Received June 30, 2020, accepted July 18, 2020, date of publication July 21, 2020, date of current version August 5, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3011017

The Estimation of Particle Swarm Distribution Algorithm With Sensitivity Analysis for Solving Nonlinear Bilevel Programming Problems

GUANGMIN WANG[1], **(Member, IEEE), AND LINMAO MA**[2]
[1]School of Economics and Management, China University of Geosciences, Wuhan 430074, China
[2]School of Management, South-Central University for Nationalities, Wuhan 430074, China
Corresponding author: Guangmin Wang (wgm97@163.com)

ABSTRACT This paper proposes an estimation of particle swarm distribution algorithm (EPSDA) to solve the nonlinear bilevel programming problem (NBLP) by embedding the estimation distribution algorithm (EDA) into the particle swarm optimization (PSO). One Gaussian function is selected to construct the probability distribution for the superior candidate from the present population before executing the velocity and position update rule at each iteration in PSO. Thus, some new individuals viewed as an offspring population are generated from the probability distribution to replace some inferior particles in the current population for making up a new population. Therefore, this proposed algorithm combines PSO (the local search method) and EDA (the global search method) through updating the population to enhance the efficiency of solving NBLP. In experiments, we select four representative examples for linear, quadratic, nonlinear, high-dimensional nonlinear cases to carry out sensitivity analysis on parameters of the proposed algorithm. The results reveal that linearly decreasing inertia weight and adaptive acceleration coefficients are better than constant parameters. Furthermore, the multivariate Gaussian distribution can achieve better performance compared with the normal distribution. These four examples are also used to compare the performance of EPSDA with ones of separate PSO and EDA by setting constant parameters. The experiments show that EPSDA is better than separate PSO and EDA from both the quality of solution and the computational efficiency. The results on four high-dimensional non-convex nonlinear examples demonstrate feasibility and efficiency of EPSDA to solve the high-dimensional nonlinear bilevel programming from the iteration number and computational time.

INDEX TERMS The estimation of particle swarm distribution algorithm (EPSDA), nonlinear bilevel programming problem (NBLP), particle swarm optimization (PSO), estimation distribution algorithm (EDA).

I. INTRODUCTION

The bilevel programming (BLP) is an optimization problem with two hierarchical levels, e.g. the upper and lower level decision maker (or the leader and follower). Recently, Sinha *et al.* provided a comprehensive review on bilevel optimization from the basic principles to solution strategies (both classical and evolutionary) as well as a few potential application problems [1]. BLP has also been applied in big data fields [2]. Much intensive research has been done including theories, methods and application of bilevel programming problem because of its effectively modelling hierarchical decision making (DM) problems [3], [4]. However, BLP is difficult due to its non-convexity and non-differentiability and has been proved to be NP-Hard [5].

Considering independence on knowledge of the search space and robustness to deal with real problems, many meta-heuristic algorithms including genetic algorithm (GA), particle swarm optimization (PSO), artificial neural networks (ANN), an estimation of distribution algorithm, intuitionistic fuzzy method, evolutionary algorithm, fruit fly optimization algorithm (FOA) have been proposed to solve BLP compared with traditional mathematical algorithms [6], [7]. PSO, as one of the metaheuristic algorithms, has been proposed to solve BLP because its advantages like low computational complexity [8], [9]. However, particles do not use its collective experience but depend on their individual memory and peer influence to explore the

The associate editor coordinating the review of this manuscript and approving it for publication was Jagdish Chand Bansal.

search space, which leads to wasting computing power due to re-exploration of already known bad regions in the search space. Therefore, PSO is local-search intensive as a weak-cooperative search method due to its mechanism, which was demonstrated by Bergh and Engelbrecht [10] from the view point of stochastic optimization.

Recently, EDA has received increasing interest because of its advantages (e.g. a high convergent reliability and low time consumption) for complex optimization problems [11]. Researchers hybrid PSO and EDA to make best of their advantages and avoid their disadvantages. Iqbal and Montes de Oca [12] presented a generic extension to the PSO paradigm that allows a particle swarm to estimate the distribution of promising regions and thus ''learn'' from previous experience of the fitness landscape by exploiting the information it gains during the optimization process. The estimation of distribution improved particle approach, proposed by Kulkarni and Venayagamoorthy [13], can push the particle towards the estimated good regions in the search space by using swarm's collective memory. Wang [14] proposed a novel discrete PSO based on EDA for combinatorial optimization problems. The proposed algorithm combines the global statistical information collected from local best solution information of all particles and the global best solution information found in the whole swarm. Liu *et al.* [15] hybridized PSO with EDA to enable the sharing of information from the collective experience of the swarm. These researches performed better than extant PSO schemes because each particle may benefit from the global statistic gathered from all the local best positions (i.e., solutions) of a swarm. However, they have two limited factors, which motivate Ahn *et al.* [16] to present a novel framework of the estimation of particle swarm distribution algorithms for using the PSO dynamics but only when this approach fails in exploring the search space.

The above studies on the combination of EDA and PSO have showed its efficiency to solve traditional single-level problem because EDA enhances the global search capacity of PSO. It is very imperative to construct an algorithm obtaining a balance between the global search and local search abilities because solving BLP is more complicated than the single level problem. Therefore, we propose an estimation of particle swarm distribution algorithm (EPSDA) to solve nonlinear BLP (NBLP). Moreover, the difference of our algorithm with that in Ref. [16] is that a direct sampling of the probability distribution is used only when this approach fails besides our algorithm can solve BLP compared with single-level problems. In our algorithm, only the upper level decision variables are chosen as the particles, of which the fitness values are assigned as the upper level objective function values after the lower level problem is solved by GA with returning its solution to the upper level objective function at the given upper level decision variable. Moreover, when the particles are flying in line with the rules designed in the common PSO, the EDA is embedded into the evolutionary process. That is, before executing the velocity and position update rule at each iteration, EPSDA selects one Gaussian function just as EDA does to construct the probability distribution for the superior candidate selected from the present population. Thus, some new individuals viewed as an offspring population are generated from the probability distribution to replace some inferior particles in the current population. So, EDA is incorporated into the common PSO, based on the replacement. Eventually, the velocity and position of the particles are updated to move the particles to the new position. Hence, the difference between EPSDA and a canonical PSO is that the selected Gaussian function is evaluated and sampled to probabilistically update the present population s with new generated particles before the execution of the velocity and the position update rule.

Therefore, the main contribution of this paper is that an EPSDA is proposed to solve NBLP by embedding EDA into PSO. This hybrid algorithm has two distinctive characteristics: one is that it can strike a balance of global search and local search capacity by combining EDA's capability of learning dependencies between variables together with the memory or directional sense of PSO, the other is that the concept of elite strategy is adopted when selecting candidate particles and incorporating the population.

The rest of this paper is organized as follows: The main research findings about basic definitions and concepts for NBLP are firstly provided. Then, we describe the framework of EPSDA to solve NBLP after providing some background knowledge of PSO and EDA. The computational experiments are presented to demonstrate the algorithm's performances with the sensitivity analysis on the parameters. Finally, we conclude the paper with giving future research directions.

II. THE GENERAL FORMULATION AND BASIC CONCEPTS FOR NBLP

The general formulation of the nonlinear bilevel programming problem can be stated [7]:

$$(NBLP) \quad \min_{x\in X, y\in Y} F(x,y) \tag{1}$$

$$s.t. G(x,y) \leq 0, \tag{2}$$

where y is the solution to the following programming problem:

$$\min_{y\in Y} f(x,y) \tag{3}$$

$$s.t.\ g(x,y) \leq 0, \tag{4}$$

where $F, f : R^m \times R^n \to R$ are the upper and lower level objective functions, respectively. The vector-valued functions $G : R^m \times R^n \to R^p$ and $g : R^m \times R^n \to R^q$ are the upper and lower level constraints functions, respectively. $x \in X \subset R^m$ and $y \in Y \subset R^n$ are the decision variables decided by the upper and lower level decision makers, respectively. The sets X and Y set additional restrictions for the involved variables.

Let $S = \{(x, y) \in X \times Y : G(x, y) \leq 0, g(x, y) \leq 0\}$ denote the constraint region of NBLP. Here, S is supposed to be nonempty and compact to guarantee that NBLP is well posed. $S(X) = \{x \in X : \exists y \in Y, (x, y) \in S\}$ denotes the projection of S onto the leader's decision space. For any fixed $\hat{x} \in S(X)$, the lower level problem can be formulated as follows:

$$P(\hat{x}) = \min_{y \in S(\hat{x})} \bar{f}(\hat{x}, y) \tag{5}$$

where $S(\hat{x}) = \{y \in Y : g(\hat{x}, y) \leq 0\}$ denotes the feasible solution set to the lower level problem for each fixed $\hat{x}$. For each x selected by the leader, we assume that the follower has some responses, i.e., $P(x) \neq \emptyset$. Bard [17] used examples to illustrate the difficulties that often arise when $P(x)$ is multi-valued and discontinuous. The readers can refer to [18] for how to do when $P(x)$ is multivalued. In this paper, we restrict the situation that a unique solution to the lower level problem exists for each fixed $x \in S(X)$. That is, $P(x)$ is the point-to-point map. Therefore, NBLP is to optimize $F(x, y)$ over the inducible region,

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

Consequently, NBLP can be restated as

$$\min_{x,y}\{F(x, y) : (x, y) \in IR\}. \tag{6}$$

Based on above notations, the feasible and optimal solution to NBLP are defined as follows [7]:

Definition 1: $(\bar{x}, \bar{y}) \in S$ is named the feasible solution to NBLP, if $(\bar{x}, \bar{y}) \in IR$.

Definition 2: $(x^*, y^*) \in IR$ is named the optimal solution to NBLP, if $F(x^*, y^*) \leq F(\bar{x}, \bar{y}) \forall (\bar{x}, \bar{y}) \in IR$.

III. THE DEVELOPMENT OF THE ESTIMATION OF PARTICLE SWARM DISTRIBUTION ALGORITHM

In this section, we propose the estimation of particle swarm distribution algorithm in details with its steps after brief introduction to PSO and EDA, separately.

A. THE BRIEF INTRODUCTION TO THE PARTICLE SWARM OPTIMAIZATION

The particle swarm optimization is one of population-based algorithms designed according to the animals' social behavior [19], [20]. PSO is a moving process that each particle flies through the multidimensional search space while updating its velocity and position according to both its own and the entire swarm's best knowledge.

Initially, a swarm of N possible particles (i.e., solutions) are randomly generated in an D-dimensional search space. Each particle then determines its movement through the search space by combining some aspect of the history of its own current and best locations with those of one or more members of the swarm, with some random perturbations. The next iteration takes place after all particles have been moved. Eventually the swarm, like a flock of birds collectively foraging for food, is likely to move close to an optimum of the fitness function.

