Swarm and Evolutionary Computation 60 (2021) 100808

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

A particle swarm optimization algorithm for mixed-variable optimization problems

Feng Wang[a,*], Heng Zhang[a], Aimin Zhou[b]

[a] School of Computer Science, Wuhan University, Wuhan 430072, China
[b] Shanghai Key Laboratory of Multidimensional Information Processing, School of Computer Science and Technology, East China Normal University, Shanghai 200241, China

ARTICLE INFO

Keywords:
Particle swarm optimization
Mixed-variable optimization
Parameter tuning

ABSTRACT

Many optimization problems in reality involve both continuous and discrete decision variables, and these problems are called mixed-variable optimization problems (MVOPs). The mixed decision variables of MVOPs increase the complexity of search space and make them difficult to be solved. The Particle Swarm Optimization (PSO) algorithm is easy to implement due to its simple framework and high speed of convergence, and has been successfully applied to many difficult optimization problems. Many existing PSO variants have been proposed to solve continuous or discrete optimization problems, which make it feasible and promising for solving MVOPs. In this paper, a new PSO algorithm for solving MVOPs is proposed, namely PSO_{mv}, which can deal with both continuous and discrete decision variables simultaneously. To efficiently handle mixed variables, the PSO_{mv} employs a mixed-variable encoding scheme. Based on the mixed-variable encoding scheme, two reproduction methods respectively for continuous variables and discrete variables are proposed. Furthermore, an adaptive parameter tuning strategy is employed and a constraints handling method is utilized to improve the overall efficiency of the PSO_{mv}.The experimental results on 28 artificial MVOPs and two practical MVOPs demonstrate that the proposed PSO_{mv} is a competitive algorithm for MVOPs.

1. Introduction

Optimization problems can be divided into continuous optimization problems (CnOPs) and discrete optimization problems (DOPs) according to the category of the decision variables. In the fast few years, lots of algorithms have been proposed to address CnOPs or DOPs. However, many real-world optimization problems not only involve continuous but discrete variables, e.g., industrial design problems and financial optimization problems, which are called Mixed-Variable Optimization Problems (MVOPs). The mixed variables definitely increase the complexity of the search space and improve the difficulty of solving these optimization problems. Therefore, it is necessary to design more efficient algorithms to solve MVOPs.

Evolutionary algorithms (EAs) are a new kind of optimization algorithms that do not rely on the mathematical characteristic of the problems, e.g., Genetic Algorithm (GA) [12], Particle Swarm Optimization (PSO) [34], Differential Evolution (DE) [36] and Ant Colony Optimization (ACO) [8]. Due to the effectiveness of EAs, many EAs have been widely utilized to handle kinds of optimization problems [1,2,7,23,37–39,39,43,44]. However, most EAs are first proposed to address CnOPs or DOPs. For example, classical DE and classical PSO are proposed for handling CnOPs, while classical ACO is only suitable for DOPs. Therefore, the design of EAs that can solve MVOPs has attracted the attention of scholars. In the literature, some EAs have been proposed for MVOPs. These EAs can be divided into three categories based on the way they handle mixed variables.

The first category is characterized by relaxation methods [11,13,18,24,28]. These algorithms relax discrete variables into continuous variables. Thus reproductions methods that are designed for CnOPs can be easily deployed to solve MVOPs. However, algorithms based on relaxation methods are not always effective. On the one hand, the performance of these algorithms mainly depend on relaxation methods which are not applicable to all problems. On the other hand, discrete variables can be subdivided into ordinal variables and categorical variables. Ordinal variables can be readily relaxed into continuous, while relaxing categorical variables into continuous variables may decline the performance of algorithms as the number of categorical variables increases [21]. Therefore, the first category of EAs is limited to MVOPs involving discrete categorical variables and continuous variables.

Different from the first category, the second category of EAs treats continuous variables by discretization method. Thus the reproduction

* Corresponding author.
E-mail address: fengwang@whu.edu.cn (F. Wang).

https://doi.org/10.1016/j.swevo.2020.100808
Received 22 February 2020; Received in revised form 8 October 2020; Accepted 12 November 2020
Available online 25 November 2020

methods that are designed for DOPs can be utilized to handle continuous variable in the same way as discrete variables. For example, based on estimation distribution algorithm, Zhou and Zhang [46] adopts a variable-width histogram to handle continuous variables. The continuous search space is partitioned into discrete bins, in which continuous variable are sampled. Another binary-code GA [26] encodes continuous variables into binary string. Thus the traditional binary crossover and mutation operators can operate on continuous variables. However, due to the limitation of the length of the binary string, the performance of binary-code GA deteriorates when solving more precise problems.

In the third category, different operators are employed to handle continuous variables and discrete variables separately. These EAs utilize continuous reproduction methods and discrete reproduction methods simultaneously to generate offsprings with mixed variables [21,33,40,45]. There are two advantages for these kinds of algorithms. Firstly, no additional operators are required to do variable conversion, which would consume extra computing resources. Secondly, these algorithms can easily be combined with efficient operators, regardless of whether the operators are from the same algorithm or not. As a result of this, the reproduction methods for continuous variables and discrete variables may not be consistent and compatible.

The third category of EAs is more simple and has been widely applied to solve MVOPs. However, the mixed variables increase the complexity of search space and improve the difficulty of solving MVOPs. PSO is a widely used optimization framework inspired by swarm intelligence [10]. Each solution in PSO is represented as a particle, which evolves in a cooperative manner. The two vectors, namely velocity vector and position vector, represent the particle's evolutionary state in the search space. Particles evolve to the best position by changing these two vectors iteratively. Because of the simplicity and effectiveness of PSO, more and more PSO variants have been proposed to address challenging optimization problems. The classical PSO is designed for CnOPs and other more efficient variants for CnOPs have been proposed in the last few decades [19,20,41]. For some DOPs, other excellent discrete version PSO variants are designed [16,27,30,31,42]. The outstanding performance of the existing PSO algorithms has validated the efficiency of PSO, and make it feasible and promising for solving MVOPs.

In this paper, we propose a mixed-variable version PSO, namely PSO_{mv}. In order to handle mixed variables with different operators, a mixed-variable encoding scheme is utilized. Based on the mixed-variable encoding scheme, two reproduction methods for continuous variables and discrete variables are proposed, i.e., the continuous reproduction method and the discrete reproduction method. The continuous reproduction method supports the swarm search in promising area, and the discrete reproduction method adequately use the search information to generate excellent offsprings. Furthermore, an adaptive parameter tuning strategy and a constraints handling method are adopted to further promote the overall efficiency of PSO_{mv} when dealing with more complex MVOPs.

The rest of this paper is organized as follows. A brief introduction of PSO and other EAs for MVOPs are presented in Section 2. The details of our proposed PSO_{mv} algorithm is described in Section 3. The experimental results and analyses on several benchmark functions are shown in Section 4, and the conclusion is provided in Section 5.

2. Related work

2.1. PSO algorithms

2.1.1. The classical PSO

PSO is a swarm intelligence algorithm which was proposed based on the observation on the swarm behaviors in some ecological system, the classical PSO algorithm was first proposed by Kennedy in 1995 [10]. Solutions in PSO are represented as particles, these particles hold two vectors, i.e., position vector and velocity vector, representing the particle's evolutionary state in the search space. For a problem with D-dimensional space, particle i holds the position vector $X_i = (x_i^1, x_i^2, ..., x_i^D)$ and velocity vector $V_i = (v_i^1, v_i^2, ..., v_i^D)$. During the iteration of algorithm, these two vectors are updated by the following rules.

$$V_i(t+1) = w \cdot V_i(t) + c_1 r_1 (pbest_i(t) - X_i(t)) + c_2 r_2 (gbest(t) - X_i(t)) \tag{1}$$

$$X_i(t+1) = X_i(t) + V_i(t) \tag{2}$$

where $pbest_i$ is the personal best position searched by the ith particle, while $gbest$ is the best-so-far position search by the whole swarm. c_1 and c_2 are two acceleration coefficients to maintain the trade-off between personal experience and social experience. r_1 and r_2 are two random numbers uniformly distributed in the range of [0,1]. w is the inertia weight, and the experimental analysis in [34] showed that a higher w is good for exploration while a smaller w is beneficial to exploitation.

2.1.2. Variants of PSO

To solve CnOPs, some continuous version variants of PSO (CPSO) have been proposed. These CPSO algorithms can be divided into three categories. The first category of CPSO focuses on the modification of parameters. In [35], the inertia weight w is adaptively adjusted in a nonlinearly way by using fuzzy inference. In [29], the acceleration coefficients c_1 and c_2 are tuned adaptively. The second category of CPSO utilizes hybrid mechanisms. The ALPSO proposed in [41] uses hybrid learning strategies to adaptively adjust search direction. SL-PSO [5] introduces social learning mechanisms, particles in SL-PSO can serve as a demonstrator for different imitators. Some variants of CPSO hybridize PSO with other search operators, including PSO with genetic algorithms [15], PSO with ant colony optimization [32], PSO with differential evolution [15], and all that. The third category of CPSO focuses on increasing the diversity of swarm. Some variants introduce topological structures into the swarm, e.g., [17]. Another kind variants of CPSO, like CLPSO [19] and FIPS [25], maintain diversity by modifying velocity vector updating rules. Due to these rules, particles not only learning their personal best position but other particles' search experience. Besides, DMS-PSO [20] guarantees diversity by maintaining multiple subswarms. Subswarms in DMS-PSO are dynamically established and changed.

