feasible decision vectors can be denoted by:

${}_{\alpha_F}S(x) = \{y \in Y : \alpha_F E_2^g + (1-\alpha_F)E_1^g \le (1-\alpha_F)E_2^{b_2} + \alpha_F E_1^{b_2}\}$, and the α_F -feasible rational reaction set of the follower can be written as:

$${}_{\alpha_F}P(x) = \{y \in Y : y \in \arg\min[\tilde{f}(x, y) : y \in {}_{\alpha_F}S(x)]\}. \quad (3)$$

Thus, the α -feasible inducible region of the fuzzy bi-level problem (1) is:

${}_\alpha IR = \{(x, y) : (x, y) \in {}_\alpha S, y \in {}_{\alpha_F}P(x)\}$.

Definition 8: For each given *x* by the leader, y^* is said to be an acceptable optimal solution to the problem $\min\{\tilde{f}(x, y) : y \in {}_{\alpha_F}S(x)\}$ if it is verified that:

$\mu_M(\tilde{f}(x, y), \tilde{f}(x, y^*)) \ge 0.5$ for $\forall y \in {}_{\alpha_F}S(x)$.

By Definition 3, we have

$$\tilde{f}(x, y) \ge_{0.5} \tilde{f}(x, y^*) \text{ for } \forall y \in {}_{\alpha_F}S(x) . \quad (4)$$

which means that y^* is a better choice of the follower at least in degree 0.5 as opposed to the other feasible solutions in ${}_{\alpha_F}S(x)$. Using the Definition 3, the previous expression (4) can be written as:

$$\frac{E_2^{\tilde{f}(x,y)} - E_1^{\tilde{f}(x,y^*)}}{E_2^{\tilde{f}(x,y)} - E_1^{\tilde{f}(x,y^*)} - (E_1^{\tilde{f}(x,y)} - E_2^{\tilde{f}(x,y^*)})} \ge 0.5$$

or

$$\frac{E_2^{\tilde{f}(x,y)} + E_1^{\tilde{f}(x,y)}}{2} \ge \frac{E_2^{\tilde{f}(x,y^*)} + E_1^{\tilde{f}(x,y^*)}}{2} .$$

In view of Definition 2 and Definition 3, the expression allows us to set the following proposition:

Proposition 1: For each fixed *x* , y^* is an α_F -acceptable optimal solution to the second-level problem $\min\{\tilde{f}(x, y) : y \in S(x)\}$ if it is an optimal solution to the following crisp problem:

$$\min\{EV(\tilde{f}(x, y)) : y \in {}_{\alpha_F}S(x)\}$$
$$= \min\{EV(\tilde{c}_2) f(x, y) : y \in {}_{\alpha_F}S(x)\}, \quad (5)$$

where $EV(\tilde{c}_2) \in F^m(R)$ is the expected value of the fuzzy vector $\tilde{c}_2$.

By Proposition 1, the expression (3) can be written as ${}_{\alpha_F}P(x) = \{y \in Y : y \in \arg\min[EV(\tilde{c}_2) f(x, y) : y \in {}_{\alpha_F}S(x)]\}$. Similarly, we have the following Proposition 2.

Proposition 2: (x^o, y^o) is an α -acceptable optimal solution to the fuzzy bi-level programming problem (1) if it is an optimal solution to the following crisp problem:

$$\min\{EV(\tilde{F}(x, y)) : (x, y) \in {}_\alpha IR\}$$
$$= \min\{EV(\tilde{c}_1) F(x, y) : (x, y) \in {}_\alpha IR\}, \quad (6)$$

where $EV(\tilde{c}_1) \in F^n(R)$ is the expected value of the fuzzy vector $\tilde{c}_1$.

In the light of the proposed definitions and propositions, we can find an optimal solution to the fuzzy bi-level programming problem (1) under the minimal feasible degrees α and α_F respectively preferred by the leader and the follower; the solution obtained is considered as at least an α -acceptable optimal solution where $\alpha \le \alpha_F$. However, to find a solution to the fuzzy bi-level programming problem (1), we should take into account two conflicting factors: the acceptable value for the objective functions and the feasible degree for the constraint conditions. On the one hand, for each fixed *x*, the objective value of the follower will become

worse following the increase in the feasible degree α_F. On the other hand, for all solutions $(x, y) \in_{\alpha} IR$, the objective value of the leader also becomes worse with the feasible degree α going up. The optimal solution obtained depends on the selection of the minimal feasible degrees α and α_F. In this paper, we will use a particle swarm optimization (PSO) algorithm to obtain optimal solutions in relation to different compromised selections of α and α_F by the leader and the follower. Finally, the decision marker can choose the preferred optimal solution in line with different decision situations. Note that if the leader and the follower share the same constraint conditions in problem (1), any α-acceptable optimal solution (x, y) ensures $\alpha = \alpha_F$.