A D-dimensional vector, $X_i^k = (x_{i1}^k, x_{i2}^k, \cdots, x_{iD}^k)$ can be used to represent the i-th particle of the swarm in the k-th generation. Another D-dimensional vector $V_i^k = (v_{i1}^k, v_{i2}^k, \cdots, v_{iD}^k)$ can be used to represent the velocity of this particle in the k-th generation. $pbest^k = (p_1^k, p_2^k, \cdots, p_D^k)$ can denote the best previously visited position of the particles in the k-th generation. $gbest = (g_1, g_2, \cdots, g_D)$ can denote the best previously visited position of the swarm. The steps of the original form of particle swarm optimization are as follows:

Step 1(Initializing). Initialize random positions and velocities on D dimensions for particles, and set the iteration $t = 0$,

Step 2(Evaluating). Evaluate each the particle by computing its fitness function value,

Step 3(Updating). Compare current value of each particle's fitness function with $pbest^k$: If current value $< pbest^t$ then $pbest^t$ = current value. Compare current value of each particle's fitness function with $gbest$: If current value $< gbest$ then $gbest$ =current value,

Step 4(Generating). Update the velocity and position for the i-th particle in the $(t + 1)$-th generation according to the following equations:

$$V_i^{t+1} = \omega V_i^t + c_1 r_1 (pbest^t - X_i^t) + c_2 r_2 (gbest - X_i^t) \tag{7}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{8}$$

where ω is the positive inertia weight, which determines the influence of the particle' past velocity on its present one; c_1 and c_2 are the cognitive and social learning factors, respectively; They determine the effect of the local and global best solutions' velocity on the particle's present one, respectively; $r_1, r_2 \in [0, 1]$ are two uniformly distributed random numbers.

Step 5(Stopping criterion). If a criterion is met, stop; otherwise go to Step 2.

The constant V_{max} was implemented to prevent the particle from leaving out of the searching space. Equations (7) -(8) reveal that each particle traverses the search space through balancing its own and the social experiences [19]–[21]. Therefore, a particle is updated according to its previous velocity, its own best information, and the whole swarm's global best knowledge. Thus, the values of inertia weight and learning factor have influence on the computing efficiency. More details about PSO with studies on the optimal inertia weight and learning factors can be referred to [22]–[24].

B. THE BRIEF INTRODUCTION TO ESTIMATION OF DISTRIBUTION ALGORITHMS

Estimation of distribution algorithm introduced by Mühlenbein and Paa β [11] as an optimization technique has received much attention for the past few years, and successfully applied to optimization, engineering, cluster analysis, machine learning and design problems [16], [25]. EDAs use information from the optimization process to build probabilistic models of the distribution of good regions in the search space and use these models to generate new solutions. In general, EDA will stop when satisfying the termination criteria through iteration of the following steps:

Step 1(Initializing). Initialize the population $P(0)$, and set the iteration $t = 0$,

Step 2(Selecting). Select population of promising $S(t)$ from $P(t)$,

Step 3(Building). Build probabilistic model $M(t)$ from $S(t)$,

Step 4(Sampling). Sample $M(t)$ to generate new candidate solutions $O(t)$,

Step 5(Incorporating). Incorporate $O(t)$ into $P(t)$, $t = t + 1$

Step 6(Stopping criterion). If a criterion is met, stop; otherwise go to step 2.

Readers can also refer to [26], [27] for more details about EDAs.

C. THE PROPOSED ESTIMATION OF PARTICLE SWARM DISTRIBUTION ALGORITHM

In this subsection, we propose the estimation of particles swarm distribution algorithm to solve the bilevel programming problem (1)-(4). EPSDA works as a canonical PSO described above but with some modifications which integrate EDA into PSO, along the lines of the following framework.

In the proposed EPSDA, the upper level decision variables are chosen as the particles and the upper level objective function value is assigned to be the fitness value of each particle. For the evaluation on the particles, the fitness value is calculated after the lower level problem is solved by GA at the given upper level decision variable and the obtained lower level solution is transferred to the upper level. Here, the lower level problem is simplified to be a common nonlinear programming problem, which can be solved by diverse methods when the upper level decision variable is determined. Thus, GA the Matlab optimization toolbox is chose as the approach to solve the lower level problem because we code EPSDA in Matlab.

In EPSDA, before executing the velocity and position update rule shown in Equations (7) and (8), at each iteration EPSDA selects one Gaussian function just as EDA does to exploit the probability distribution for the promising region of the superior candidate in the current population. And then some new particles are generated from the probability distribution to form an offspring population. Subsequently, the population is replaced by incorporating the offspring and parent populations. In addition, the concept of elite strategy is adopted by selecting the better candidate particles from the offspring population when incorporating the population. Therefore, EDA is embedded into PSO based on the replacement so that the presented algorithm can achieve a balance between global search and local search ability. The velocity and position of new population are updated by Equations (7) and (8).

The detailed steps of the estimation of particle swarm distribution algorithm are listed as follows:

Step 0(Setting). The population size (the number of particles) is set to be *PopSize*. The Maximal velocity, V_{max}, the positive inertia weight ω, and two learning factors, c_1 and c_2 are set. The maximum number of iterations is set as T_{max}. The counter of iteration is set to be $t = 0$. The velocity of each particle V_i^t in the initial population $Pop(0)$ is randomly generated in interval $[0, V_{max}]$.

Step 1(Initializing). The initial population $Pop(0)$ is generated according to the following procedure: The particle $X_i^t = (x_{i1}^t, x_{i2}^t, \cdots, x_{im}^t)^T$ is randomly selected in $S(X)$. For the fixed $X_i^t \in S(X)$, the lower level problem (5) can be solved by GA. If the optimal solution is obtained, denoted by $Y_i^t(X_i^t) = (y_{i1}^t, y_{i2}^t, \cdots, y_{in}^t)^T$, the point $Z_i^t = (X_i^t, Y_i^t)^T$ is the feasible point of BLPP according to the definition 1. X_i^t is called a feasible particle. After generating *PopSize* such particles to make up the particle population $Pop(0)$, go to step 2.

Step 2(Evaluating). We compute the objective function value of the problem (6) at the solution Z_i^t as its fitness value of the particle X_i^t. Sort the particles in the population $Pop(t)$ as the ascending of the particles' fitness values.

Step 3(Updating). Set the first particle's position as the $pbest^t$. Compare the fitness value at the position $pbest^t$ with the one at the global best position $gbest$. Choose the better one as $gbest$.

Step 4(Selecting). Using the truncation selection with threshold τ to select $N(< PopSize)$ superior candidate solutions from $Pop(t)$ to form the parent population $Parent(t)$ based on the particles' fitness function values.

Step 5(Building). Build the distribution of the selected parent population $Parent(t)$ to establish a probabilistic model $M(t)$.

The one-dimensional normal densities to factorize the joint probability density function are as follows:

$$F(X_i^t|\mu,\sigma) = \prod_{j=0}^{m-1} N(x_{ij}^t;\mu_j,\sigma_j) = \prod_{j=0}^{m-1} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{1}{2}(\frac{x_{ij}^t-\mu_j}{\sigma_j})^2}, \tag{9}$$

where m is the dimension of the vector, x_{ij}^t denotes the j-th component of the vector, μ_j and σ_j denote the mean value and the standard deviation (SD) of the j-th component, respectively. For each component, these two parameters are estimated from the population, $S(t)$, employing the following two equations:

$$\mu_j = \frac{1}{K}\sum_{i=0}^{K-1} x_{ij}^t, \tag{10}$$

$$\sigma_j = \sqrt{\frac{1}{K-1}\sum_{i=0}^{K-1}(x_{ij}^t - \mu_j)^2}, \tag{11}$$

where K is the amount of samples.

In each generation, these parameters are estimated according to the selected superior candidates, in such way as to extract the global statistical information of the previous population. Hence, the parameters are updated adaptively and accordingly in each iteration.

Step 6(Sampling). Sample *PopSize* proper candidates to be the offspring population *Offspring*(t) according to the model $M(t)$.

Based on the probabilistic model built for the parent population, the proposed algorithm takes samples to generate individuals randomly. To make these newly generated candidates appropriate and to make the algorithm efficient, we set up a regulation for them. In other words, the new particle can be selected as a candidate solution, only when it is feasible. Otherwise, the algorithm should abandon it and make another sample operation. Repeat the generating and "check-up" procedures until *PopSize* new proper particles are generated.

Step 7(Incorporating). Choose $(PopSize - N)$ relatively better candidates from the population *Offspring*(t) based on their fitness values. The chosen best offspring candidates and the parent population *Parent*(t) make up the population *Pop*$(t+1)$.

Step 8(Generating). For all particles in the population *Pop*$(t+1)$, the velocity of each particle in the population *Pop*$(t+1)$ and its new position will be assigned according to Equations (7) and (8). For each particle X_i^{t+1}, the lower level problem (5) is solve to Y_i^{t+1}, which is the optimal solution. The point $Z_i^{t+1} = (X_i^{t+1}, Y_i^{t+1})^T$ is the feasible point of BLPP according to the definition 1. We compute the objective function value of the problem (6) with the point Z_i^{t+1} as its fitness value of the particle X_i^{t+1}.

Step 9(Stopping criterion). $t = t + 1$. The stopping criterion is that the iteration number is bigger than T_{max} or the best fitness value doesn't change in five iterations. Go to Step 2, if the stopping criterion is not met.

Step 10(Outputting). Output the optimal particle, compute and output the upper and lower levels' objective function values.

Note: We embed EDA into PSO through Step 4-7 regarded as the replacement operation for the population. EDA can reflect the distribution information of superior solutions by constructing promising probabilistic models. Exploiting the probabilistic models to traverse the search space, EDA efficiently evolves the whole population towards the promising regions of the global optimum. PSO can achieve local searching for the solutions by combining its own best information and the best knowledge of the entire swarm. The integration of these two algorithms enables the proposed EPSDA to have both the local searching ability of PSO and the global search ability of EDA.

IV. NUMERICAL EXPERIMENTS

In this section, we illustrate the numerical experiments on the selected examples from [28]–[30] to demonstrate EPSDA' feasibility and performance. First, we use a small test example to demonstrate the feasibility of EPSDA with constant parameters. Second, we explain how to select test examples to make further numerical experiments for verifying the performances of the proposed algorithm. Third, we carry out sensitivity analysis on parameters of the proposed algorithm using Examples 1-4. Four, numerical experiments on these 4 examples are used to compare the computational efficiency of EPSDA with PSO and EDA from aspects of the quality of solution, iteration number and the computational time. Finally, the high-dimensional cases are solved for further tests on the performance of EPSDA. In each experiment exclude Example 1, we execute the proposed EPSDA (as well as PSO and EDA) with 20 independent runs on corresponding examples. We record upper level objective function at the corresponding solution, as well as the computational time and iterative numbers. Then, we analyze descriptive statistics values of the data such as the best, worst, mean and standard deviation (SD) to demonstrate the performance of the proposed EPSDA. The best (or worst) means the minimal (or maximal) upper level objective function value, iteration number or computational time.