To solve DOPs, some discrete version variants of PSO (DPSO) have been proposed. The first discrete version PSO was proposed by Kennedy in 1997 [16], namely BPSO. BPSO operates on discrete binary variables, where the trajectories of particles are changed in the probability that a position will take on 0 or 1. However, BPSO is only applicable to discrete binary optimization problems. Some other general variants of DPSO are established subsequently [27,30,31,42]. These DPSO variants utilize swap-operator scheme, space transformation technique or fuzzy-matrix based scheme to handle discrete variables. The set-based PSO (S-PSO) [4] is proposed by introducing a set-based theory, in which the original position and velocity updating rules are redefined on crisp sets.

2.2. Evolutionary algorithms for MVOPs

Some variants of EAs have been used to solve MVOPs, such as ACO_{mv}[21], DE_{mv} [45], $AEDA$ [33] and EDA_{mvn} [40]. To handle mixed variables, ACO_{mv} adopts three operators, which are specially designed for continuous variables, ordinal variables, and categorical variables. In detail, a continuous optimization mechanism ACO_R, a continuous relaxation mechanism ACO_{mv-O} for ordinal variables, and a categorical optimization mechanism ACO_{mv-c} for categorical variables. Besides that, a DE variant for MVOPs is proposed in [45]. The DE_{mv} hybridizes the original DE [36] for CnOPs and the set-based DE [22] for DOPs. Experimental results and comparisons with other representative algorithms show that DE_{mv} is a competitive algorithm for MVOPs. Furthermore, an adaptive EDA algorithm AEDA is proposed for multipolicy insurance investment planning [33], which involves continuous variables (investment amount) and discrete variables (hospitalization policies).

In AEDA, continuous variables are adaptively sampled by the Gaussian distribution model and the Cauchy distribution model. As for discrete variables, a histogram model is established and samples discrete values. These variants of algorithms are proposed based on their classical algorithms by taking advantages of the classical algorithms.

3. PSO algorithm for MVOPs

Many PSO variants for CnOPs or DOPs have been proposed to solve kinds of optimization problems. However, PSO variants for MVOPs have rarely been discussed in literature. For the purpose of solving MVOPs and taking advantage of the PSO framework, we propose a PSO variant for MVOPs, named as PSO_{mv}.

3.1. Mixed-variable encoding scheme

As mentioned above, there are mainly three categories of algorithms to handle mixed variables. The first category of algorithms utilize relaxation method, e.g., round-off method, to transfer continuous variables into discrete variables. Therefore, evolutionary operators for CnOPs can be directly adopted to handle both continuous variables and discrete variables. However, the performance of the relaxation method will decrease when the number of discrete categorical variables increases [21]. Different from the first category, the second category of algorithms uses the discretization method. For example, continuous variables in [16,26] are encoded with binary strings. However, the length of binary string restricts the precision of algorithm. As for the third category of algorithms, different operators for CnOPs and DOPs are hybridized to separately handle continuous variable and discrete variables. There are mainly two advantages to the third category. On the one hand, the third category of algorithms doesn't require additional variable conversion technique, which consumes extra computation resources. On the other hand, many excellent evolutionary operators for CnOPs or DOPs can be simply hybridized to solve MVOPs.

As a result of this, it is reasonable for PSO_{mv} to hybridize different operators to handle MVOPs. Here, to fit different operators for CnOPs and DOPs, PSO_{mv} adopts a mixed-variable encoding scheme. The position vectors of particles with mixed-variable encoding scheme can be illustrated in Fig. 1.

As shown in Fig. 1, the position vector of the particle is divided into two segments. The first Z dimensions of variables are encoded with continuous variables and the rest V variables are encoded with discrete variables. Then the position vector of the particle i can be represented as $X_i = (x_i^1, x_i^2, ..., x_i^Z, x_i^{Z+1}, x_i^{Z+2}, ..., x_i^{Z+V})$. Thus the particles that encoded with mixed variables can represent the solutions to MVOPs. Besides that, benefitting from this mixed-variables encoding scheme, PSO_{mv} does not need additional variables transformation method but can utilize continuous evolutionary operators and discrete evolutionary operators to handle continuous variables and discrete variables separately.

3.2. Reproduction methods

Base on the mixed variables encoding scheme, PSO_{mv} adopts hybrid reproduction methods to handle mixed variables. The continuous reproduction method and the discrete reproduction method are utilized simultaneously to generate the complete position vectors of offsprings. The process of how the two reproduction methods work is shown in Fig. 2.

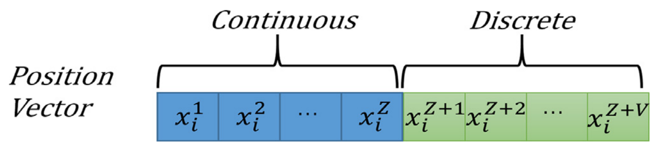


Fig. 1. Mixed-Variable Encoding Scheme.

During the iteration of the PSO_{mv}, two reproduction methods separately generate a part of position vector for offspring based on parent particle, i.e., continuous variables and discrete variables. Then, the two segments of position vector will be combined to construct a complete offspring particle. Since two reproduction method work independently and simultaneously, PSO_{mv} can take any efficiency reproduction methods. Although the independent reproduction methods ignore the correlation between different kind variables, this approach ensures the scalability and flexibility of the PSO_{mv}. In this paper, we present a continuous reproduction method and a discrete reproduction method for PSO_{mv} respectively.

3.2.1. Continuous reproduction method

In the classical PSO algorithm, particles not only learn from their own personal best coordinates but the global best particle. Thus the swarm can easily get trapped into local optima as long as the global best particle cannot be improved. In order to avoid the premature convergence, various variants of PSO that aim at promoting the diversity of population are proposed [19,20,25]. These algorithms promote diversity by maintaining multiple subswarms, modifying position updating rules or mixing other evolutionary operators. However, increasing diversity cannot guarantee good convergence. In order to promote diversity of swarm while maintain good convergence efficiently, we propose a continuous reproduction method (CR) for PSO_{mv}. To guarantee the diversity of swarm, particles in PSO_{mv} randomly learn the personal best coordinates of the whole particles, including its own. The modified velocity updating rule is shown below.

$$V_i(t+1) = w \cdot V_i(t) + c \cdot rand(pbest_r(t) - X_i(t)) \tag{3}$$

where c is set to 2.0 and $rand$ is a random number distributed in [0,1]. $pbest_r(t)$ is a randomly selected personal best coordinate in the t^{th} generation. V_i is the velocity vector of the ith particle. To accurately represent the motion state of particles in continuous space, V_i is encoded with continuous variables.

However, the randomly selected personal best $pbest_r(t)$ may be worse than the ith particle. Thus the randomly selected learning objects for particles may promote the diversity of swarm but deteriorate the convergence of swarm. To ensure the trade-off between diversity and convergence in PSO_{mv}, the ith particle only learn from the personal best whose fitness value is superior to $pbest_i$'s fitness value. Therefore, before each iteration, particles are firstly sorted from the worst to the best in terms of their $pbest$'s fitness.

For the ith particle, a learning object $pbest_r$ is generated, $i \le r \le N$. Then the particle' velocity vector is updated by learning from $pbest_r$, and position vector is updated by the same way in classical PSO. The overall process of the continuous reproduction method is shown in Algorithm 1.

Algorithm 1 Continuous Reproduction Method

```
1: Input: the sorted swarm, the particle $X_i$ and the parameter $w_i$ for $X_i$;
2: Output: $C = (x_i^1, x_i^2, ..., x_i^Z)$.
3: for each continuous variable $x_i^j$, $j = 1$ to $Z$ do
4:     Randomly generate index number $r$, $i \le r \le N$;
5:     Update the $v_i^j$ by learning the $r^{th}$ personal best $pbest_r$ in Eq. 3;
6:     Update $x_i^j$ in the same way as classical PSO in Eq. 2;
7: end for
8: Return the continuous variables $C = (x_i^1, x_i^2, ..., x_i^Z)$.
```

As shown in Algorithm 1, particles are first sorted in terms of the fitness values of their personal best. Then, particles will randomly select their learning object. Different particles may learn from different personal best, which help the swarm maintain good diversity. To ensure the convergence meanwhile, each particle only learns from the personal best whose fitness value is superior to the fitness value of its own personal best. If the ith particle is ranked in the front of the whole swarm, which