IV. A Compromise-based Particle Swarm Optimization Algorithm

In this section, we develop a PSO algorithm for solving the problem (1) based on the compromise on the feasible degree for the constraint conditions between the leader and the follower.

Particle swarm optimization (PSO) is a category of the population-based heuristic algorithm that is motivated by the social behavior of organisms such as fish schooling and bird flocking. The population of PSO is known as a swarm, while each element in the swarm is termed a particle. In a swarm with the size *N*, the position vector of each particle with index i $(i = 1,2,\ldots,N)$ is denoted as $X_i^t = (x_i^t, y_i^t)$ at iteration *t*, which represents a potential solution to the problem (1). For the sake of convenient discussion, we let $X_i^t = (x_i^t, y_i^t) = (x_{i1}^t, x_{i2}^t)$. At iteration *t*, each particle *i* moves from X_i^t to X_i^{t+1} in the search space at a velocity $V_i^{t+1} = (v_{i1}^{t+1}, v_{i2}^{t+1})$ along each dimension. Each particle keeps track of its coordinates in hyperspace which are associated with the best solution (fitness), called *pbest* ($p_i = (p_{i1}, p_{i2})$), it has achieved so far; while the PSO algorithm is divided into two versions, respectively known as the GBEST version and the LBEST version, due to different definitions of the global best solution [21]. In the GBEST version, the particle swarm optimizer keeps track of the overall best value, called *gbest* ($p_g = (p_{g1}, p_{g2})$), and its location obtained thus far by any particle in the population, known as the global neighborhood. For the LBEST version, particles only contain their own and their nearest array neighbors' best information within a local topological neighborhood, rather than that of the entire group. However, in either PSO version, the PSO concept , at each iteration, always consists of an aggregated acceleration of each particle towards its *pbest* and *gbest* position. In this paper, the GBEST version of PSO is followed, and in this section, detailed procedures for solving the problem (1) are developed.

1) Initial population

In an initial population of particles with the number N, each particle i $(i = 1,2,\ldots,N)$ can be represented as $X_i^0 = (x_i^0, y_i^0) = (x_{i1}^0, x_{i2}^0)$. We randomly construct an initial population with the size *N*. where X_i^0 is randomly generated in ${}_{\alpha}S$ by setting $\alpha = 0$.

2) The updating rules of particles

In the PSO algorithm, each particle *i* moves toward $X_i^{t+1} = (x_i^{t+1}, y_i^{t+1}) = (x_{i1}^{t+1}, x_{i2}^{t+1})$ in the search space at a velocity $V_i^{t+1} = (v_{i1}^{t+1}, v_{i2}^{t+1})$ at each iteration *t*. In this paper, the velocity and position of each particle *i* are updated as follows for $j = 1,2, i = 1,2,\ldots,N$ based on related definitions proposed by Shi and Eberhart [30]:

$$v_{ij}^{t+1} = w v_{ij}^t + c_1 r_1 (p_{ij}^t - x_{ij}^t) + c_2 r_2 (p_{gj}^t - x_{ij}^t), \tag{7}$$

$$x_{ij}^{t+1} = x_{ij}^t + v_{ij}^{t+1}. \tag{8}$$

We now determine the selection of parameters involved in the formula (7). For the updating velocity, there are usually maximum and minimum velocity levels $v_{\max}$ and $v_{\min}$. If the current velocity $v_j^{t+1} > v_{\max}$, we set $v_j^{t+1} = v_{\max}$; while $v_j^{t+1} = v_{\min}$ if $v_j^{t+1} < v_{\min}$. In the beginning, we set $v_j^0 = v_{\max}$.

w is inertia weight, which controls the impact of the previous velocities on the current velocity. The inclusion of the inertia weight involves two definitions proposed by Shi and Eberhart [30]: a fixed constant and a decreasing function with time. In our PSO algorithm, we use the latter to define the inertia weight, because large inertial weight can be used to possess more exploitation ability at the beginning to find a good seed while it is reduced for better local exploitation later on in the search [30]. The inertia weight is represented as:

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{Iter\_\max} * t, \tag{9}$$

where $w_{\max}$ and $w_{\min}$ are the upper and lower bounds on the inertia weight, which are determined by the practical problem; *Iter_max* is the maximum number of PSO iterations while *t* represents the current iteration number.

c_1 and c_2 are known as learning factors or acceleration coefficients, which control the maximum step size that the particle can do. A recommended choice for constant c_1 and c_2 is integer 2 as proposed by Kennedy and Eberhart [20].

r_1 and r_2 are uniform random numbers between 0 and 1.

3) Fitness evaluation

For each particle *i* at the iteration *t* $X_i^t = (x_i^t, y_i^t)$, adopt the existing simplex method or interior point method to solve the problem (5) under $x = x_i^t$ and $\alpha_F = \alpha_F^*$ specified by the follower using the existing simplex method or interior point method, then obtain the solution (x_i^t, y^*) where $y^* \in_{\alpha_F^*} P(x_i^t)$ and update $X_i^t = (x_i^t, y_i^t) = (x_i^t, y^*)$. If

$\min\{\mu_M(\tilde{b}_1, \tilde{G}(x_i^t, y_i^t))\} = \alpha_L^* \geq \alpha^*$ where α^* is specified by the leader, then $y_i^t \in_{\alpha_F^*} P(x_i^t)$ and $(x_i^t, y_i^t) \in_{\alpha^*} S$, which means $(x_i^t, y_i^t) \in_{\alpha^*} IR$; that is, (x_i^t, y_i^t) is at least a α^*-acceptable feasible solution to the fuzzy bi-level programming problem (1). The *pbest* solution is $p_i = (p_{i1}, p_{i2}) = (x_i^t, y_i^t)$ and the exact feasible degree for the constraint region S is $\alpha(p_i) = \min\{\alpha_L^*, \alpha_F^*\} \geq \alpha^*$, if $EV(\tilde{F}(x_i^t, y_i^t)) \leq EV(\tilde{F}(p_{i1}, p_{i2}))$ where we set $p_i = (p_{i1}, p_{i2}) = (x_i^0, y_i^0)$, $EV(\tilde{F}(p_{i1}, p_{i2})) = +\infty$ and $\alpha(p_i) = 0$ at the beginning. The global best solution *gbest* of the swarm is $p_g = (p_{g1}, p_{g2})$ and the corresponding feasible degree for the constraint region S is $\alpha = \alpha(p_g)$ where

$$EV(\tilde{F}(p_{g1}, p_{g2})) = \min\{EV(\tilde{F}(p_{i1}, p_{i2})), i = 1,2,\ldots,N\}.$$

Clearly, $p_g = (p_{g1}, p_{g2})$ is an α-acceptable optimal solution to the fuzzy bi-level problem (1).

4) Termination criterion

The PSO algorithm will be terminated after a maximum number of iterations *Iter_max* or when it achieves a maximum CPU time.

Based on the theoretical basis proposed above, we will present the complete computational procedures of the PSO algorithm for solving the fuzzy bi-level programming problem (1).

Step 1: *Initialization*.

a) Construct the population size N and generate the initial population of particles $X_i^0 = (x_i^0, y_i^0), i = 1,2,\ldots,N$;

b) Initialize the *pbest* solution $p_i = (p_{i1}, p_{i2}) = (x_i^0, y_i^0)$, the fitness $EV(\tilde{F}(p_i)) = +\infty$ and the feasible degrees for the constraint conditions $\alpha_F = \alpha_F^*$, $\alpha = \alpha^*$ and $\alpha(p_i) = 0$;

c) Set the maximum and minimum velocity levels $v_{\max}$ and $v_{\min}$, and initialize $v_{ij}^0 = v_{\max}$;

d) Set the upper and lower bounds on the inertia weight $w_{\max}$ and $w_{\min}$, acceleration coefficients c_1 and c_2, and the maximum iteration number *Iter_max*;

e) Set the current iteration number t=0 and go to Step 2.

Step 2: *Compute the fitness value and update the pbest solution for each particle*. Set i=1 and go to Step 2.1.