In all numerical experiments, a personal computer with Intel Core 2 Duo 1.80 GHz CPU, 4.0 GB RAM, and Windows XP operating system is used for all test examples. In addition, Matlab with the version 9.2 (R2017a) is used for coding the algorithm. For convenience, the lower level problem is solved by GA in genetic arithmetic toolbox [55] to make the presented algorithm competent for more problems and to minimize the effect on the performance of the proposed algorithm. As for GA employed in this paper, the parameter setting is set as follows: Each individual is stochastically initialized utilizing real-number encoding; the maximal number of iterations is 200 and the initial population size is 50; roulette selection, scattered crossover, and uniform mutation are implemented and the crossover fraction and mutation fraction is 0.8 and 0.01, respectively. Each experiment is terminated either when the maximal number of iterations is exceeded or when the objective of every individual doesn't change in 10 iterations.

A. A TEST EXAMPLE

In this subsection, we use the following test example to demonstrate the feasibility of this propose algorithm:

Example [28]:

$$\min_{x,y} F(x,y) = (x-1)^2 + (y-1)^2,$$

where y solves the following problem:

$$\min_{y} f(x,y) = 0.5y^2 + 500y - 5xy$$

The parameters in the proposed EPSDA are all constant. They are chose as follows: The population size is set $PopSize = 50$, the positive inertia weight $\omega = 0.729$, two learning factors $c_1 = c_2 = 2.05$, the truncation selection with threshold $\tau = 0.3$ and the multivariate Gaussian probability model is adopted. The maximal number of iterations is set $T_{max} = 50$.

We execute the proposed EPSDA on this problem. The solution $(x^*, y^*) = (10.0171, 0.8558)$ is obtained and the corresponding upper and lower level objective function values are $F^*(x^*, y^*) = 81.3292$ and $f^*(x^*, y^t) = -0.3662$,

TABLE 1. Types of the test examples.

No.	Type	Scale	Functions	Source
Ex.1	Linear	m=3, n=1	Convex, differential	[46]
Ex.2	Quadratic	m=1, n=1	Convex, differential	[45]
Ex.3	Nonlinear	m=2, n=2	Non-convex, non-differential	[47]
Ex.4	Nonlinear	m=10, n-10	Non-convex, non-differential	[47]
Ex.5	Nonlinear	m=10, n-10	Non-convex, non-differential	[47]
Ex.6	Nonlinear	m=10, n-10	Non-convex, non-differential	[47]
Ex.7	Nonlinear	m=10, n-10	Non-convex, non-differential	[47]
Ex.8	Nonlinear	m=10, n-10	Non-convex, non-differential	[47]

respectively. The upper level objective function value changes as the iterative number as Figure 1:

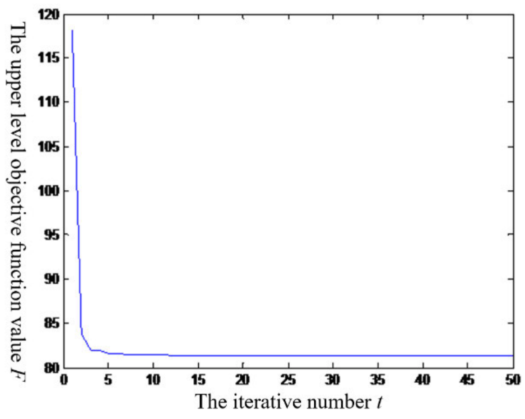


FIGURE 1. The convergence procedure of the test example.

The above figure shows that the proposed algorithm is convergent and feasible to solve the nonlinear bilvel programming problem. In [28], the results computed by the penalty function (a traditional method) for this test example are $(x^*, y^*) = (10.02, 0.82)$ as well as $F^*(x^*, y^*) = 81.33$. The EPSDA algorithm can give comparable results to the traditional method.

B. THE SELECTED EXAMPLES

Wang *et al.* [29] used 31 benchmark problems for simulations on the proposed evolutionary algorithm based on a new constraint-handling scheme for solving nonlinear bilevel programming. Wan *et al.* [30] selected 16 examples including linear, quadratic, and other nonlinear cases to test the proposed estimation of distribution algorithm for solving nonlinear bilevel programming. All benchmark problems are classified into linear, quadratic, nonlinear, high-dimensional nonlinear case. Thus, we select eight test examples (Appendix for details about test examples) from references to make numerical experiments on the proposed EPSDA and test its performances. Examples 1-4 are selected from references as the representative examples for linear, quadratic, nonlinear, high-dimensional nonlinear case, respectively. These four examples are used to carry out sensitivity analysis of the parameters of the proposed algorithm. Furthermore, they are also used to compare the performances of the proposed EPSDA with those of PSO and EDA separately. To further efficiency of the proposed algorithm, Examples 5-8, which are all high-dimensional non-convex and non-differentiable nonlinear bilevel programming, are solved by the proposed EPSDA. Table 1 lists the characteristics of these examples as follows.

Some essential statements about the analysis result for all examples are presented as follows. In our algorithm, the fitness values of the particles are assigned as the upper level objective function values, so the upper level objective values reflect the performance of our algorithm directly and appropriately. For this reason, we analyze the descriptive statistics values of the upper level to illustrate the performance of our algorithm in subsection "Sensitivity analysis of the parameters in EPSDA". Furthermore, the solution quality is taken as a measure of the performance and efficiency of the proposed algorithm, so subsection "Comparison of EPSDA with PSO and EDA" and "The numerical experiments on the high-dimensional case" present the upper and lower decision variables (or the solutions) to compare the accuracy of the solutions with the corresponding optimal results obtained in the existing references. Meanwhile, three typical indicators, i.e. the upper level solutions, the computational time and the iterative numbers, are recorded in the following subsections to evaluate the performance of the proposed algorithm and compare it with separate EDA and PSO, as well as the extant literatures from the aspects of the quality of solution and efficiency. Here the upper level solutions represent the solution quality in view of the accuracy while the other two values denote the efficiency considering the convergent rate and computational complexity. These three indicators are as the performance measure.

C. SENSITIVITY ANALYSIS ON THE PARAMETERS IN EPSDA

In this subsection, we carry on the sensitivity analysis (SA) on the parameters in EPSDA including the population

TABLE 2. Parameters for SA.

Parameters	$PopSize$	ω	c_1, c_2	Probabilistic model	T_{max}	τ
SA1	50;100;150;200;250	0.729	2.05	Multivariate	100	0.3
SA2	100	0.5;0.729;0.9; $\omega(t)=\omega^{max}-\frac{\omega^{max}-\omega^{min}}{T_{max}}\times t$ $\omega(t)=\omega^{min}\times\frac{\omega^{max}}{\omega^{min}}^{\frac{1}{1+10*t/T_{max}}}$				
SA3		0.729	(1.5,1.5) ;(1.5,2.05) ;(2.05,1.5) ;(2.05,2.05); $(c_1^{max}-\frac{c_1^{max}-c_1^{min}}{T_{max}}\times t,$ $c_2^{min}+\frac{c_2^{max}-c_2^{min}}{T_{max}}\times t)$			
SA4			2.05	Normal; Multivariate		
SA5				Multivariate		
Remark		$\omega^{max}=0.9, \omega^{min}=0.4$	$c_i^{max}=1, c_i^{min}=0.4; i=1,2$			

TABLE 3. The descriptive statistics values of the upper level objective function value with different population size.

	50				100				150				200				250			
	Best	Worst	Mean	SD	Best	Worst	Mean	SD	Best	Worst	Mean	SD	Best	Worst	Mean	SD	Best	Worst	Mean	SD
1	-29.1980	-29.1657	-29.2935	0.011444	-29.1996	-29.1809	-29.1939	0.006844	-29.1944	-29.1233	-29.1865	0.028659	-29.19	-29.1893	-29.1967	0.00338	-29.1980	-29.1768	-29.1848	0.006825
2	100.0004	100.8612	100.1788	0.234694	100.0002	100.8099	100.135	0.222911	100.0001	100.4235	100.0517	0.105162	100.00293	101.4276	100.36212	0.348363	100.0003	100.3065	100.0415	0.71176
3	0	0.165882	0.020273	0.038107	0	0.018929	0.003862	0.005463	0	0.030422	0.00496	0.008114	0	0.030422	0.00496	0.008114	0	0.023191	0.001758	0.005151
4	1.31E-05	2.02E-05	1.62E-05	1.84E-06	1.61E-05	2.12E-05	1.89E-05	1.61E-06	1.41E-05	3.40E-02	1.84e-03	7.59E-03	1.46E-05	2.02E-05	1.76E-05	1.61E-06	1.36E-05	8.57E-02	4.31E-03	1.92E-02

size *PopSize*, inertia weight ω, learning factors c_1, c_2 and the probabilistic model. We analyze sensitivity of each parameter by varying it while other parameters are fixed. The schemes of choosing parameters to carry out SA are listed in Table 2.

1) SENSITIVITY ANALYSIS OF THE POPULATION SIZE

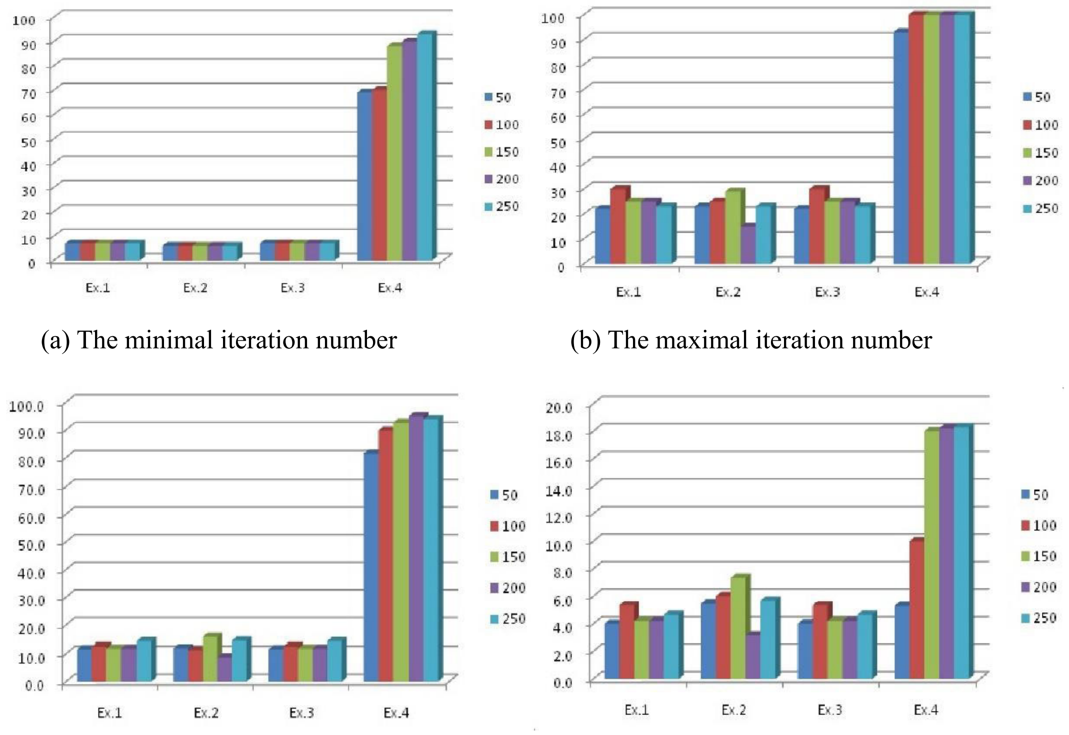
Population size affects performance of the proposed EPSDA. Too few particles prompt the algorithm to get trapped in local optima, while too many particles slow down the algorithm. There is no exact rule in literature for selecting appropriate population size suit for the instances because the optimal size depends on the solved instance. We research the sensitivity analysis of the population size by choosing *PopSize*= 50, 100, 150, 200, 250, respectively. Table 3 lists descriptive statistics values of the upper level objective function value for Examples 1-4 when executing 20 times running by choosing different population sizes. Figures 2 and 3 illustrate the descriptive statistics values of the iteration number and computational time, respectively.