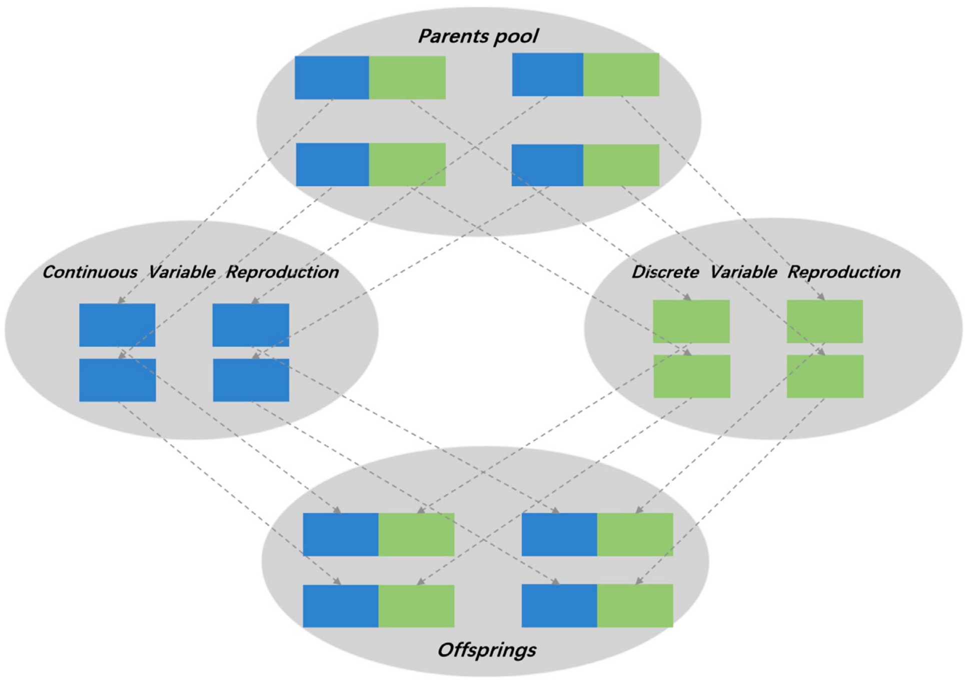

Parents pool
Continuous Variable Reproduction
Discrete Variable Reproduction
Offsprings

Fig. 2. Process of Two Reproduction Methods.

means particle i holds the worst personal best in the current swarm, the learning object of particle i can be any other personal best. On the contrary, the particle ranked at the end of the swarm only learns its own personal best. This continuous reproduction method ensures that particles learn from better objects and search in the direction of the promising space. Thus, the convergence of swarm can be guaranteed. Besides, different particles may learn from different objects, so the swarm can maintain good diversity.

3.2.2. Discrete reproduction method

PSO is a swarm intelligence algorithm, in which all particles evolve in a cooperative manner. The search information obtained by one particle may be useful to other particles. In order to take full advantages of the search information obtained by the whole swarm, the discrete reproduction method (DR) of PSO_{mv} is proposed. For the jth discrete variable, values of the variable are assigned according to probabilities rather than the velocity vectors. The available values with higher probabilities are more likely to be assigned to this variable. The probabilities for available values to be assigned to the jth variable is initialized at the beginning and then be updated by collecting the search information obtained by the whole swarm. The probability for an available value is initialized as follows.

$$Prob_{j,n}(0) = \frac{1}{n_j} \tag{4}$$

where n_j is the number of the available values for the jth variable, $Prob_{j,n}(0)$ denotes the probability of the n^{th} available value to be assigned to the jth variable. At the beginning of the iteration, the probabilities for all available values are equal.

During the iteration of the algorithm, the superior particles maintain better position vectors, which are worth being learned by other particles. Therefore, this reproduction method collects the search information obtained by half of the swarm's superior particles and updates the probabilities for all available values. The process of updating probabilities and generating offsprings is shown in Algorithm 2.

As shown in Algorithm 2, from line 3 to line 13, the probabilities for available values are updated by considering the historical search information and the current search information. The equation in line 11 illustrates the updating rule, where the first term denotes the historical search information and the second term denotes the current search information. The historical search information can be represented by the previous probabilities of the available values to be assigned. For the nth available value of the jth variable in the tth generation, the probability of this available value is represented by $Prob_{j,n}(t)$. The current search information is represented by $\frac{Count_{j,n}}{N/2}$, where $Count_{j,n}$ denotes the number of particles that are assigned with the nth value in the jth variable. After the probabilities being updated, offsprings can be generated. As shown in the line from 14 to 16, each discrete variable of offsprings is assigned with a value according to the updated probabilities.

In PSO_{mv}, the continuous reproduction method and the discrete reproduction method are simultaneously utilized to handle continuous variables and discrete variables. The continuous reproduction method randomly selects excellent learning objects for particles, thus swarm can converge quickly while maintaining good diversity. The discrete reproduction method method holds the population search information and

Algorithm 2 Discrete Reproduction Method

```
1: **Input:** the sorted swarm, the particle $X_i$ and the parameter $\alpha_i$ for $X_i$;
2: **Output:** $D = (x_i^{Z+1}, x_i^{Z+2}, ..., x_i^{Z+V})$.
3: for each discrete variable $x_i^{Z+j}$, $j = 1$ to $V$ do
4:     for each available value $n$, $n = 1$ to $n_j$ do
5:         $Count_{j,n} = 0$
6:         for each personal best $pbest_i$, $i = \frac{N}{2}$ to $N$ do
7:             if $pbest_{i,j} == Values_{j,n}$ then
8:                 $Count_{j,n} = Count_{j,n} + 1$
9:             end if
10:        end for
11:        $Prob_{j,n}(t+1) = \alpha_i \times Prob_{j,n}(t) + (1-\alpha_i) \times \frac{Count_{j,n}}{N/2}$
12:    end for
13: end for
14: for each discrete variable $x_i^{Z+j}$, $j = 1$ to $V$ do
15:     Assign an available value to $x_i^{Z+j}$ according probability $Prob_j$;
16: end for
17: Return the discrete variables $D = (x_i^{Z+1}, x_i^{Z+2}, ..., x_i^{Z+V})$.
```

generates offsprings by considering the search information including the historical information and the current information. The comprehensive information makes the discrete reproduction method generates more promising offsprings. Since both continuous reproduction method and discrete reproduction method aim at taking advantage of search information obtained by particles and the whole swarm, more promising particles with mixed variables can be generated.

3.3. Adaptive parameter tuning strategy

The parameters of the reproduction method, e.g., inertia weight w, are significant to the performances of PSO_{mv}. Intuitively, it is inappropriate to set parameters with fixed values for all particles. To make parameters of PSO_{mv} suitable for various problems, an adaptive parameter tuning strategy (APT) is utilized in PSO_{mv}. During each iteration in PSO_{mv}, based on the mixed-variable encoding scheme mentioned above, the tuning parameters w and α would be generated for a particle X before reproduction, in which w is used for continuous variable reproduction and α is used to balance historical search information and current search information in discrete variable reproduction.

Let $\bar{w}$ be the average inertia weight and $\bar{\alpha}$ be the average value of α found from historical best particles. The basic idea of the APT strategy is taking full use of the historical information of the particles to generate good parameters to improve convergence. The parameter tuning process is shown in Algorithm 3.

Algorithm 3 Parameter Tuning Strategy

```
1: Input: the average value of historical best parameters of particles $\bar{w}$
   and $\bar{\alpha}$, the parameter set $Set_P$;
2: Output: the tuning parameters $w$ and $\alpha$.
3: Generate a random number $rand$;
4: if $rand \leq 0.5$ then
5:     $w = Cauchy(\bar{w}, 0.1)$
6:     $\alpha = Cauchy(\bar{\alpha}, 0.1)$
7: else
8:     $w = Gaussian(\bar{w}, 0.1)$
9:     $\alpha = Gaussian(\bar{\alpha}, 0.1)$
10: end if
11: Return $w$ and $\alpha$.
```

At the beginning, both $\bar{w}$ and $\bar{\alpha}$ are initialized as 0.5. As shown in the line from 4 to 10 in Algorithm 3, for each particle, the tuning parameters w_i and α_i will be produced by using Cauchy distribution or Gaussian distribution. Since the Gaussian distribution is a bell-shaped curve with thin tails on both sides thus most values will be close to the mean value, while the Cauchy distribution has relatively fat tails, the Cauchy distribution can generate values with more diversity. To balance the convergence and diversity, the probability of choosing the Gaussian distribution is set to 0.5.

We use the parameter set Set_P to record the parameter values that can promote particles, and S records the number of each parameter in Set_P. If the generated offspring X_i^{new} is better than X_i in terms of the fitness, the parameters $\bar{w}$ and $\bar{\alpha}$ will be recorded in the set Set_P, and $S = S + 1$. Then the parameters can be updated as follows.

$$\begin{aligned} \bar{w} &= (\bar{w} * S + w)/(S+1) \\ \bar{\alpha} &= (\bar{\alpha} * S + \alpha)/(S+1) \end{aligned} \tag{5}$$

And the parameter set Set_P can be updated by,

$$Set_P = Set_P \bigcup \{(w, \alpha)\} \tag{6}$$

Here, the parameter set is initialized as an empty set and $S = 0$ at the beginning of the iteration, and would be updated during the iteration of PSO_{mv}.

3.4. Constraints handling method

Real-world MVOPs always involve multiple constraints. Besides the mixed variables, various equality constraints and inequality constraints also increase the complexity of search space. In order to further promote the practical performance of PSO_{mv}, a simple constraint handling method is utilized. There are mainly two kinds of methods to handle constraints. The first one is death penalty method, in which the infeasible solutions are arbitrarily assigned with a large fitness value, and it make infeasible solutions lose superiority during the selection process. However, the death penalty method does not consider the differences between infeasible solutions. The second method is the penalty function method. The fitness value assigned to infeasible solutions not only involves the original fitness function but the constraint violation (CV), which is presented in Eq. (7).

$$CV(X) = \sum_{j=1}^{P} |min(g_j(X), 0)| + \sum_{k=1}^{L} |h_k(X)| \tag{7}$$

where $g_j(X)$ is the j_{th} inequality constraint and $h_k(X)$ is the k_{th} equality constraint. Thus the fitness value assigned to infeasible solutions can be represented as the following Eq. (8).

$$f^*(X) = f(X) + Q(CV(X), q) \tag{8}$$

where $f(X)$ is the original fitness function and $Q()$ is the CV control function influenced by control parameter q.