Step 2.1: Under $x = x_i^t$, solve the problem (5) under $x = x_i^t$ and $\alpha_F = \alpha_F^*$ using the existing simplex method or interior point method, obtain the solution (x_i^t, y^*) and update $X_i^t = (x_i^t, y_i^t) = (x_i^t, y^*)$. Go to Step 2.2.

Step 2.2: If $\min\{\mu_M(\tilde{b}_1, \tilde{G}(x_i^t, y_i^t))\} = \alpha_L^* \geq \alpha^*$, go to Step 2.3; otherwise, go to Step 2.4.

Step 2.3: If $EV(\tilde{F}(x_i^t, y_i^t)) \leq EV(\tilde{F}(p_{i1}, p_{i2}))$, update $p_i = (p_{i1}, p_{i2}) = (x_i^t, y_i^t)$ and $\alpha(p_i) = \min\{\alpha_L^*, \alpha_F^*\}$. Go to Step 2.4.

Step 2.4: If i<N, set i=i+1 and go to Step 2.1; otherwise, go to Step 3.

Step 3: *Update the gbest solution*. Set $p_g = (p_{g1}, p_{g2})$ and $\alpha = \alpha(p_g)$ where

$$EV(\tilde{F}(p_{g1}, p_{g2})) = \min\{EV(\tilde{F}(p_{i1}, p_{i2})), i = 1,2,\ldots,N\}.$$

Go to Step 4.

Step 4: *Termination criterion*. If t<*Iter_max*, go to Step 5; otherwise, stop and $p_g = (p_{g1}, p_{g2})$ is an α-acceptable optimal solution to the fuzzy bi-level programming problem (1).

Step 5: *Update the inertia weight, and the velocity and the position of each particle* by the formulas (7), (8) and (9). If the current velocity $v_{ij}^{t+1} > v_{\max}$, set $v_{ij}^{t+1} = v_{\max}$; while $v_{ij}^{t+1} = v_{\min}$ if $v_{ij}^{t+1} < v_{\min}$. Set t=t+1 and go to Step 2.

V. Numerical Examples

In this section, we first illustrate how the proposed compromise-based PSO algorithm works through solving a fuzzy nonlinear bi-level programming problem in which the fuzzy numbers are characterized by nonlinear membership functions. Second, we use the proposed PSO algorithm to solve two benchmark problems and compare the results with that obtained by the existing algorithms.

A. An Illustrative Example

Consider the following fuzzy nonlinear bi-level programming problem (10):

$$\begin{aligned}
&\min_x \tilde{F}(x,y) = -\tilde{1}x_1^2 - \tilde{3}x_2^2 - \tilde{4}y_1 + \tilde{1}y_2^2 \\
&\text{s.t. } \tilde{1}x_1^2 + \tilde{2}x_2 \leq \tilde{4}, \\
&\quad x \geq 0, \\
&\quad \text{where } y \text{ solves:} \\
&\quad \min_y \tilde{f}(x,y) = \tilde{2}x_1^2 + \tilde{1}y_1^2 - \tilde{5}y_2 \\
&\quad \text{s.t. } \tilde{1}x_1^2 - \tilde{2}x_1 + \tilde{1}x_2^2 - \tilde{2}y_1 + \tilde{1}y_2 \geq -\tilde{3}, \\
&\qquad \tilde{1}x_2 + \tilde{3}y_1 - \tilde{4}y_2 \geq \tilde{4}, \\
&\qquad y \geq 0.
\end{aligned} \tag{10}$$

The membership functions of the coefficients in this example are given as follows:

$$\mu_{\tilde{1}}(t) = \begin{cases} 0 & t < 0 \\ t^2 & 0 \leq t \leq 1 \\ \dfrac{4-t^2}{3} & 1 \leq t \leq 2 \\ 0 & t > 2 \end{cases}, \quad \mu_{\tilde{2}}(t) = \begin{cases} 0 & t < 1 \\ \dfrac{t^2-1}{3} & 1 \leq t \leq 2 \\ \dfrac{9-t^2}{5} & 2 \leq t \leq 3 \\ 0 & t > 3 \end{cases},$$

$$\mu_{\tilde{3}}(t)=\begin{cases}0 & t<2\\ \dfrac{t^2-4}{5} & 2\le t\le 3\\ \dfrac{16-t^2}{7} & 3\le t\le 4\\ 0 & t>4\end{cases},\quad \mu_{\tilde{4}}(t)=\begin{cases}0 & t<3\\ \dfrac{t^2-9}{7} & 3\le t\le 4\\ \dfrac{25-t^2}{9} & 4\le t\le 5\\ 0 & t>5\end{cases},$$

$$\mu_{\tilde{5}}(t)=\begin{cases}0 & t<4\\ \dfrac{t^2-16}{9} & 4\le t\le 5\\ \dfrac{36-t^2}{11} & 5\le t\le 6\\ 0 & t>6\end{cases}.$$