Table 3 shows that the solutions are not sensitive to the population size, however, the computational cost (iteration number and computational time) are all increasing as the population size increases from Figures 2 and 3, which has been reported in [22]. Thus, large population size will increase computational efforts and may make slow convergence.

2) SENSITIVITY ANALYSIS OF INERTIA WEIGHT

The inertial weight can balance the global and local search abilities. A larger inertia weight is better for global search, and a smaller inertia weight is more suitable for local search. Researchers advocated that the value of inertia weight should be large in the exploration state and small in the exploitation state [31]–[33]. Different procedures for setting inertia weight are fixed inertia weight, fuzzy adaptive, linearly decreasing, multi-stage linearly decreasing, linearly increasing, non-linear, random, chaotic, exponential, Gaussian, parallel and simulated annealing inertia weight [34]. A comparative study on different types of inertia weight and linearly decreasing one achieves best performance [35]. We compare the performance of fixed (0.5,0.729 and 0.9), linearly and nonlinear decreasing inertia weight (LD and NLD). Table 4 lists descriptive statistics values of the upper level objective function value for Examples 1-4. Figures 4 and 5 illustrate descriptive statistics values of the iteration number and computational time, respectively.

Table 4 shows that the linearly deceasing inertia weight can obtain the better solution not only the best case but also the



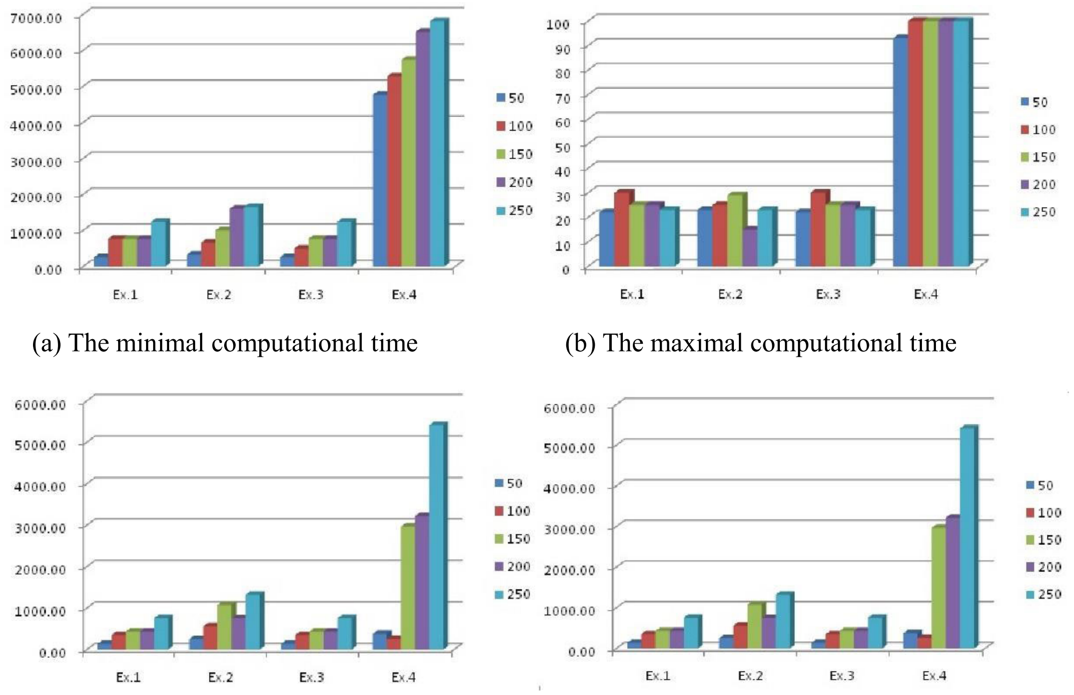
(a) The minimal iteration number

(b) The maximal iteration number

(c) The mean of it iteration number

(d) The SD of iteration number

FIGURE 2. **The descriptive statistics values of the iteration number when choosing different population size.**



(a) The minimal computational time

(b) The maximal computational time

(c) The mean of computational time

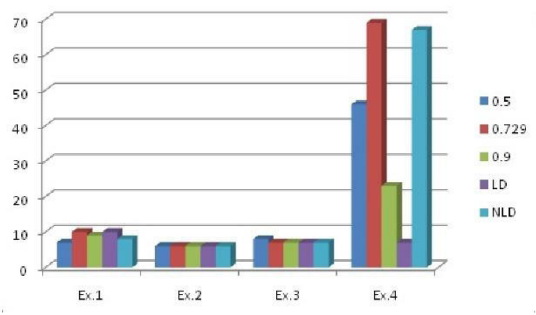
(d) The SD of computational time

FIGURE 3. **The descriptive statistics values of the computational time when choosing different population size.**

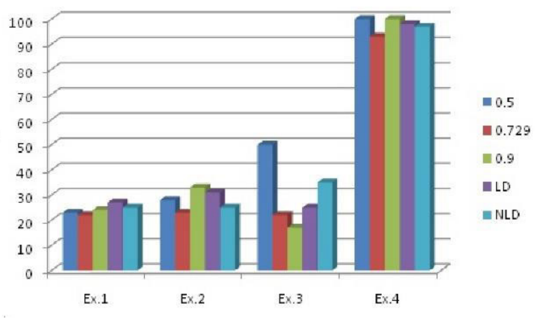
worst case. Moreover, the mean and SD are the best of all the situations. Similarly, the linearly decreasing inertia weight can achieve the better performance on the iteration number and computational time from Figures 4 and 5.

TABLE 4. The descriptive statistics values of the upper level objective function value with different inertia weight.

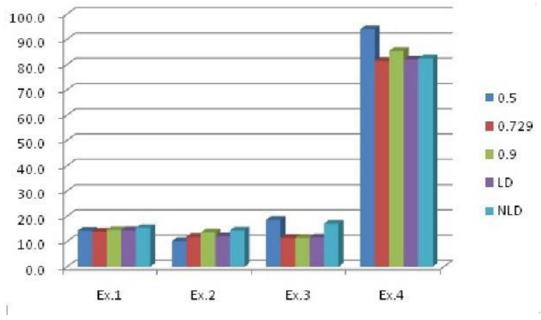
	0.5				0.729				0.9				LD				NLD			
	Best	Worst	Mean	SD	Best	Worst	Mean	SD	Best	Worst	Mean	SD	Best	Worst	Mean	SD	Best	Worst	Mean	SD
1	-29.1998	-29.1557	-29.1915	0.011444	-29.1995	-29.1709	-29.1929	0.006844	-29.1992	-29.1033	-29.1855	0.024659	-29.2	-29.1873	-29.1977	0.00338	-29.1999	-29.1668	-29.1948	0.007825
2	100.0007	100.6342	100.1789	0.187549	100.0004	100.8612	100.1788	0.234694	100.0002	100.5699	100.0991	0.156428	100	100.5355	100.102	0.10047	100	100.704	100.1173	0.175061
3	0	0.018111	0.002116	0.004475	0	0.165882	0.020273	0.038107	0	0.050027	0.005823	0.013612	0	0.013024	0.00092	0.001034	0	0.01564	0.001697	0.002868
4	1.45E-05	2.56E-03	1.54E-04	5.67E-04	1.31E-05	2.02E-05	1.62E-05	1.84E-06	1.60E-05	1.89E-01	1.53E-02	4.60E-02	1.14E-07	1.9E-05	5.46E-06	2.44E-06	1.02E-05	2.44E-05	1.72E-05	3.89E-06



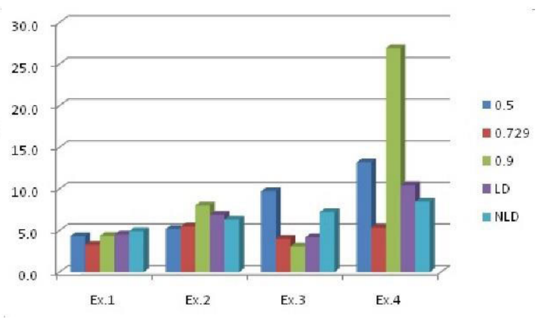
(a) The minimal iteration number



(b) The maximal iteration number



(c) The mean of iteration number



(d) The SD of iteration number

FIGURE 4. The descriptive statistics values of iteration number when choosing different inertia weight.