The performance of the second method is extraordinarily sensitive to the control functions and parameters. Hence, for simplicity, a linear control function with non-parameter is adopted in PSO_{mv}. For infeasible solutions, the fitness value is calculated by the following rule.

$$f^*(X) = f(X) + CV(X) \tag{9}$$

During the selection process, feasible solutions are selected in terms of their original fitness values while infeasible solutions are distinguished by their penalty function values.

3.5. Framework of PSO_{mv} algorithm

The framework of PSO_{mv} algorithm can be summarized in Algorithm 4.

As shown in Algorithm 4, from line 9 to line 12, offspring X_i^{new} is generated by two reproduction methods. Since PSO_{mv} employs a mixed-variable encoding scheme, and the search space and evolution operators of continuous variables and discrete variables are totally different, it is reasonable to generate new variables by different reproduction methods separately at first, and combine them to get a new population.

4. Experimental studies

To verify the efficiency of PSO_{mv}, we choose 28 artificial benchmarks and two real-world MVOPs to test PSO_{mv}. These 28 artificial benchmarks are developed based on the CEC2013 benchmark functions. The other real-world MVOPs come from practical industrial design problems. Four state-of-the-art EAs for MVOPs are chosen to compare with PSO_{mv}.

4.1. Artificial benchmarks generation

The artificial benchmarks are developed based on the 28 CEC2013 benchmark functions. The decision variables of these benchmarks involve continuous variables and discrete variables. For the Z-dimensional continuous variables, the new benchmarks maintain the original characteristic of CEC2013 benchmarks, including continuous search range $[-100, 100]^Z$, continuous shift global optimum $O = (o_1, o_2, o_3, ..., o_Z)$ (randomly distributed in $[-80, 80]^Z$), and rotation matrix. For the V-dimensional discrete variables, the continuous shift

Table 1
The generated 28 benchmark functions.

	No.	Functions	Optimum
Unimodal functions	1	Sphere Function	–1400
	2	Rotated High Conditioned Elliptic Function	–1300
	3	Rotated Bent Cigar Function	–1200
	4	Rotated Discus Function	–1100
	5	Different Powers Function	–1000
Multimodal functions	6	Rotated Rosenbrocks Function	–900
	7	Rotated Schaffers F7 Function	–800
	8	Rotated Ackleys Function	–700
	9	Rotated Weierstrass Function	–600
	10	Rotated Griewanks Function	–500
	11	Rastrigins Function	–400
	12	Rotated Rastrigins Function	–300
	13	Non-Continuous Rotated Rastrigins Function	–200
	14	Schwefel's Function	–100
	15	Rotated Schwefel's Function	100
	16	Rotated Katsuura Function	200
	17	Lunacek-Bi-Rastrigin Function	300
	18	Rotated Lunacek Bi-Rastrigin Function	400
	19	Expanded Griewanks plus Rosenbrocks Function	500
	20	Expanded Scaffers F6 Function	600
Composition functions	21	Composition Function 1 (n=5,Rotated)	700
	22	Composition Function 2 (n=3,Unrotated)	800
	23	Composition Function 3 (n=3,Rotated)	900
	24	Composition Function 4 (n=3,Rotated)	1000
	25	Composition Function 5 (n=3,Rotated)	1100
	26	Composition Function 6 (n=5,Rotated)	1200
	27	Composition Function 7 (n=5,Rotated)	1300
	28	Composition Function 8 (n=5,Rotated)	1400

Algorithm 4 Framework of PSO_{mv}

```
1: Input: A swarm with $N$ particles;
2: Output: the global best $gbest$.
3: Randomly initialize a set of $N$ particles, and evaluate the fitness value of all particles;
4: Update all particles' $pbest$ and the global best $gbest$;
5: Set $\bar{w} = \bar{\alpha} = 0.5$, $Set_P = \emptyset$;
6: while stop condition is not reached do
7:     Sort particles from the worst to the best in terms of their $pbest$s' fitness values;
8:     for each particle $X_i$, $i = 1$ to $N$ do
9:         Get the parameters $w_i$ and $\alpha_i$ by Algorithm 3;
10:        Construct the continuous variables $C = (x_i^1, x_i^2, ..., x_i^Z)$ by Algorithm 1;
11:        Construct the discrete variables $D = (x_i^{Z+1}, x_i^{Z+2}, ..., x_i^{Z+V})$ by Algorithm 2;
12:        Set $X_i^{new} \preceq \leftarrow \preceq (C, D) = (x_i^1, x_i^2, ..., x_i^Z, x_i^{Z+1}, x_i^{Z+2}, ..., x_i^{Z+V})$;
13:        Evaluate the fitness value of offspring $X_i^{new}$;
14:        if $fitness(X_i) > fitness(X_i^{new})$ then
15:            $X_i = X_i^{new}$
16:        end if
17:        if $fitness(pbest_i) > fitness(X_i^{new})$ then
18:            $pbest_i = X_i^{new}$
19:            Update parameters $\bar{w}$, $\bar{\alpha}$ and $Set_P$ by Eq. 5 and Eq. 6, respectively;
20:        end if
21:    end for
22:    Update $gbest$;
23: end while
24: Return $gbest$.
```

global optimum is transferred into integer (randomly distributed in $[-80, 80]^V$) while discrete variables hold the original search range $[-100, 100]^V$ and rotation matrix. The total number of continuous variables and discrete variables is set to M, i.e., $M = Z + V$.

In this way, 28 artificial benchmarks are developed, the characteristics of these functions are shown in Table 1. Note that the first five functions ($f1$ to $f5$) are unimodal functions while the next 15 functions ($f6$ to $f20$) are multimodal functions. The rest of these functions ($f21$ to $f28$) are composition functions which include complex features of multiple functions.

4.2. EAs for comparison and parameter settings

In order to validate the efficiency and effectiveness of PSO_{mv}, four state-of-the-art algorithms for MVOPs, i.e., DE_{mv}, ACO_{mv}, $AEDA$ and $S-PSO$, are chosen to compare with PSO_{mv}. DE_{mv}, ACO_{mv} and $AEDA$ are specific designed for solving MVOPs and have been mentioned in section II. $S-PSO$ is designed for DOPs [4]. Different from the previous discrete version PSO variants, $S-PSO$ introduces a set-based theory and transforms the original position vector and velocity vector updating rules based on crisp sets. In order to guarantee that $S-PSO$ can handle continuous variables, we embed classical continuous version PSO in $S-PSO$.

All experimental results are produced after 30 independent runs. The termination condition for all algorithms is satisfied after 1×10^5 function evaluations. For the 28 artificial MVOPs, the number of continuous variables Z is set to 25 and the number of discrete variables V is set to 25, thus the total number of all decision variables D is 50. The parameters and operators for each algorithms is set as follows. DE_{mv} uses the "best/2" mutation operator and exponential crossover operator. The parameters F and CR are set to 0.2 with the aims to maintain diversity according to the author's discussion. In $S-PSO$, $c = 2.0$ and the inertia weight w linearly decreasing from 0.9 to 0.4 [34]. In ACO_{mv}, the quality factor q is set to 0.05099. In AEDA, the $NP-best$ is set to 0.45 and g_p is initialized as 0.5.

4.3. Results comparison on benchmark functions

The numerical results on 28 benchmarks obtained by PSO_{mv} and other compared algorithms are listed in Table 2. Many statistical analyses have been proposed in the literature for evaluating the algorithms

Table 2
The numerical results on 28 benchmarks.