Whereas the existing solution approaches cannot be adopted to solve the fuzzy nonlinear bi-level programming problem, we use the proposed PSO algorithm to find compromised solutions for the problem. Based on the PSO procedures developed in Section IV, the related parameters involved in the algorithm are initialized in TABLE I.

TABLE I. PARAMETERS EMPLOYED IN THE PSO ALGORITHM FOR SOLVING THE PROBLEM (10)

N	$v_{\max}$	$v_{\min}$	$w_{\max}$	$w_{\min}$	c_1	c_2	*Iter_max*
30	1.0	-1.0	0.5	0.01	2.0	2.0	60

The PSO algorithm is implemented in MATLAB R2014a. The computational results under different compromised selections of α and α_F are reported in TABLE II. In TABLE II, the first column α^* and the second column α_F^* are the minimal α and α_F respectively preferred by the leader and the follower. The fifth column α represents the exact feasible degree for constraint conditions under the solution (x, y), which indicates that the solution (x, y) is an α -acceptable optimal solution to the numerical example. The last column shows the iteration number when the PSO algorithm is convergent. In the real world, decision entities can make free choices of their preferred solutions from TABLE II in view of various decision situations in relation to their decentralized management problems.

TABLE II. THE COMPUTATIONAL RESULTS OF PROBLEM (10) UNDER DIFFERENT COMPROMISED CONDITIONS

α^*	α_F^*	**(x, y)**	$(EV(\tilde{F}), EV(\tilde{f}))$	α	**Iterations**
0.5	0.5	(0.0014, 1.9669, 1.7003, 0.8241)	(-17.7991, -0.9220)	0.5	39
	0.6	(0.0010, 1.9669, 1.6050, 0.6290)	(-17.7309, -0.2933)	0.5	32
	0.7	(0.0016, 1.9669, 1.5142, 0.4506)	(-17.5797, 0.2870)	0.5	35
	0.8	(0.0002, 1.9669, 1.4276, 0.2872)	(-17.3653, 0.8235)	0.5	39
	0.9	(0.0001, 1.9669, 1.3448, 0.1374)	(-17.1032, 1.3202)	0.5	31
	1.0	(0, 1.9669, 1.2672, 0.0007)	(-16.8121, 1.7809)	0.5	38
0.6	0.6	(0.0003, 1.8307, 1.6050, 0.5952)	(-16.2103, -0.1238)	0.6	33
	0.7	(0.0005, 1.8307, 1.5142, 0.4205)	(-16.0423, 0.4381)	0.6	32
	0.8	(0.0011, 1.8307, 1.4276, 0.2606)	(-15.8149, 0.9571)	0.6	44
	0.9	(0.0009, 1.8307, 1.3448, 0.1141)	(-15.5432, 1.4371)	0.6	37
	1.0	(0.0018, 1.8307, 1.3019, 0)	(-15.3852, 1.8832)	0.6	36
0.7	0.7	(0.0009, 1.7063, 1.5142, 0.3930)	(-14.7354, 0.5761)	0.7	46
	0.8	(0.0039, 1.7063, 1.4276, 0.2363)	(-14.4965, 1.0790)	0.7	29
	0.9	(0.0051, 1.7063, 1.3448, 0.0928)	(-14.2162, 1.5438)	0.7	41
	1.0	(0.0008, 1.7063, 1.3346, 0)	(-14.1848, 1.9790)	0.7	47
0.8	0.8	(0.0004, 1.5924, 1.4276, 0.2140)	(-13.3696, 1.1907)	0.8	36
	0.9	(0.0035, 1.5924, 1.3448, 0.0734)	(-13.0817, 1.6414)	0.8	36
	1.0	(0.0097,1.5924, 1.3646, 0)	(-13.1667, 2.0692)	0.8	46
0.9	0.9	(0.0009, 1.4877, 1.3448, 0.0555)	(-12.1071, 1.7313)	0.9	38
	1.0	(0.0110, 1.4875, 1.3922, 0)	(-12.3000,2.1537)	0.9	30
1.0	1.0	(0.0544, 1.3708, 1.4229, 0)	(-11.4164, 2.2556)	1.0	48