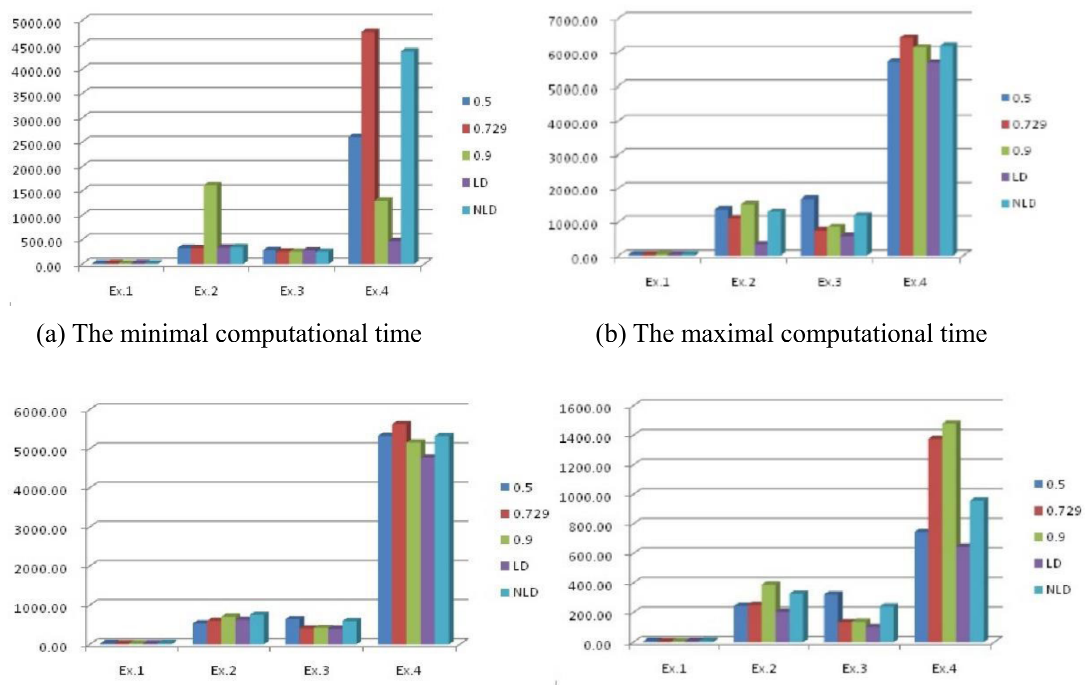
3) SENSITIVITY ANALYSIS OF LEARNING FACTORS

Parameter c_1 represents the self-cognition“ that pulls the particle to its own historical best position, helping explore local niches and maintaining the diversity of the swarm. Parameter c_2 represents the social influence” that pushes the swarm to converge to the current globally best region, helping with fast convergence [32], [36]. When c_1 is larger, the current particle is highly affected by its previous best particle. When c_2 is larger, the current particle is highly influenced by the global best particle. These are two different learning mechanisms and should be given different treatments in different evolutionary states [37]. Bao and Mao [38] pointed out that a self-adaptive mechanism outperformed basic PSO variants for most of test functions in terms of global optimality, convergence speed and solution accuracy. We use five schemes of choosing learning factors (See Row “SA3” in Table 2 for details) to analyze their sensitivity on the results. Table 5 lists the descriptive statistics values of the upper level objective function value for Examples 1-4 when choosing different schemes of learning factors. Figures 6 and 7 illustrate the descriptive statistics values of the iteration number and computational time, respectively.

Obviously, dynamically adjusting learning factors can obtain better solutions than other schemes from Table 5. Similarly, it also outperforms other schemes in term of the iteration number and computational time for the test examples from Figures 6 and 7.

4) SENSITIVITY ANALYSIS OF PROBABILISTIC MODEL

We compare the performance of two different probabilistic models (normal distribution and multivariate Gaussian distribution) (See Row “SA4” in Table 2 for details) to analyze the probabilistic model’s influence on the results. Table 6 lists the descriptive statistics values of the upper level objective



(a) The minimal computational time

(b) The maximal computational time

(c) The mean of computational time

(d) The SD of computational time

FIGURE 5. The descriptive statistics values of the computational time when choosing different inertia weight.

TABLE 5. The descriptive statistics values of the upper level objective function value with different learning factors.

	LF1:(1.5,1.5)				LF2:(1.5,2.05)				LE3:(2.05,1.5)				LD				NLD			
	Best	Worst	Mean	SD	Best	Worst	Mean	SD	Best	Worst	Mean	SD	Best	Worst	Mean	SD	Best	Worst	Mean	SD
1	-29.19994	-29.18671	-29.19696	0.00329	-29.19975	-29.16125	-29.19516	0.0093085	-29.19994	-29.18068	-29.195	0.0052114	-29.1995	-29.17091	-29.19286	0.0068444	-29.19291	-29.19291	-29.194	1.46E-02
2	100	100.5742	100.13649	0.177958	100.00013	101.2018	100.1452	0.2765362	100.00005	100.44316	100.13032	0.1402159	100.0004	100.86123	100.17885	0.2346941	100	100.00145	100.--1823	0.002224
3	0	0.0478826	0.0101318	0.013398	0	0.1382258	0.0117219	0.0308683	0	0.0488035	0.0075014	0.0118172	0	0.1658821	0.0202735	0.0381065	0	0	0	0
4	1.70E-05	5.7E-01	4.40E-02	1.39E-01	1.53E-05	9.91E-02	5.18E-03	2.21E-02	1.32E-05	2.65E-01	1.95E-02	6.40E-02	1.31E-05	2.02E-05	1.62E-05	1.84E-06	1.53E-06	9.91E-05	5.18E-05	2.21E-06

function value for Examples 1-4 when choosing different probabilistic model. Figures 8 and 9 illustrate the descriptive statistics values of the iteration number and computational time, respectively.

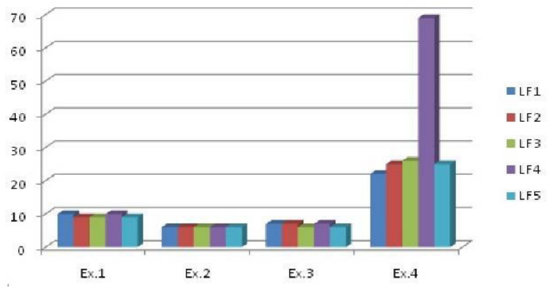
Compared with the normal distribution, the multivariate Gaussian distribution can achieve better performance from both accuracy of the solution and the computational efficiency (iteration number and computational time) from Table 6 and Figures 8 and 9.

D. COMPARISON OF EPSDA WITH PSO AND EDA

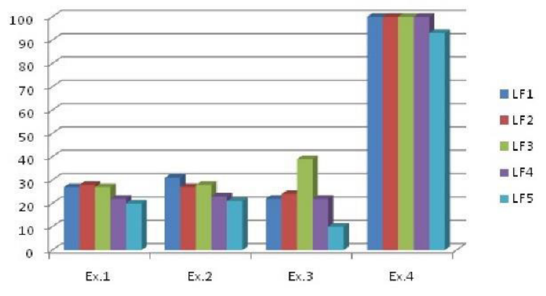
We compare the proposed algorithm with PSO and EDA from the following aspects such as the quality of the solution, iteration number and computational time. First, we solve Examples 1-4 by the proposed algorithm with the better parameters as Row "SA5" in Table 2. We list descriptive statistics values of the data in Tables 7-9. Second, we solve Examples 1-4 by PSO and EDA. We execute both algorithms for 20 independent runs on each problem. Parameters in PSO and EDA are as those in EPSDA.

To make a comparison between the proposed EPSDA with existing algorithms. We record the results by this proposed EPSDA with ones by existing algorithms and list all the results in Table 10. In addition, in Tables 7 and 10, Column "Ref." sequentially lists the results obtained by Wang *et al.* for Example 1 [29], Oduguwa and Roy for Example 2 [28] and Wan *et al.* for Examples 3-4 [30].

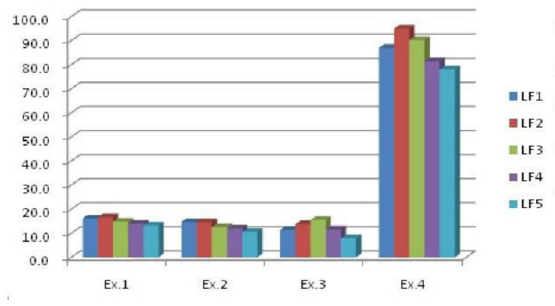
Tables 7-9 show that the proposed algorithm can obtain the solution efficiently in terms of accuracy and quality of the solutions as well as the iteration number and computational time. We also observe that the iteration number and the computational time increase as the scale and complexity



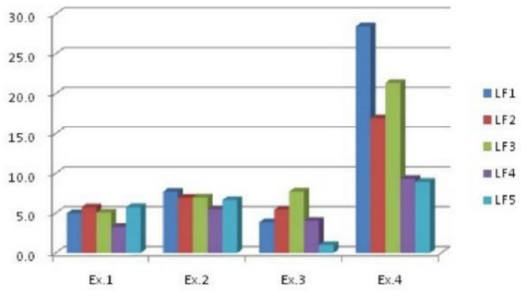
(a) The minimal iteration number



(b) The maximal iteration number

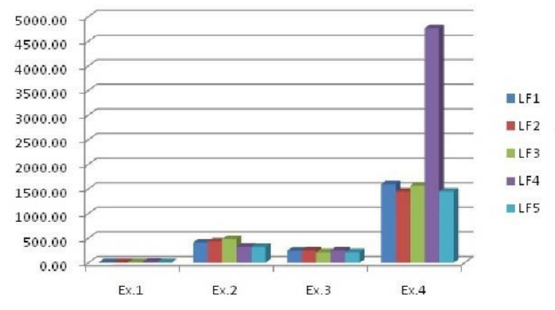


(c) The mean of iteration number

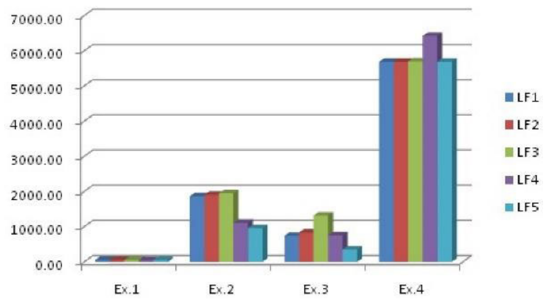


(d) The SD of iteration number

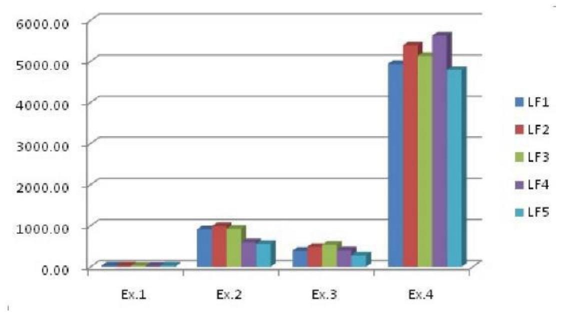
FIGURE 6. The descriptive statistics values of iteration number when choosing different learning factors.