Functions	PSO_{mv}	DE_{mv} [45]	ACO_{mv} [21]	$AEDA$ [33]	$S-PSO$ [4]
f_1 MEAN	3.71E-11	4.31E+03(+)	1.26E+02(+)	3.89E-02(~)	8.36E+02(+)
STD	4.63E-10	3.62E+03	2.13E+03	9.83E-01	1.01E+03
f_2 MEAN	3.39E+07	1.64E+08(+)	2.79E+08(+)	8.60E+07(+)	9.31E+07(+)
STD	8.88E+07	1.34E+08	6.75E+08	2.99E+08	2.50E+08
f_3 MEAN	1.53E+09	7.34E+10(+)	3.75E+10(+)	6.33E+08(-)	2.86E+10(+)
STD	6.39E+09	7.92E+10	1.07E+11	2.36E+09	4.99E+10
f_4 MEAN	1.00E+04	9.18E+04(+)	6.80E+04(+)	4.80E+04(~)	5.15E+04(~)
STD	9.47E+03	4.71E+04	3.05E+04	3.08E+04	5.30E+04
f_5 MEAN	2.44E-05	7.43E+02(+)	9.78E+00(~)	3.69E-01(~)	1.05E+02(+)
STD	1.73E-04	4.51E+02	7.66E+01	2.33E+00	1.34E+02
f_6 MEAN	2.866E+02	6.729E+02(+)	4.430E+02(+)	2.104E+02(-)	2.868E+02(~)
STD	5.53E+02	4.59E+02	8.00E+02	2.12E+02	2.74E+02
f_7 MEAN	2.88E+01	1.88E+02(+)	1.36E+02(+)	2.55E+01(-)	1.36E+02(+)
STD	2.88E+01	1.18E+02	1.63E+02	3.21E+01	1.35E+02
f_8 MEAN	2.1176E+01	2.1175E+01(~)	2.1185E+01(~)	2.1180E+01(~)	2.1176E+01(~)
STD	2.11E-01	2.47E-01	2.05E-01	1.71E-01	2.07E-01
f_9 MEAN	1.12E+01	6.58E+01(+)	6.93E+01(+)	6.40E+01(+)	6.60E+01(+)
STD	2.20E+01	9.55E+00	3.21E+01	1.11E+02	3.54E+01
f_{10} MEAN	2.08E-03	1.46E+03(+)	4.04E+02(+)	2.77E+01(+)	5.09E+02(+)
STD	3.51E-02	8.86E+02	2.01E+03	7.31E+01	6.06E+02
f_{11} MEAN	9.98E+01	3.82E+02(+)	4.09E+02(+)	1.57E+02(+)	1.90E+02(+)
STD	2.63E+02	1.15E+02	4.40E+02	1.11E+02	1.16E+02
f_{12} MEAN	3.55E+02	6.23E+02(+)	5.31E+02(+)	3.70E+02(~)	4.79E+02(+)
STD	1.12E+02	1.49E+02	2.54E+02	1.01E+02	2.43E+02
f_{13} MEAN	3.62E+02	6.34E+02(+)	5.26E+02(+)	3.71E+02(~)	5.10E+02(+)
STD	7.56E+01	1.61E+02	2.56E+02	7.72E+01	1.92E+02
f_{14} MEAN	1.81E+03	6.24E+03(+)	1.35E+04(+)	6.47E+03 (+)	2.45E+03(~)
STD	9.00E+03	1.44E+03	1.58E+03	1.68E+03	4.56E+03
f_{15} MEAN	1.28E+04	1.31E+04(~)	1.37E+04(~)	1.36E+04(~)	1.37E+04(~)
STD	2.08E+03	1.94E+03	2.28E+03	2.04E+03	2.52E+03
f_{16} MEAN	3.68E+00	3.75E+00(~)	3.92E+00(~)	3.73E+00(~)	3.80E+00(~)
STD	2.12E+00	2.08E+00	1.65E+00	1.27E+00	1.52E+00
f_{17} MEAN	9.38E+01	6.28E+02(+)	4.63E+02(+)	1.69E+02(+)	3.96E+02(+)
STD	3.11E+02	2.23E+02	7.83E+02	7.76E+01	3.60E+02
f_{18} MEAN	7.54E+01	6.40E+02(+)	4.51E+02 (+)	1.76E+02(+)	3.58E+02(+)
STD	3.10E+02	1.91E+02	7.99E+02	3.62E+02	2.84E+02
f_{19} MEAN	1.24E+01	3.71E+02(+)	4.75E+03(+)	1.62E+01(~)	5.80E+01(+)
STD	3.15E+01	6.69E+02	2.63E+04	1.15E+01	5.28E+01
f_{20} MEAN	1.917E+01	2.176E+01(~)	2.157E+01(~)	2.162E+01 (~)	2.161E+01(~)
STD	2.70E+00	1.22E+00	1.65E+00	1.32E+00	2.06E+00
f_{21} MEAN	2.0000E+02	4.7762E+02(+)	2.2783E+02(~)	2.0003E+02(~)	2.6648E+02(+)
STD	4.41E-10	1.84E+02	4.03E+02	1.71E-01	8.69E+01
f_{22} MEAN	1.85E+03	6.39E+03(+)	1.36E+04(+)	6.61E+03(+)	2.22E+03(~)
STD	9.50E+03	1.43E+03	2.27E+03	1.68E+03	3.08E+03
f_{23} MEAN	1.39E+04	1.62E+04(~)	1.64E+04(~)	1.64E+04 (~)	1.64E+04(~)
STD	3.29E+03	1.50E+03	1.16E+03	9.06E+02	1.84E+03
f_{24} MEAN	1.25E+03	1.74E+03(+)	1.68E+03(+)	1.43E+03(~)	1.65E+03(+)
STD	2.84E+02	2.75E+02	3.01E+02	2.96E+02	3.52E+02
f_{25} MEAN	5.33E+02	6.14E+02(+)	5.81E+02(~)	5.36E+02 (~)	5.60E+02(~)
STD	5.34E+01	5.44E+01	6.25E+01	6.83E+01	8.88E+01
f_{26} MEAN	2.90E+02	3.05E+03 (~)	9.54E+02(+)	7.70E+02(+)	1.51E+03(+)
STD	9.90E+02	2.33E+03	6.37E+03	8.39E+02	1.97E+03
f_{27} MEAN	5.35E+03	5.54E+03(-)	5.20E+03(-)	5.08E+03(-)	5.27E+03(-)
STD	5.97E+02	2.19E+02	1.36E+03	5.83E+02	8.07E+02
f_{28} MEAN	1.06E+04	1.12E+04(~)	1.13E+04(~)	1.15E+04(~)	9.78E+03(-)
STD	4.51E+03	1.45E+03	1.38E+03	1.18E+03	2.05E+03
$+/-/\sim$		20/1/7	18/1/9	9/4/15	16/2/10

[3,14]. In our experimental study, if the proposed PSO_{mv} significantly outperforms other compared algorithms, a notation $'+'$ is marked in front of the corresponding algorithm, if PSO_{mv} performs similarly to the compared algorithm, a notation $'='$ is marked and if PSO_{mv} is outperformed, a notation $'-'$ is marked. The statistical comparing results are shown in the last line of the table.

In Table 2, the mean errors and the standard deviations after 30 independent runs are shown in each row. The best results on the function are highlighted with dark shade while the second-best results are marked with a light shade. From Table 2, we can see that PSO_{mv} obtains the best result on 22 out of 28 functions (f_1, f_2, f_4, f_5 and f_9 to f_{26}), including four unimodal functions, 12 multimodal functions and 6 composition functions. As for the other functions, PSO_{mv} performs well and obtains the second-best results on 5 out of 6 functions (f_3, f_6, f_7, f_8 and f_{28}), including one unimodal function, three multimodal functions and one composition functions. These statistical results indicate that PSO_{mv} has excellent overall performance on most functions.

On f_3, f_6, f_7, f_{27} and f_{28}, $AEDA$ and $S-PSO$ show better performance than PSO_{mv}, but PSO_{mv} obtains the second-best results on these functions except f_{27}. ACO_{mv} and DE_{mv} performs a little worse than the other compared algorithms. These two algorithms only show a similar efficiency with PSO_{mv} on a few functions. The reason is that, DE_{mv} utilizes continuous version DE to handle continuous variables with fixed operators and parameters. The parameters of PSO_{mv} are adaptively ad-

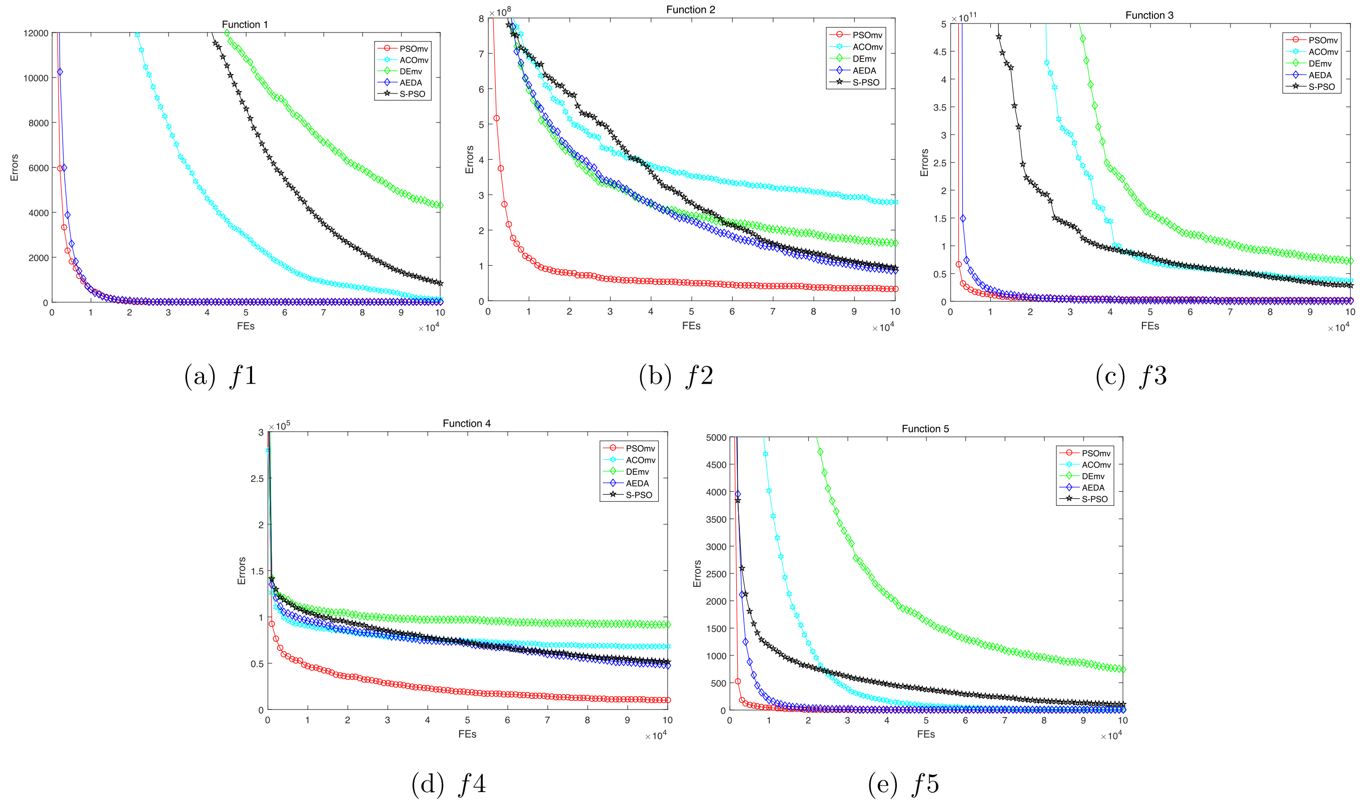


Fig. 3. Convergence process on 5 unimodal benchmark functions.

justed during the iteration by the APT strategy, which help PSO_{mv} handle more complex problems efficiently. ACO_{mv} utilizes a continuous relaxation mechanism to handle ordinal discrete variables with a simple round method, but the simple round method may produce invalid solutions and decline the efficiency of ACO_{mv}. Unlike ACO_{mv}, PSO_{mv} simultaneously uses continuous reproduction method and discrete reproduction method to separately handle continuous variables and discrete variables. The discrete variables are treated as categorical variables and does not need additional relaxation methods, which help guarantee the efficiency of PSO_{mv}.