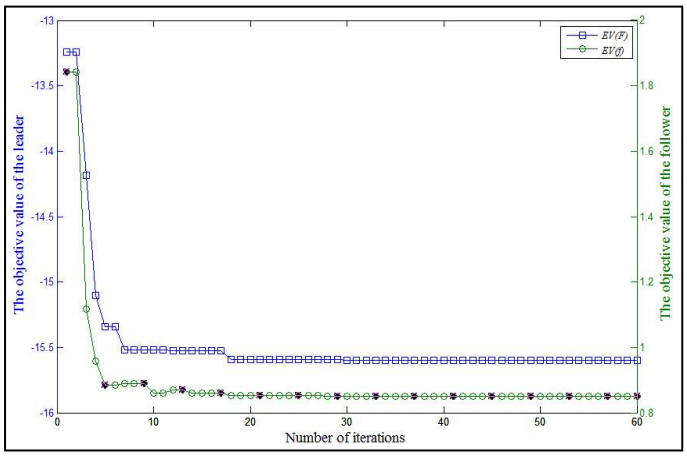

Figure 1. The convergence curves of the leader's and the follower's expected objective values

In regard to solving this numerical example (10), we can also randomly generate a pair of α^* and α_F^* in the interval [0.5, 1] and [α^*, 1]. The computational results imply that we can obtain a convergent solution using the proposed PSO algorithm under the parameters shown in TABLE I. For example, the convergence curves of the expected objective values of the leader and the follower $(EV(\tilde{F}), EV(\tilde{f}))$ under $(\alpha^*, \alpha_F^*) = (0.8320, 0.9386)$ are shown in Figure 1. It can be seen from Figure 1 that the expected objective values of the leader and the follower have converged to $(EV(\tilde{F}), EV(\tilde{f})) = (-15.5979, 0.8508)$ since the 30th iteration. With this observation, we can obtain a *gbest* solution $p_g = (0.0040, 1.8048, 1.4499, 0.2960)$ for the fuzzy nonlinear bi-level programming problem. Clearly, the PSO algorithm provides a practical way to solve bi-level programming problems with fuzzy parameters.

B. Benchmark Examples

In this section, the proposed PSO algorithm is applied to solve two benchmark problems that respectively appear in [8] and [10]. Also, we compare the computational results that are respectively obtained by the PSO algorithm and provided in [8] and [10].

The related parameters involved in the PSO algorithm for solving the problems are initialized as the same, shown in TABLE III. TABLE IV and TABLE V respectively display the results for the problems in [8] and [10] obtained by the PSO algorithm under different compromised α^* and α_F^*. Decision entities are able to choose their preferred optimal solution from TABLE IV and TABLE V in line with different decision situations. It is noticeable that the solution provided in [8] is $(x, y) = (0.5, 1.25)$ that is the same as our solution obtained under the compromised condition $\alpha^* = \alpha_F^* = 0.7727$. As well, the solution reported in [10] is $(x, y) = (0, 0.5)$ and $(EV(\tilde{F}), EV(\tilde{f})) = (-1.0, 0.50)$ that satisfies $(x, y) \in_{\alpha^*} IR$ with $\alpha^* = \alpha_F^* = 0.8333$. Under the same decision situation $\alpha^* = \alpha_F^* = 0.8333$, our PSO algorithm can find a better solution $(x, y) = (0.3334, 11667)$ and $(EV(\tilde{F}), EV(\tilde{f})) = (-2.0, 1.5001)$. Clearly, the PSO algorithm provides not only more options of solutions due to different decision environments but also better solutions under the same decision situation for the decision entities.