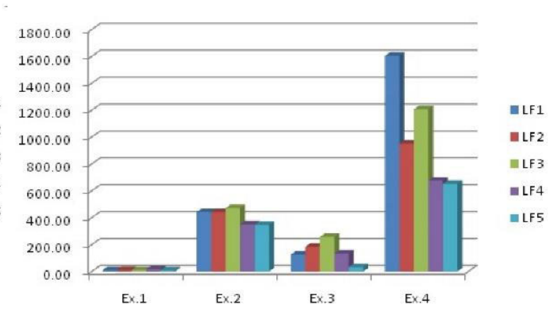
(a) The minimal computational time



(b) The maximal computational time



(c) The mean of computational time



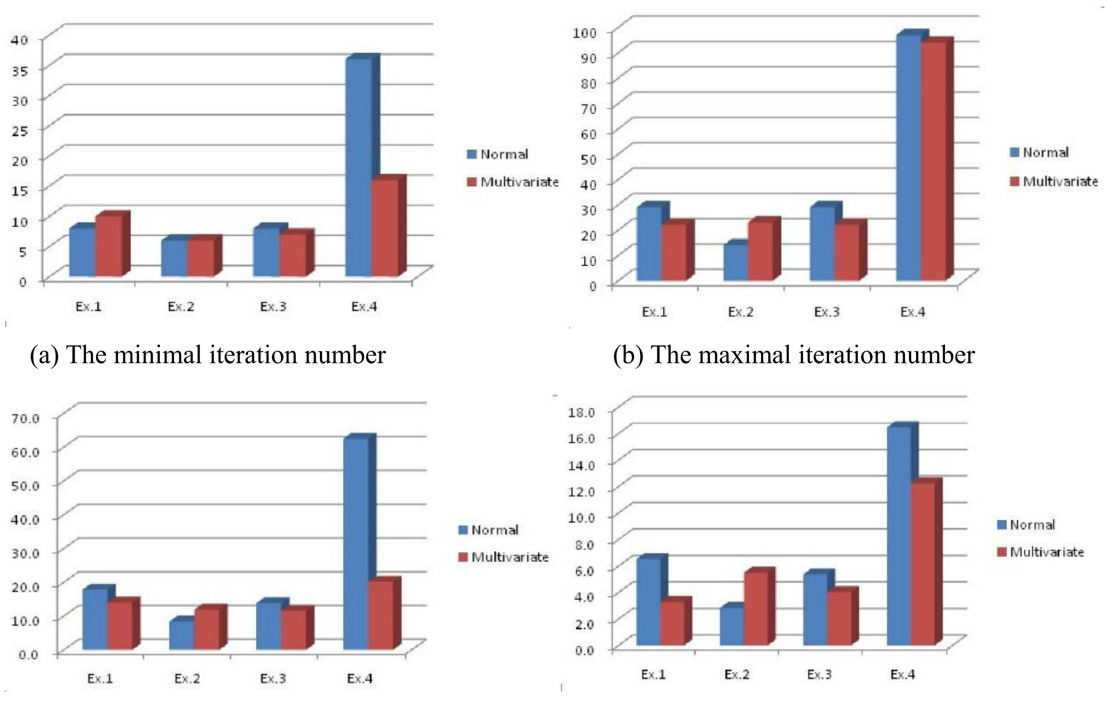
(d) The SD of computational time

FIGURE 7. The descriptive statistics values of the computational time when choosing different learning factors.

of the problems increase. We also find it is time-consuming to solve the lower level problem. Therefore, it will take more time to address BLPP with high-dimensional decision variable when we use GA for obtaining the solution to the

TABLE 6. The descriptive statistics values of the upper level objective function value with different.

	Normal				Multivariate			
	Best	Worst	Mean	SD	Best	Worst	Mean	SD
1	-29.19946206	-29.17090505	-29.19286299	6.84E-03	-29.19996742	-29.17261732	-29.1934037	4.12E-03
2	100.0131378	108.4577089	102.2617272	2.43777585	100.0003679	100.861234	100.178849	0.234694145
3	2.84E-05	0.047213821	0.008229792	0.011355557	0	0.005882054	0.002273498	0.008106523
4	0.022797417	4.505928272	2.374613782	1.228790617	1.02E-05	2.44E-05	1.72E-05	3.89E-06



(a) The minimal iteration number

(b) The maximal iteration number

(c) The mean of iteration number

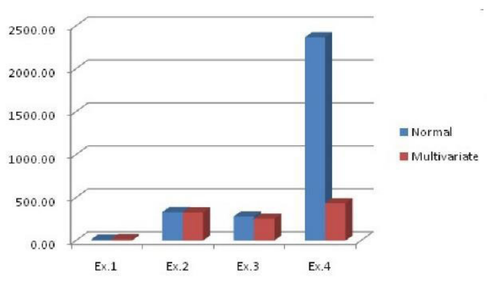
(d) The SD of iteration number

FIGURE 8. The descriptive statistics values of iteration number when choosing different probabilistic model.

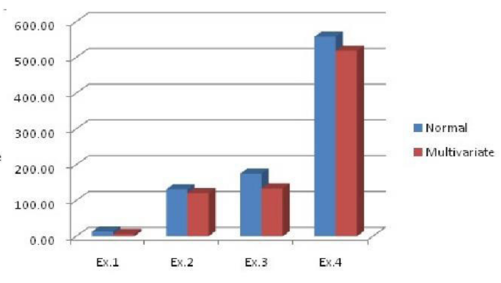
lower level problem, although the iteration number is less than 100.

Tables 7 reveals that the upper level objective function values with their descriptive statistics by EPSDA are better than those by PSO and EDA. Moreover, EPSDA has smaller standard deviation than PSO and EDA separately, so, EPSDA is a more stable method in terms of the solution quality. The best upper level objective function values by EPSDA are as good as or are very close to the best solutions by methods in [28]–[30]. Even the worst upper level objective function values by EPSDA are also very near the best solutions in [28]–[30]. So, EPSDA can find high-quality approximate global solutions for Examples 1-4.

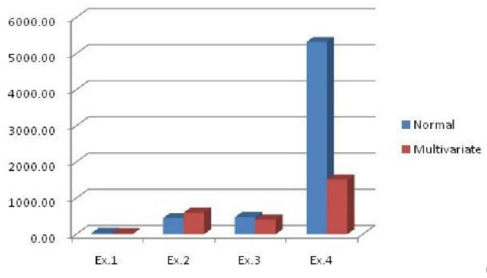
From Tables 10, we conclude that EPSDA can obtain more accurate solutions compared with PSO and EDA. As a result, EPSDA performs better than PSO and EDA separately from the aspects of computation efficiency. Furthermore, the solutions (both the best solutions and worst solutions) by EPSDA are as good as ones by the compared algorithms in the references. And some best solutions are better than existing algorithms. For example, the solution is better than the one by [30] for Example 1 and the solution is better than the one by [28] for Example 2. Hence, it can be concluded that the solution by EPSDA for Examples 1-4 are global solutions or close-to-optimal solutions.



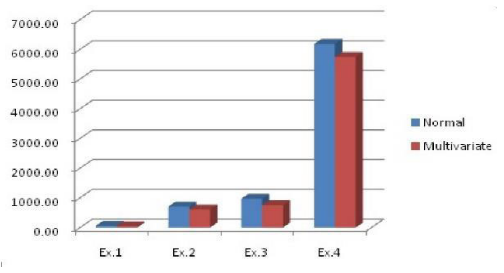
(a) The minimal computational time



(b) The maximal computational time



(c) The mean of computational time



(d) The SD of computational time

FIGURE 9. The descriptive statistics values of the computational time when choosing different probabilistic model.

TABLE 7. The upper level objective function values with their descriptive statistics analysis for the examples.

Ex.	EPSDA				PSO				EDA				[30]	Ref.
	Best	Worst	Mean	SD	Best	Worst	Mean	SD	Best	Worst	Mean	SD		
1	-29.199879	-29.187888	-29.191474	0.0089316	-29.199208	-29.162735	-29.191173	0.0120585	-29.197208	-29.160735	-29.181173	0.0320585	-29.200009	-29.1709 *
2	100.0049	100.1796	100.0822	0.061742	100.103	110.3534	103.232	3.335201	100.0006	100.4571	100.1196	0.147095	100.01	100.58
3	0	0	0	0	4.20e-12	0.006938	0.001625	0.002055	3.97e-09	0.014412902	2.529074e-2	2.663510e-1	0	0
4	2.04e-05	1.14e-05	1.62e-05	2.27e-06	0.229347	3.044411	2.012951	1.052083	0.010226	2.969907	1.879951	1.279341	0	6.21498e-04

* The objective function value in [54] is 29.1709 because of maximizing objective function instead of minimizing it in this paper.

TABLE 8. The iteration numbers with their descriptive statistics analysis for the examples.

Ex.	EPSDA				PSO				EDA			
	Minimal	Maximal	Mean	SD	Minimal	Maximal	Mean	SD	Minimal	Maximal	Mean	SD
1	10	31	20	7	10	24	18	5	9	43	20	11
2	6	19	10	5	6	6	6	6	6	17	9	4
3	7	50	21	13	6	25	10	5	17	50	31	11
4	75	99	85	7	7	51	15	13	7	17	9	4

E. THE NUMERICAL EXPERIMENTS ON THE HIGH-DIMENSIONAL CASES

To further test the performance of the proposed algorithm, Examples 5-8 with high-dimensional decision variable are solved. The parameters are also set as Row "SA5" in Table 2. Descriptive statistics values of the data are computed and listed in TABLE 11 to show the solution quality and computational efficiency of EPSDA.

TABLE 9. The computational time (second) with their descriptive statistics analysis for the examples.

Ex.	EPSDA				PSO				EDA			
	Minimal	Maximal	Mean	SD	Minimal	Maximal	Mean	SD	Minimal	Maximal	Mean	SD
1	5	14	9	3	5	11	8	2	4	12	10	6
2	335	983	529	239	379	399	391	6	137	329	185	62
3	422	2752	1190	732	228	874	373	192	314	872	553	176
4	4885	6406	5466	430	496	3247	1070	939	665	1546	892	332

TABLE 10. The comparison of the solutions by EPSDA and algorithm in References.