Figs. 3 –5 show the convergence process of each algorithm. On the 5 unimodal benchmarks, PSO_{mv} converges quickly and significantly outperforms the compared algorithms on f_2 and f_4. On the 15 multimodal benchmarks, PSO_{mv} shows excellent performance and significantly outperforms the other algorithms on f_9, f_{14}, f_{15} and f_{20}. Note that PSO_{mv} not only maintains a good convergence rate but good precision of solutions on these 15 benchmarks. The reason can be concluded that the continuous reproduction method and the discrete reproduction method of PSO_{mv} ensure both convergence and diversity of swarm. Due to the reproduction methods guarantee the diversity and the convergence of swarm, PSO_{mv} is appropriate for handling multimodal optimization problems. On the last 8 composition benchmarks, the compared algorithms, e.g., ACO_{mv}, DE_{mv} and $S-PSO$, are easily get trapped into the local optima. Benefitting from the APT strategy, parameters in PSO_{mv} are adaptively changed according to the evolutionary state which ensure PSO_{mv} converges fast and obtains the best results on most of the 8 composition benchmarks.

4.4. Results comparison on real-world problems

In order to further validate the practical performance of PSO_{mv}, we do experiments on two real-world MVOPs. These two MVOPs also involve some constraints which further increase the complex of the search space.

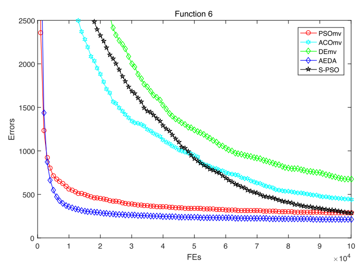
Table 3
Mathematical formulation of PVD-D.

min $f = 0.6224T_sRL + 1.7781T_hR^2 + 3.1611T_s^2L + 19.84T_s^2R$	
g_1	$-T_s + 0.0193R \le 0$
g_2	$-T_h + 0.00954R \le 0$
g_3	$-\pi R^2 L - \frac{4}{3}\pi R^3 + 750.1728 \le 0$
g_4	$L \le 240$
g_5	$0 \le T_s \le 100$
g_6	$0 \le T_h \le 100$
g_7	$10 \le R \le 200$
g_8	$10 \le L \le 200$

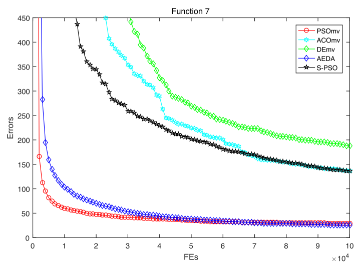
4.4.1. Results comparison on pressure vessel design problem

The Pressure Vessel Design Problem (PVD) arises from the manufacturing industry [9]. Engineers are required to design a cylindrical container containing two ellipses. The object of the problem is to minimize the manufacturing cost. There are four decision variables in PVD: the thickness of the container T_s, the thickness of the head T_h, the inner radius of the container R and the length of the container L. Note the variable R and the variable L are continuous while the thickness for variables T_s and T_h are a set of optional values which multiple 0.0625 inches. The case D of PVD is a more difficult problem because of the extended search space. In order to further test algorithms' performance, we chose PVD-D as the test problem which is shown in Table 3, where f is the objective function and g_1 to g_8 are constraints.

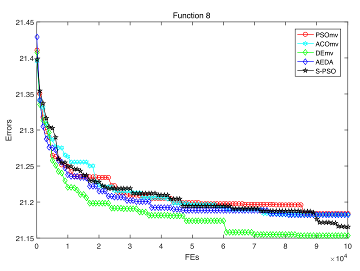
The convergence curve on PVD-D is illustrated in Fig. 6. Note that the best-known objective function value of PVD-D is 6059.7143. Obviously, PSO_{mv}, DE_{mv} and $S-PSO$ have obtained the best-known results. Besides, PSO_{mv} converged quickly and searched the best-known results after about 1×10^4 function evaluations. Since the PVD-D involves 8



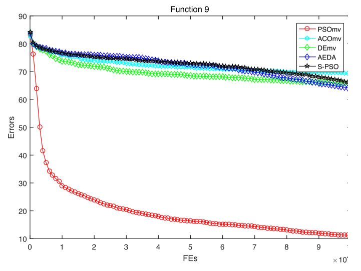
(a) $f6$



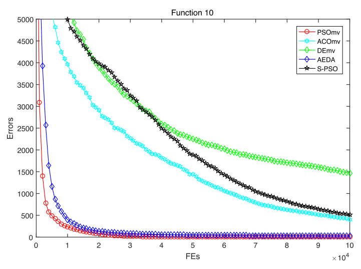
(b) $f7$



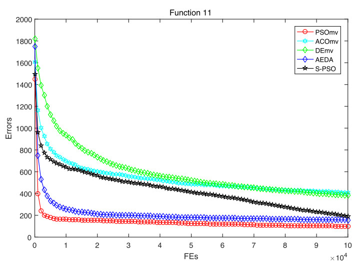
(c) $f8$



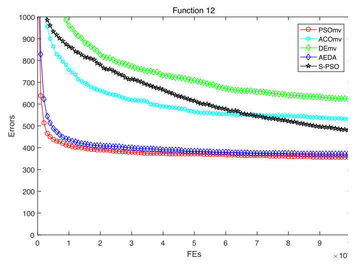
(d) $f9$



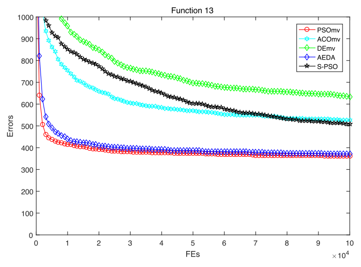
(e) $f10$



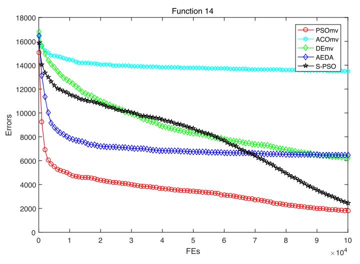
(f) $f11$



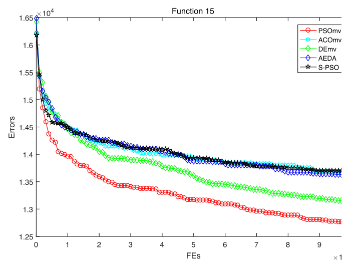
(g) $f12$



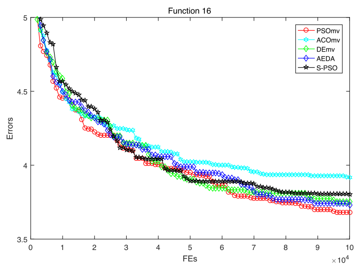
(h) $f13$



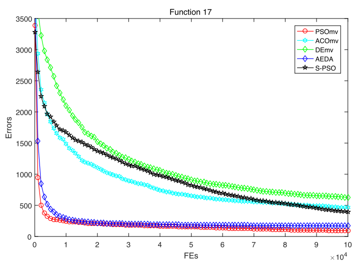
(i) $f14$



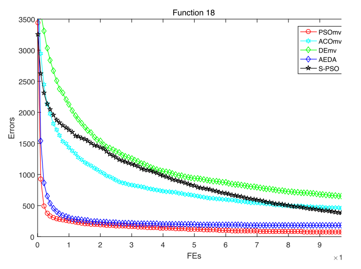
(j) $f15$



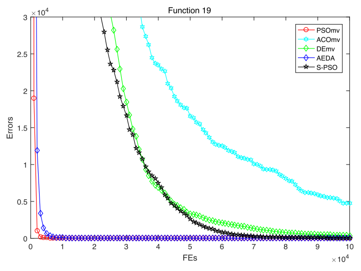
(k) $f16$



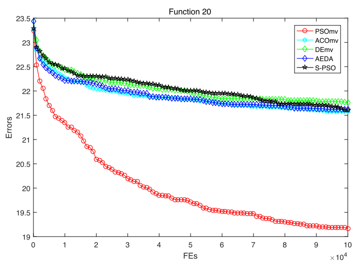
(l) $f17$



(m) $f18$



(n) $f19$



(o) $f20$

Fig. 4. Convergence process on 15 multimodal benchmark functions.

constraints which increase the complex of the search space, $AEDA$ and ACO_{mv} easily got trapped into the local optima. DE_{mv} and $S-PSO$ adopt set-based reproduction method to handle discrete variables, thus they showed a similar performance, but they reached the best-known after PSO_{mv}.