TABLE III. PARAMETERS EMPLOYED IN THE PSO ALGORITHM FOR SOLVING THE PROBLEMS IN [8] AND [10]

N	v_{max}	v_{min}	w_{max}	w_{min}	c_1	c_2	*Iter_max*
20	0.5	-0.5	0.5	0.01	2.0	2.0	40

TABLE IV. THE COMPUTATIONAL RESULTS OF THE PROBLEM IN [8] OBTAINED BY THE PSO ALGORITHM

α^*	α_F^*	**(x, y)**	$(EV(\tilde{F}), EV(\tilde{f}))$	α	**Iterations**
0.5	0.5	(2.0, 2.0)	(5.0, 2.0)	0.5	23
0.6	0.6	(1.2308, 1.6154)	(5.0, 1.6154)	0.6	22
0.7	0.7	(0.75, 1.375)	(5.0, 1.3750)	0.7	29
0.7727	0.7727	(0.50, 1.25)	(5.0, 1.2500)	0.7727	22
0.8	0.8	(0.421, 1.2105)	(5.0, 1.2105)	0.8	21
0.9	0.9	(0.1818, 1.0909)	(5.0, 1.0909)	0.9	21
1.0	1.0	(0, 1.0)	(5.0, 1.0)	1.0	33

TABLE V. THE COMPUTATIONAL RESULTS OF THE PROBLEM IN [10] OBTAINED BY THE PSO ALGORITHM

α^*	α_F^*	**(x, y)**	$(EV(\tilde{F}), EV(\tilde{f}))$	α	**Iterations**
0.5	0.5	(2.0, 2.0)	(-2.0, 4.0)	0.5	23
0.6	0.6	(1.2308, 1.6154)	(-2.0, 2.8462)	0.6	24
0.7	0.7	(0.75, 1.375)	(-2.0, 2.1250)	0.7	31
0.8	0.8	(0.421, 1.2105)	(-2.0, 1.6315)	0.8	28
0.8333	0.8333	(0.3334, 1.1667)	(-2.0, 1.5001)	0.8333	30
0.9	0.9	(0.1818, 1.0909)	(-2.0, 1.2727)	0.9	25
1.0	1.0	(0, 1.0)	(-2.0, 1.0)	1.0	32

VI. CONCLUSIONS AND FURTHER STUDY

This study developed a compromise-based PSO algorithm to solve fuzzy bi-level programming problems. First, we proposed the general fuzzy bi-level programming problem and discussed related theoretical properties. Second, we presented the procedures of the PSO algorithm, based on the compromise on the feasible degree for the constraint conditions between the leader and the follower, for solving the proposed fuzzy bi-level programming problem. Lastly, we illustrated how the PSO algorithm works through three numerical examples. The computational results show that the

PSO algorithm provides a practical way to solve fuzzy bi-level programming problems involve the linear and nonlinear versions. Moreover, compared with the existing solution approaches, the PSO algorithm can provide not only more options of solutions due to different decision environments but also better solutions under the same decision situation for the decision entities. In the future, we will apply the fuzzy bi-level programming techniques to handle decentralized decision-making problems under an uncertain situation in the real world. Also, we will explore the efficiency performance of the PSO algorithm for solving large-scale fuzzy bi-level programming problems in applications.

ACKNOWLEDGMENT

This work is supported by the Australian Research Council (ARC) under discovery grant DP140101366 and the National High Technology Research and Development Program of China (NO. 2013AA040402).

REFERENCES

[1] H. V. Stackelberg, The Theory of Market Economy. Oxford: Oxford University Press, 1952.

[2] W. F. Bialas and M. H. Karwan, "On two-level optimization," IEEE Trans Automatic Control, vol. AC-26, pp. 211-214, 1982.

[3] J. F. Bard, Practical Bilevel Optimization: Algorithms and Applications. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1998.

[4] C. Shi, H. Lu, and G. Zhang, "An extended Kth-best approach for linear bilevel programming," Applied Mathematics and Computation, vol. 164, pp. 843-855, May 2005.

[5] S. Dempe, V. V. Kalashnikov, G. A. Pérez-Valdés, and N. I. Kalashnykova, "Natural gas bilevel cash-out problem: Convergence of a penalty function method," European Journal of Operational Research, vol. 215, pp. 532-538, 2011.

[6] G. Zhang, G. Zhang, Y. Gao, and J. Lu, "Competitive strategic bidding optimization in electricity markets using bilevel programming and swarm technique," IEEE Transactions on Industrial Electronics, vol. 58, pp. 2138-2146, 2011.

[7] J. Han, J. Lu, Y. Hu, and G. Zhang, "Tri-level decision-making with multiple followers: Model, algorithm and case study," Information Sciences, vol. 311, pp. 182-204, 8/1/ 2015.

[8] G. Zhang and J. Lu, "Model and approach of fuzzy bi-level decision making for logistics planning problem," Journal of Enterprise Information Management, vol. 20, pp. 178-197, 2007.