Method	Ex.	1		2		3		4	
		x	y	x	y	x	y	x	y
EPSDA	Best	(1.00e-10,0.8984972)	(4.04e-12,0.5989981,0.3959927)	(10.00153)	(9.999259)	(0, 30)	(-10, 10)	(0.999999, 1, 0.999999, 1.000001, 0.999999, 1, 1, 1, 1.000001, 1)	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
	Worst	(1.00e-10,0.8999938)	(3.90e-10,0.599995,0.3999834)	(9.995681)	(9.996681)	(0, 30)	(-10, 10)	(1.000002, 0.999999, 1, 1, 0.999999, 0.999999, 1.000001, 0.999999, 0.999999, 0.999999)	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
PSO	Best	(1.00e-10,0.8990984)	(4.60e-13,0.5993989,0.3975958)	(10.00515)	(9.994854)	(1e-10, 30)	(-10, 10)	(1.013537,1.083308,1.074175,1.003301,1.013273, 1.004521,0.998368,1.005153,1.011284,1.019146)	(6.73e-07,2.58e-06,3.53e-07,2.31e-06,2.09e-06, 2.13e-06,2.05e-06,8.85e-07,1.21e-06,3.12e-06)
	Worst	(1.00e-10,0.8999808)	(3.63e-10,0.5999872,0.3999489)	(10.49333)	(9.506669)	(11.51144, 13.938)	(-8.48856,-6.06201)	(0.999733,0.828209,1.098318,1.380281,0.569551, 1.647481,0.831165,0.384564,1.037593,0.50606)	(2.71e-06,2.35e-06,9.53e-08,3.64e-06,1.34e-06, 3.20e-07,3.64e-06,1.03e-06,3.01e-06,2.55e-06)
EDA	Best	(2.00e-10,0.8990996)	(8.60e-13,0.5996549,0.3985958)	(10.000029)	(9.999971)	(1e-10, 30)	(-10, 10)	(0.999997,0.999029,0.994428,0.999999,1.000921, 1.00221411,1,1,1.000033,1.000494)	(2.47e-06,2.74e-06,2.68e-06,2.72e-06,1.06e-06, 7.48e-07,5.54e-07,1.92e-06,4.25e-07,2.65e-06)
	Worst	(3.00e-10,0.8999988)	(6.63e-10,0.5999896,0.3999679)	(10.022803)	(9.977197)	(1.00e-1030.02883)	(-1010.01441)	(1.092968,1.378811,0.403206,1.473644,0.87687, 1.759502,1.087577,1.144191,1.119742,1.193527)	(2.63e-06,1.16e-06,2.38e-06,6.47e-07,1.93e-06, 9.40e-07,1.09e-06,3.19e-06,2.16e-063,.07e-06)
[30]		(2e-6, 0.899997)	(4e-6, 0.6, 0.400005)	(10.0005)	(9.9995)	(0, 30)	(-10, 10)	(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
Ref.		(0,0.8993)	(0,0.5995,0.3981)	(10.03)	(9.969)	(0, 30)	(-10, 10)	(0.99999998, 0.99999999, 1.00000006,0.99999999, 1.00000000, 1.00000001, 0.99999999,0.99999992, 0.99999998, 1.00000001)	10e-3(-0.04845, -0.03471, 0.11674, 0.09264, 0.2121, 0.09969, 0.07125, 0.05798, 0.04344, 0.03512)

TABLE 11. The upper level objective function values with their descriptive statistics analysis for the examples.

Ex.	The iteration numbers				The computational times (second)				The upper level objective function values				[30]	[29]
	Best	Worst	Mean	SD	Best	Worst	Mean	SD	Best	Worst	Mean	SD		
5	81	93	86	4	3174	4037	3618	297	1.26E-05	2.00E-05	1.54E-05	2.19E-06	0	8.50866e-09
6	192	200	199	3	10149	10835	10550	195	1.29E-05	2.41E-04	1.18E-04	7.17E-05	4.64e-06	2.03246e-05
7	20	100	86	26	282	1375	1185	355	4.92E-04	5.66E-05	2.21E-04	1.63E-04	1.12e-05	6.17254e-08
8	51	66	55	4	2691	3483	2913	236	7.78E-05	9.96E-05	9.05E-05	7.54E-06	0	7.22651e-03

The upper level objective function values are added compare it with one by the proposed EPSDA. Furthermore, the best and worst solutions by our algorithm, and the solutions by Wang *et al.* [29] and Wan *et al.* [30] are listed in TABLE 12.

The results in TABLS XI and XII demonstrate that EPSDA is feasible and efficient to solve the nonlinear bilevel programming even for the high-dimensional cases from the iteration number and computational time. Furthermore, EPSDA can obtain global solutions or close-to-optimal solutions compared with the results in references because the solutions (both the best solutions and worst solutions) by EPSDA are as good as the ones by compared algorithms.

TABLE 12. The computational results for high-dimensional bilevel programming by EPSDA.

	Method	EPSDA proposed in this paper		EDA in [47]		Evolutionary algorithm in [29]	
Ex.		x	y	x	y	x	y
5	Best	(0.99999872,1.00000012,1.00000092,1.00000036,1.00000009,1.00000022,0.99999979,1.00000016, 1.00000023,0.99999981)	(1.37E-06,-9.72E-07,1.24E-06,5.57E-07, 1.18E-06, 7.92E-07,7.38E-07, -1.01E-06, -9.00E-07,2.27E-09)	(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)	(1.00000000, 1.00000000, 1.00000000, 1.00000000,1.00000000,0.99999999, 0.99999999, 1.00000000,0.99999999, 0.99999999	10E-9(-0.89425, 0.85196, 0.85196, -0.89425, -0.89425, -0.89425, -0.67166, 0.85196, 0.85196,0.85196)
	Worst	(0.99999841,1.00000122,1.00000015,1.00000029,0.99999962,1.00000099,1.00000064,1.00000035, 0.99999960,1.00000245)	(4.11E-07,2.22E-06,6.29E-08,9.65E-08, 3.81E-06, 2.33E-06,1.10E-06,6.45E-07, 5.45E-07,3.08E-07)				
6	Best	(-0.08852126,1.72970550,1.24504456,-0.30864768,0.88995355,-2,-0.32984718,-1.22744879,0.98768034,-1.51478841)	(1.12E-06, -1.12E-06,2.94E-06,2.54E-07, 5.85E-07, -5.96E-08, -4.74E-08,3.45E-06, -2.61E-06,-1.85E-06)	(1.149034, 0.08833383, 1.254797, 1.182997,2.130051,1.742112, 0.3082794, 1.591319, 1.409942, -0.2195419)	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)	(0.21650445, 0.70129941, 0.73361882, 0.27478271, -0.39242837,0.18513273, 0.69041802, 0.46409579,0.07205230, 0.77427495)	10E-3(-0.04398, -0.00529, 0.02614, -0.41031, -0.08806, 0.52992,0.29883, -0.39756, -0.57348, 0.58285)
	Worst	(-0.07168044,0.02383437,1.28695439,0.66706651,2.59339114, -0.17585257, 0.91173151,0.89556637,2.33021969,1.59348559)	(-5.75E-07, -2.36E-07,3.48E-06,2.50E-06, 1.59E-06, -2.19E-06,2.37E-06, -2.44E-06, -3.15E-06,-1.73E-06)				
7	Best	(0.79633866,-0.34707730,-0.45585192,0.13129514,1.99801069, 0.27451713,0.416218819,1.047696677, 1.09578361,-1.00364721)	(3.83E-08,3.58E-07,5.10E-07,3.27E-07, 2.01E-07,3.72E-07,1.86E-07,2.89E-07, 6.05E-07,3.13E-07)	(-1.275612, 0.4240169, -1.292204, -0.57017, 1.238698,2.83057, 1.313386, 0.65589, -2.799304, -1.467915)	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)	(1.55838477, -0.64856587, 1.13718010, 0.10569137, -1.76868487,0.58635685, 0.74817824, -0.84311991,0.59501691, 0.49591424)	10E-9(0.90664, -0.75161, -0.33431, -0.54023, 0.90664, -0.54023,0.90663, -0.75161, -0.75161, 0.90664)
	Worst	(0.65596591,0.23490651,-0.57343024,-0.31574694,1.22662592, 0.88769415,0.75942197,1.99944101,-0.16288438,0.09018667)	(9.64E-08,1.81E-07, 4.84E-07,3.81E-07, 4.52E-07,4.84E-08,1.05E-07,3.61E-08, 2.25E-07,2.36E-07)				
8	Best	(1.00001141,1.0000021,0.99999071,0.99999759,1.000009,0.99999881,1.00000190,1.00000436, 1.00001025,1.00000509)	(2.94E-06,1.03E-06,1.76E-06, 3.23E-06, -3.32E-06, -2.19E-06,1.98E-06,2.41E-06, 2.15E-07,-1.59E-06)	(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)	(1.00347786, 1.00046181, 0.99991034, 0.99996156,1.00034019,1.00020616, 0.99985678, 0.9990869,1.00016481, 0.99989127)	10E-3(-0.03757, -0.15561, 0.03938,0.06108, -0.27616, 0.35876, -0.29831,0.02883, 0.00558, -0.02131)
	Worst	(0.999974236,1.00001846,0.999999102,0.999996405,1.00000244,1.000001296,1.00000255,1.00000284,1.00001750,1.00000564)	(1.06E-06, -2.35E-06,2.85E-06, -2.74E-06, 9.70E-07, -1.46E-06, -1.73E-06,2.69E-06, -6.02E-07,2.12E-06)				

V. CONCLUSIONS AND FUTURE WORK

This paper proposes a hybrid PSO with EDA that enables the ability of the particles to learn from the information of the collective experience of the swarm and utilizes the primitive intelligence to search the local area of solutions to solve BLPP. This proposed algorithm combines PSO (the local search method) and EDA (the global search method) together to enhance the efficiency of solving problems. Only the upper level decision variables are used as particles to avoid handing unfeasible particles. In the numerical experiments, 8 examples are selected to demonstrate the performance of EPSDA, where Examples 1-4 are the representative examples for linear, quadratic, nonlinear, high-dimensional nonlinear case and Examples 5-8 are all high-dimensional non-convex and non-differentiable cases. The first four examples are used to carry out sensitive analysis. The experiments show that the linearly decreasing inertia weight and adaptive acceleration coefficients are better than the constant parameters when execute the proposed algorithm. Therefore, these four examples are also used to compare the performance of EPSDA with ones of separate PSO and EDA. The last four examples by the EPSDA demonstrate the its efficiency. The experiments illustrate that EPSDA are better than the separate PSO and EDA. Moreover, it also can obtain global solutions or close-to-optimal solutions compared with the results in the references because the solutions (both the best solutions and worst solutions) by this proposed EPSDA are as good as the ones by the compared algorithms in the references.

In the experiments, the process of dealing with the lower level problem by GA toolbox in Matlab is very time-consuming. Thus, the efficient algorithm for the lower level problem will be one of our future work. The mathematical demonstration of the algorithm convergence with computation complexity, the improvement on the probabilistic model of the EDA operator and the further refinement of the algorithm are also our future work.