4.4.2. *Results comparison on coil spring design problem*

The Coil Spring Design Problem problem (CSD) aims to design a helical compression spring that can support axial load [6,9]. The objective is to minimize the wire used in making the spring. In CSD, the decision variables are the number of spring coils, denoted as N, the outer

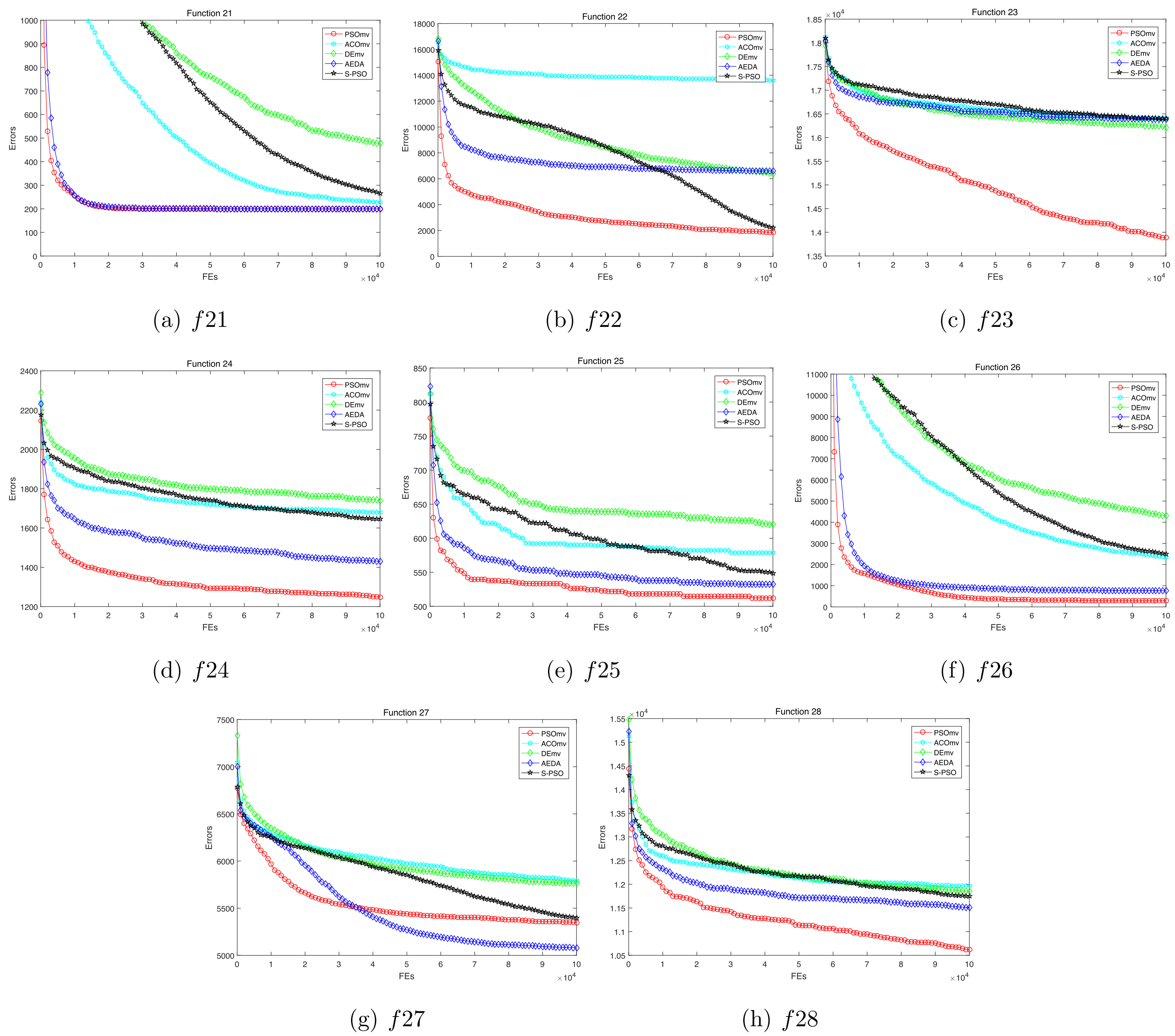


Fig. 5. Convergence process on 8 composition benchmark functions.

Table 4
Standard Wire Diameters Available for the Spring Coil.

Optional values of wire diameters[inch]					
0.0090	0.0095	0.0104	0.0118	0.0128	0.0132
0.0140	0.0150	0.0162	0.0173	0.0180	0.0200
0.0230	0.0250	0.0280	0.0320	0.0350	0.0410
0.0470	0.0540	0.0630	0.0720	0.0800	0.0920
0.1050	0.1200	0.1350	0.1480	0.1620	0.1770
0.1920	0.2070	0.2250	0.2440	0.2630	0.2830
0.3070	0.3310	0.3620	0.3940	0.4375	0.5000

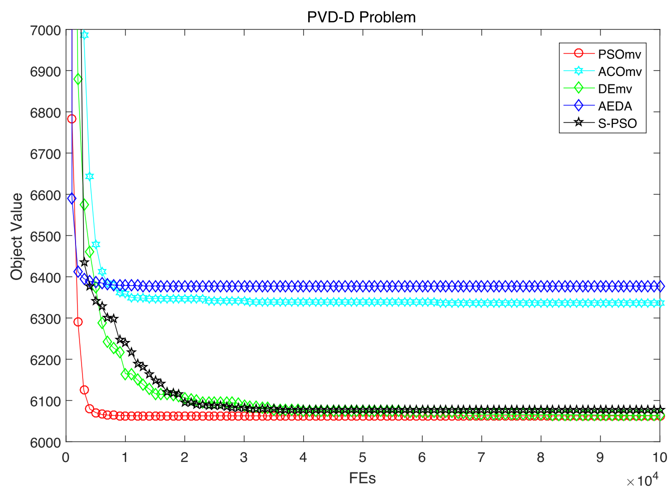
Table 5
Mathematical formulation of CSD.

min $f_c(N,D,d) = \frac{\pi^2 Dd^2(N+2)}{4}$	
g1	$\frac{8C_f F_{max} D}{\pi d^3} - S \le 0$
g2	$l_f - l_{max} \le 0$
g3	$d_{min} - d \le 0$
g4	$D - D_{max} \le 0$
g5	$3.0 - \frac{D}{d} \le 0$
g6	$\sigma_p - \sigma_{pm} \le 0$
g7	$\sigma_p + \frac{F_{max}-F_p}{K} + 1.05(N+2)d - l_f \le 0$
g8	$\sigma_w - \frac{F_{max}-F_p}{K} \le 0$
where	$C_f = \frac{4\frac{D}{d}-1}{4\frac{D}{d}-4} + \frac{0.0615d}{D}$
	$K = \frac{Gd^4}{8ND^3}$
	$\sigma_p = \frac{F_p}{K}$
	$l_f = \frac{F_{max}}{K} + 1.05(N+2)d$

diameter of the spring D and the diameter of the wire d. Note that the numbers of spring coils are integer variables, the outer diameter of the spring D is continuous, the diameter of the wire d is discrete variables whose optional values are shown in Table 4. The best-known objective function value of CSD problem is 2.65856.

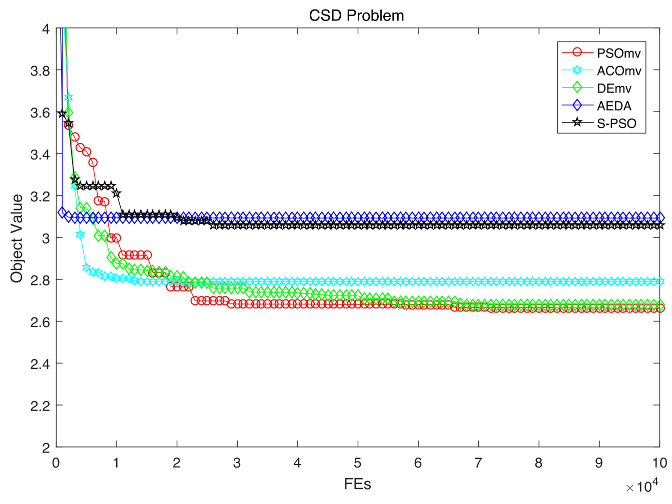
Accordingly, this problem can be detailed in Table 5.

The convergence curve on CSD is shown in Fig. 7. Note that the best-known objective function value of CSD problem is 2.65856. Obviously, only PSO_{mv} reached the best result. The other compared algorithms converge quickly but get trapped into the local optima.



(a)

Fig. 6. Convergence curve on PVD-D.



(a)

Fig. 7. Convergence curve on CSD.