[9] G. Zhang, J. Lu, and Y. Gao, Multi-Level Decision Making: Models, Methods and Applications vol. 82. Berlin: Springer, 2015.

[10] G. Zhang and J. Lu, "The definition of optimal solution and an extended Kuhn-Tucker approach for fuzzy linear bi-level programming," IEEE Computational Intelligence Bulletin, vol. 2, pp. 1-7, 2005.

[11] Y. Gao, G. Zhang, J. Lu, T. Dillon, and X. Zeng, "A λ-cut-approximate algorithm for goal-based bilevel risk management systems," International Journal of Information Technology and Decision Making vol. 7, pp. 589-610, 2008.

[12] A. Budnitzki, "The solution approach to linear fuzzy bilevel optimization problems," Optimization, pp. 1-15, 2013.

[13] G. Zhang, J. Lu, and T. Dillon, "Decentralized multi-objective bilevel decision making with fuzzy demands," Knowledge-Based Systems, vol. 20, pp. 495-507, 2007.

[14] Y. Gao, G. Zhang, J. Ma, and J. Lu, "A λ-cut and goal programming based algorithm for fuzzy linear multiple objective bi-level optimization," IEEE Transactions on Fuzzy Systems, vol. 18, pp. 1-13, 2010.

[15] M. Sakawa, I. Nishizaki, and Y. Uemura, "Interactive fuzzy programming for multi-level linear programming problems with fuzzy parameters," Fuzzy Sets and Systems, vol. 109, pp. 3-19, 2000.

[16] M. Sakawa, I. Nishizaki, and Y. Uemura, "Interactive fuzzy programming for two-level linear fractional programming problems with fuzzy parameters," Fuzzy Sets and Systems, vol. 115, pp. 93-103, 2000.

[17] M. Sakawa, I. Nishizaki, and M. Hitaka, "Interactive fuzzy programming for multi-level 0-1 programming problems with fuzzy parameters through genetic algorithms," Fuzzy Sets and Systems, vol. 117, pp. 95-111, 2001.

[18] M. Sakawa and I. Nishizaki, "Interactive fuzzy programming for two-level nonconvex programming problems with fuzzy parameters through genetic algorithms," Fuzzy Sets and Systems, vol. 127, pp. 185-197, 2002.

[19] S. Pramanik, "Bilevel programming problem with fuzzy parameters: a fuzzy goal programing approach," Journal of Applied Quantitative Methods, vol. 7, pp. 9-24, 2012.

[20] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proceedings of IEEE International Conference on Neural Networks, vol.4, pp. 1942-1948, 1995.

[21] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39-43, 1995.

[22] T. Lixin and W. Xianpeng, "A Hybrid Multiobjective Evolutionary Algorithm for Multiobjective Optimization Problems," IEEE Transactions on Evolutionary Computation, vol. 17, pp. 20-45, 2013.

[23] C. Wei-Neng, Z. Jun, H. S. H. Chung, Z. Wen-Liang, W. Wei-gang, and S. Yu-hui, "A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems," IEEE Transactions on Evolutionary Computation, vol. 14, pp. 278-300, 2010.

[24] L. Xiaodong and Y. Xin, "Cooperatively Coevolving Particle Swarms for Large Scale Optimization," IEEE Transactions on Evolutionary Computation, vol. 16, pp. 210-224, 2012.

[25] M. Jiménez, "Ranking fuzzy numbers through the comparison of its expected intervals," International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 04, pp. 379-388, 1996.

[26] S. Heilpern, "The expected value of a fuzzy number," Fuzzy Sets and Systems, vol. 47, pp. 81-86, 1992.

[27] M. Jiménez, M. Arenas, A. Bilbao, and M. V. Rodrı´guez, "Linear programming with fuzzy parameters: An interactive method resolution," European Journal of Operational Research, vol. 177, pp. 1599-1609, 2007.

[28] M. JIMÉNEZ, "Ranking fuzzy numbers through the comparison of its expected intervals," International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 4, pp. 379-388, 1996.

[29] V. Kalashnikov and R. Ríos-Mercado, "A natural gas cash-out problem: A bilevel programming framework and a penalty function method," Optimization and Engineering, vol. 7, pp. 403-420, 2006.

[30] C. Shi and R. Eberhart, "A modified particle swarm optimizer," in IEEE World Congress on Computational Intelligence, Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, pp. 69-73, 1998.