APPENDIX

Ex. 1 [29]:

$$\min_{x,y} F(x,y) = -8x_1 - 4x_2 + 4y_1 - 40y_2 - 4y_3,$$
$$s.t. x \geq 0,$$

where y solves:

$$\min_{y} f(x,y) = x_1 + 2x_2 + y_1 + y_2 + 2y_3,$$
$$s.t. - y_1 + y_2 + y_3 \leq 1, 2x_1 - y_1 + 2y_2 - 0.5y_3 \leq 1,$$
$$2x_2 + 2y_1 - y_2 - 0.5y_3 \leq 1, y \geq 0.$$

Ex. 2 [28]:

$$\min_{x,y} F(x,y) = (x-1)^2 + (y-1)^2,$$

where y solves:

$$\min_{y} f(x,y) = 0.5y^2 + 500y - 50xy.$$

Ex. 3 [30]:

$$\min_{x,y} F(x,y) = |2x_1 + 2x_2 - 3y_1 - 3y_2 - 60|,$$
$$s.t. x_1 + x_2 + y_1 - 2y_2 \leq 40, 0 \leq x \leq 50,$$

where y solves:

$$\min_{y} f(x,y) = (y_1 - x_1 + 20)^2 + (y_2 - x_2 + 20)^2,$$
$$s.t. 2y_1 - x_1 + 10 \leq 0, 2y_2 - x_2 + 10 \leq 0,$$
$$-10 \leq y \leq 20.$$

Ex. 4 [30]:

$$\min_{x,y} F(x,y) = \sum_{i=1}^{10} [|x_i - 1| + |y_i|],$$

where y solves:

$$\min_{y} f(x,y) = exp\{[1 + \sum_{i=1}^{10} (y_i^2/4000) - \prod_{i=1}^{10} cos(y_i/\sqrt{i})] \sum_{i=1}^{10} x_i^2\}$$
$$s.t. - \pi \le y \le \pi.$$

Ex. 5 [30]:

$$\min_{x,y} F(x,y) = \sum_{i=1}^{10} [|x_i - 1| + |y_i|],$$

where y solves:

$$\min_{y} f(x,y) = exp\{[100 + \sum_{i=1}^{10} (y_i^2 - 10cos(2\pi y_i))] \sum_{i=1}^{10} x_i^2\},$$
$$s.t. - 3 \le y \le 3.$$

Ex. 6 [30]:

$$\min_{x,y} F(x,y) = |sin(\sum_{i=1}^{10} [|x_i - 1| + |y_i|])|,$$

where y solves:

$$\min_{y} f(x,y) = exp\{[1 + \sum_{i=1}^{10} (y_i^2/4000) - \prod_{i=1}^{10} cos(y_i/\sqrt{i})] \sum_{i=1}^{10} x_i^2\},$$
$$s.t. - \pi \le y \le \pi.$$

Ex. 7 [30]:

$$\min_{x,y} F(x,y) = |sin(\sum_{i=1}^{10} [|x_i - 1| + |y_i|])|,$$

where y solves:

$$\min_{y} f(x,y) = exp\{[100 + \sum_{i=1}^{10} (y_i^2 - 10cos(2\pi y_i))] \sum_{i=1}^{10} x_i^2\},$$
$$s.t. - 3 \le y \le 3.$$

Ex. 8 [30]:

$$\min_{x,y} F(x,y) = \sum_{i=1}^{10} [|x_i - 1| + |y_i|],$$

where y solves:

$$\min_{y} f(x,y) = exp[1 + \sum_{i=1}^{10} ((x_i y_i)^2/4000) - \prod_{i=1}^{10} cos(x_i y_i/\sqrt{i})]\},$$
$$s.t. - \pi \le y \le \pi.$$

REFERENCES

[1] A. Sinha, P. Malo, and K. Deb, “A review on bilevel optimization: From classical to evolutionary approaches and applications,” *IEEE Trans. Evol. Comput.*, vol. 22, no. 2, pp. 276–295, Apr. 2018.

[2] J. Agor and O. Y. Özaltın, “Feature selection for classification models via bilevel optimization,” *Comput. Oper. Res.*, vol. 106, pp. 156–168, Jun. 2019.

[3] D. Aussel and C. S. Lalitha, “Generalized Nash equilibrium problems,” in *Bilevel Programming and MPEC*. Singapore: Springer, 2018.

[4] P.-L. Poirion, S. Toubaline, C. D’Ambrosio, and L. Liberti, “Algorithms and applications for a class of bilevel MILPs,” *Discrete Appl. Math.*, vol. 272, pp. 75–89, Jan. 2020.

[5] X. Deng, “Complexity issues in bilevel linear programming,” in *Multilevel Optimization: Algorithms and Applications*, A. Migdalas, P. M. Pardalos, and P. Varbrand, Eds. Norwell, MA, USA: Kluwer, 1998, pp. 149–164.

[6] B. Colson, P. Marcotte, and G. Savard, “An overview of bilevel optimization,” *Ann. Oper. Res.*, vol. 153, no. 1, pp. 235–256, Jun. 2007.

[7] G. M. Wang, Z. P. Wan, and X. J. Wan, “Bibliography on bilevel programming,” (in Chinese), *Adv. Math.*, vol. 36, no. 5, pp. 513–529, 2007.

[8] Y. Jiang, X. Li, C. Huang, and X. Wu, “Application of particle swarm optimization based on CHKS smoothing function for solving nonlinear bilevel programming problem,” *Appl. Math. Comput.*, vol. 219, no. 9, pp. 4332–4339, Jan. 2013.

[9] T. Zhang and X. Li, “The backpropagation artificial neural network based on elite particle swam optimization algorithm for stochastic linear bilevel programming problem,” *Math. Problems Eng.*, vol. 2018, pp. 1–9, Oct. 2018.

[10] F. Vandenbergh and A. Engelbrecht, “A study of particle swarm optimization particle trajectories,” *Inf. Sci.*, vol. 176, no. 8, pp. 937–971, Apr. 2006.

[11] H. Mühlenbein and G. Paaß, “From recombination of genes to the estimation of distributions I. Binary parameters,” in *Proc. 4th Int. Conf. Parallel Problem Solving Nature*, Berlin, Germany, Sep. 1996, pp. 178–187.

[12] M. Iqbal and M. Montes de Oca, “An estimation of distribution particle swarm optimization algorithm,” in *Proc. ANTS*, in Lecture Notes in Computer Science, vol. 4150, 2006, pp. 72–83.

[13] R. V. Kulkarni and G. K. Venayagamoorthy, “An estimation of distribution improved particle swarm optimization algorithm,” in *Proc. 3rd Int. Conf. Intell. Sensors, Sensor Netw. Inf.*, Melbourne, VIC, Australia, 2007, pp. 539–544.

[14] J. Wang, “A novel discrete particle swarm optimization based on estimation of distribution,” in *Proc. ICIC LNAI*, vol. 4682, 2007, pp. 791–802.

[15] H. Liu, L. Gao, and Q. Pan, “A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem,” *Expert Syst. Appl.*, vol. 38, no. 4, pp. 4348–4360, Apr. 2011.

[16] C. W. Ahn, J. An, and J.-C. Yoo, “Estimation of particle swarm distribution algorithms: Combining the benefits of PSO and EDAs,” *Inf. Sci.*, vol. 192, no. 1, pp. 109–119, Jun. 2012.

[17] J. F. Bard, “Some properties of the bilevel linear programming,” *J. Optim. Appl.*, vol. 68, no. 2, pp. 146–164, 1991.

[18] S. Dempe, *Foundation of Bilevel Programming*. Norwell, MA, USA: Kluwer, 2002.

[19] R. C. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in *Proc. 6th Int. Symp. Micro Mach. Human Sci.*, vol. 1, 1995, pp. 39–43

[20] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proc. IEEE Int. Conf. Neural Netw.*, Nov. 1995, pp. 1942–1948.

[21] K. E. Parsopoulos and M. N. Vrahatis, “Recent approaches to global optimization problems through particle swarm optimization,” *Natural Comput.*, vol. 1, nos. 2–3, pp. 235–306, 2002.

[22] A. Alfi, “PSO with adaptive mutation and inertia weight and its application in parameter estimation of dynamic systems,” *Acta Automatica Sinica*, vol. 37, no. 5, pp. 541–549, May 2011.

[23] H. Modares, A. Alfi, and M.-M. Fateh, "Parameter identification of chaotic dynamic systems through an improved particle swarm optimization," *Expert Syst. Appl.*, vol. 37, no. 5, pp. 3714–3720, May 2010.

[24] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: An overview," *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, 2007.

[25] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Norwell, MA, USA: Kluwer, Univ. of the Basque Country, 2001.

[26] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm Evol. Comput.*, vol. 1, no. 3, pp. 111–128, Sep. 2011.

[27] J. A. Lozano, P. Larrañaga, I. Iñza, and E. Bengoetxea, *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*. Berlin, Germany: Springer, 2006.

[28] V. Oduguwa and R. Roy, "Bi-level optimisation using genetic algorithm," in *Proc. IEEE Int. Conf. Artif. Intell. Syst. (ICAIS)*, 2002, pp. 322–327.

[29] Y. Wang, Y.-C. Jiao, and H. Li, "An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme," *IEEE Trans. Syst., Man Cybern. C, Appl. Rev.*, vol. 35, no. 2, pp. 221–232, May 2005.

[30] Z. Wan, L. Mao, and G. Wang, "Estimation of distribution algorithm for a class of nonlinear bilevel programming problems," *Inf. Sci.*, vol. 256, pp. 184–196, Jan. 2014.

[31] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proc. Congr. Evol. Comput. (CEC)*, 1999, pp. 1945–1950.

[32] R. Eberhart and Y. Shi, "Particle swarm optimization: Developments, applications and resources," in *Proc. Congr. Evol. Comput.*, Seoul, South Korea, 2001, pp. 81–86.

[33] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Int. Conf. Evol. Comput., IEEE World Congr. Comput. Intell.*, May 1998, pp. 69–73.

[34] A. R. Jordehi and J. Jasni, "Parameter selection in particle swarm optimisation: A survey," *J. Experim. Theor. Artif. Intell.*, vol. 25, no. 4, pp. 527–542, Dec. 2013.

[35] W. Han, P. Yang, H. Ren, and J. Sun, "Comparison study of several kinds of inertia weights for PSO," in *Proc. IEEE Int. Conf. Prog. Informat. Comput.* Washington, DC, USA: IEEE Computer Society, Dec. 2010, pp. 280–284.

[36] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 240–255, Jun. 2004.

[37] Z.-H. Zhan, J. Xiao, J. Zhang, and W.-n. Chen, "Adaptive control of acceleration coefficients for particle swarm optimization based on clustering analysis," in *Proc. IEEE Congr. Evol. Comput.* Singapore, Sep. 2007, pp. 3276–3282.

[38] G. Q. Bao and K. F. Mao, "Particle swarm optimization algorithm with asymmetric time varying acceleration coefficients," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*. Washington, DC, USA: IEEE Computer Society, Dec. 2009, pp. 2134–2139.

GUANGMIN WANG (Member, IEEE) was born in Henan, China, in 1978. He received the B.S. and M.S. degrees in applied mathematics and the Ph.D. degree in system engineering from Wuhan University, in 2001, 2004, and 2007, respectively.

Since 2014, he has been a Professor with the China University of Geosciences, Wuhan. His research interests include theory and methods of optimization and system engineering.

LINMAO MA was born in Hubei, in 1988. He received the B.A. degree in management from the China University of Geoscience, Wuhan, Hubei, the B.E. degree in engineering from the Huazhong University of Science and Technology, in 2011, and the M.A. and Ph.D. degrees in management from the China University of Geoscience, in 2014 and 2019, respectively.

From 2017 to 2018, he worked as a Visiting Ph.D. with the University of Southern Denmark, Esbjerg, Denmark, with the funding from the China Scholarship Council. Since 2020, he has been a Lecturer with the School of Management, South-Central University for Nationalities. His research interests include evaluation of sustainability development, multicriteria decision-making, bilevel programming problems, and intelligent algorithms.