5. Conclusion and future work

In this paper, we propose a novel PSO algorithm for MVOPs, namely PSO_{mv}. In order to simultaneously handle discrete variables and continuous variables, PSO_{mv} utilizes a mixed-variable encoding scheme, which guarantees that PSO_{mv} can adopt hybrid reproduction method to respectively solve mixed variables, i.e., the continuous reproduction method (CR) and discrete reproduction method (DR). The CR method randomly selects excellent learning objects for particles, thus the convergence and the diversity of the swarm can be guaranteed. The DR method generates offsprings by considering both historical search information and current swarm's search information. In addition, to effectively adjust parameters in the reproduction methods, an adaptive parameter tuning strategy (APT) is proposed, which updates parameters adaptively. The constraints handling method is proposed to further improve the practical performance. In the experimental study, to evaluate the performances of PSO_{mv}, we first generate 28 benchmark functions from CEC2013 test suite by transferring the continuous shift global optimum into integer. Compared with four state-of-the-art evolutionary algorithms for MVOPs, PSO_{mv} shows competitive performance with better results and faster converge speed in most cases. Then we further testify it on two real-world MVOPs, the results also validate the performance of PSO_{mv}. However, the correlation between continuous and discrete variables has been overlooked by the reproduction methods of PSO_{mv}, the performance of PSO_{mv} can not be guaranteed when facing much more complex MVOPs.

Due to the different landscapes of continuous variables and discrete variables, we have not yet found a suitable unified model to handle the correlation between different kind variables. However, considering the correlation between discrete and continuous variables will truly promote the theoretical performance of the algorithm. In the future, we will try to design reproduction methods which can consider the correlation between continuous and discrete variables. And we will continue to investigate the applications of the proposed PSO_{mv} algorithm on more complex real-world MVOPs.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Feng Wang: Conceptualization, Methodology, Writing - original draft, Writing - review & editing. **Heng Zhang:** Writing - review & editing. **Aimin Zhou:** Writing - review & editing, Formal analysis.

Acknowledgments

This work is supported by National Nature Science Foundation of China [Grant nos. 61773296, 61673180].

References

[1] N.H. Awad, M.Z. Ali, R. Mallipeddi, P.N. Suganthan, An efficient differential evolution algorithm for stochastic OPF based active-reactive power dispatch problem considering renewable generators, Appl. Soft Comput. 76 (2019) 445–458.
[2] P.P. Biswas, P.N. Suganthan, R. Mallipeddi, G.A.J. Amaratunga, Optimal reactive power dispatch with uncertainties in load demand and renewable energy sources adopting scenario-based approach, Appl. Soft Comput. 75 (2019) 616–632.
[3] J. Carrasco, S. García, M. Rueda, S. Das, F. Herrera, Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review, Swarm Evol. Comput. 54 (2020) 100665.
[4] W.N. Chen, J. Zhang, H. Chung, W. Zhong, W.-L. Wu, Y.-G. hui Shi, A novel set-based particle swarm optimization method for discrete optimization problems, IEEE Trans. Evol. Comput. 14 (2) (2014) 278–300.
[5] R. Cheng, Y. Jin, A social learning particle swarm optimization algorithm for scalable optimization, Inf. Sci. 291 (2015) 43–60.
[6] D. Datta, J.R. Figueira, A real-integer-discrete-coded differential evolution algorithm: a preliminary study, in: European Conference on Evolutionary Computation in Combinatorial Optimization, Springer, 2010, pp. 35–46.
[7] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. 1 (1) (2011) 3–18.
[8] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1) (1997) 53–66.
[9] S. E, Nonlinear integer and discrete programming in mechanical design optimization, J. Mech. Des. 112 (2) (1990) 223–229.
[10] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of International Symposium on Human Science, 2002, pp. 39–43.
[11] L. Gao, A. Hailu, Comprehensive learning particle swarm optimizer for constrained mixed-variable optimization problems, Int. J. Comput. Intell.Syst. 3 (2010) 832–842.
[12] D.E. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, Addison-Wesley, 1989.
[13] C. Guo, J. Hu, B. Ye, Y. Cao, Swarm intelligence for mixed-variable design optimization, J. Zhejiang Univ. 5 (7) (2004) 851–860.
[14] A.H. Halim, I. Ismail, S. Das, Performance assessment of the metaheuristic optimization algorithms: an exhaustive review, Artif. Intell. Rev. (2020).
[15] C.F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, IEEE Trans. Syst. Man Cybern. 34 (2) (2004) 997–1006.
[16] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, IEEE Int. Conf. Syst. ManCybern. 5 (1997) 4104–4108.
[17] J. Kennedy, M. Rui, Population structure and particle swarm performance, in: Proceedings of the Congress on Evolutionary Computation (CEC2002), 2002, pp. 1671–1676.

[18] J. Lampinen, I. Zelinka, Mixed integer-discrete-continuous optimization by differential evolution, in: Proceedings of the 5th International Conference on Soft Computing, 1999, pp. 71–76.
[19] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (3) (2006) 281–295.
[20] J.J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer with local search, in: Proceedings of the Congress on Evolutionary Computation (CEC2005), 2005, pp. 124–129.
[21] T. Liao, K. Socha, M.A. Montes de Oca, T. Stutzle, M. Dorigo, Ant colony optimization for mixed-variable optimization problems, IEEE Trans. Evol. Comput. 18 (4) (2014) 503–518.
[22] Y. Liu, W. Chen, Z. Zhan, Y. Lin, Y. Gong, J. Zhang, A set-based discrete differential evolution algorithm, in: International Conference on Systems, Man, and Cybernetics (SMC 2013), 2013, pp. 1347–1352.
[23] N. Lynn, M.Z. Ali, P.N. Suganthan, Population topologies for particle swarm optimization and differential evolution, Swarm Evol. Comput. 39 (2018) 24–35.
[24] M.H. Mashinchi, O.M. A., W. Pedrycz, Hybrid optimization with improved tabu search, Appl. Soft Comput. 11 (2011) 1993–2006.
[25] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Trans. Evol. Comput. 8 (3) (2004) 204–210.
[26] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, second ed., Springer, 1994.
[27] W. Pang, K.P. Wang, C.G. Zhou, L.J. Dong, M. Liu, H.Y. Zhang, J.Y. Wang, Modified particle swarm optimization based on space transformation for solving traveling salesman problem, in: Proceedings of the 2004 International Conference on Machine Learning and Cybernetics (ICMLC2004), 2004, pp. 2342–2346.
[28] S.S. Rao, X. Y., A hybrid genetic algorithm for mixed-discrete design optimization, J. Mech. Des. 127 (6) (2005) 1100–1112.
[29] A. Ratnaweera, S.K. Halgamuge, H.C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evol. Comput. 8 (3) (2004) 240–255.
[30] A. Salman, I. Ahmad, S. Al-Madani, Particle swarm optimization for task assignment problem, Microprocessors Microsyst. 26 (8) (2002) 363–371.
[31] D.Y. Sha, C.Y. Hsu, A hybrid particle swarm optimization for job shop scheduling problem, Comput. Ind. Eng. 51 (4) (2012) 791–808.
[32] P.S. Shelokar, P. Siarry, V. Jayaraman, B. Kulkarni, Particle swarm and ant colony algorithms hybridized for improved continuous optimization, Appl. Math. Comput. 188 (1) (2007) 129–142.
[33] W. Shi, W. Chen, Y. Lin, T. Gu, S. Kwong, J. Zhang, An adaptive estimation of distribution algorithm for multipolicy insurance investment planning, IEEE Trans. Evol. Comput. 23 (1) (2019) 1–14.
[34] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: Proceeding of IEEE International Conference on Entertainment Computing (ICEC1998), 1998, pp. 69–73.
[35] Y. Shi, R.C. Eberhart, Fuzzy adaptive particle swarm optimization, in: Proceedings of the 2001 Congress on Evolutionary Computation, 2001, pp. 101–106.
[36] R. Storn, K. Price, Differential evolution a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359.
[37] J. Sun, H. Zhang, A. Zhou, Q. Zhang, K. Zhang, A new learning-based adaptive multi-objective evolutionary algorithm, Swarm Evol. Comput. 44 (2019) 304–319.
[38] F. Wang, Y. Li, F. Liao, H. Yan, An ensemble learning based prediction strategy for dynamic multi-objective optimization, Appl. Soft Comput. 96 (2020) 106592, doi:10.1016/j.asoc.2020.106592.
[39] F. Wang, Y. Li, H. Zhang, T. Hu, X.L. Shen, An adaptive weight vector guided evolutionary algorithm for preference-based multi-objective optimization, Swarm Evol. Comput. 49 (2019) 220–233.
[40] F. Wang, Y. Li, A. Zhou, K. Tang, An estimation of distribution algorithm for mixed–variable newsvendor problems, IEEE Trans. Evol. Comput. 24 (3) (2020) 479–493.
[41] F. Wang, H. Zhang, K. Li, Z. Lin, X.L. Shen, A hybrid particle swarm optimization algorithm using adaptive learning strategy, Inf. Sci. 436–437 (2018) 162–177.
[42] K.P. Wang, N.L. Huang, C.G. Zhou, N.W. Pang, Particle swarm optimization for traveling salesman problem, in: Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (ICMLC2003), 2003, pp. 1583–1585.
[43] G. Wu, R. Mallipeddi, P.N. Suganthan, Ensemble strategies for population-based optimization algorithms – a survey, Swarm Evol. Comput. 44 (2019) 695–711.
[44] X. Yan, J. Zhao, C. Hu, D. Zeng, Multimodal optimization problem in contamination source determination of water supply networks, Swarm Evol. Comput. 47 (2019) 66–71.
[45] L. Ying, L. Yu, W Chen, J. Zhang, A hybrid differential evolution algorithm for mixed-variable optimization problems, Inf. Sci. (2018) 177–188.
[46] A. Zhou, J. Sun, Q. Zhang, An estimation of distribution algorithm with cheap and expensive local search methods, IEEE Trans. Evol. Comput. 19 (2015) 807–822.