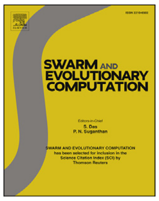
Swarm and Evolutionary Computation 49 (2019) 158–177

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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



An improved artificial bee colony algorithm combined with extremal optimization and Boltzmann Selection probability

Min-Rong Chen [a,*], Jun-Han Chen [b], Guo-Qiang Zeng [c,**], Kang-Di Lu [d], Xin-Fa Jiang [a]

[a] *School of Computer Science, South China Normal University, Guangzhou, 510631, China*
[b] *College of Information Science and Technology/College of Cyber Security, Jinan University, Guangzhou, 510632, China*
[c] *National-Local Joint Engineering Laboratory of Digitalize Electrical Design Technology, Wenzhou University, Wenzhou, 325035, China*
[d] *Department of Automation, College of Information Sciences and Technology, Donghua University, Shanghai, 201620, China*

ARTICLE INFO

Keywords:
Artificial bee colony
Extremal optimization
Unconstrained continuous optimization problems
Boltzmann selection probability

ABSTRACT

Artificial Bee Colony (ABC) algorithm is an optimization algorithm based on a particular intelligent behavior of honeybee swarms. The standard ABC has been utilized to deal with a lot of optimization problems in real world. However, there are still some defects of the standard ABC such as weak local-search capability and low solution precision. In order to improve the performance of ABC, in this paper, we propose two improved versions of ABC-EO and IABC-EO presented in our previous work, called ABC-EO II and IABC-EO II, where Extremal Optimization (EO) is introduced to ABC and IABC in different ways. There are some advanced characteristics of our proposed algorithms: (1) Compared with ABC-EO and IABC-EO, the improved versions have lower computational costs by introducing EO in different ways; (2) An easier-operated mutation method is introduced which can increase the diversity of new offspring and helps our algorithms jump out of local optima; (3) The selection pressure can be dynamically adjusted in evolutionary process by means of Boltzmann selection probability; (4) A novel selection probability is used to select the worse solutions for the mutation operation by EO mechanism. The experimental results on three groups of benchmark functions indicate that the performance of the proposed algorithms is as good as or superior to those of 15 state-of-the-art optimization algorithms in terms of solution accuracy, convergence speed, successful rate and statistical tests. Finally, in order to testify the feasibilities of the proposed methods for solving the real life problems, our algorithms are applied to solving two kinds of parameters identification of photovoltaic models and four well-recognized evolutionary algorithms are selected as the competitors. The simulation results indicate that the proposed IABC-EO II algorithm has superior performance in comparison with other five algorithms, while the proposed ABC-EO II outperforms at least competitive with other four algorithms in term of solution accuracy and statistical tests. As a result, our algorithms may be good alternatives for solving complex unconstrained continuous optimization problems.

1. Introduction

Artificial Bee Colony (ABC) algorithm, first introduced by Karaboga in 2005 [1], is a new branch of evolutionary algorithms (EAs) that is inspired by the collective foraging behavior of real honey bee colonies. Compared with some popular EAs, such as Particle Swarm Optimization (PSO), Genetic Algorithm (GA) and Differential Evolution (DE), this algorithm obtains superior or comparable performance [2–4]. Recently, Rajasekhar et al. [5] have provided a comprehensive survey on the algorithms inspired by the honey bees, and they outlined the major bees-inspired algorithms, their prospects in the respective problem domains and their similarities and dissimilarities with the other swarm intelligence algorithms, readers may get more details about ABC algorithm from the literature [5]. It is well known that ABC algorithm owns many obvious advantages, such as strong global-searching ability, fewer control parameters and simple in concept. However, there are also some limitations in the standard ABC, such as low convergence accuracy, and slow convergence speed at the later stage of evolution. Thus it has attracted more and more researchers to investigate the improved versions of ABC algorithm and their applications to a variety of fields [6–24]. Akbari et al. [6] proposed a multi-objective optimization method based on the ABC, called MOABC, which used a grid-based approach to

* Corresponding author.
** Corresponding author.
E-mail addresses: mrongchen@126.com (M.-R. Chen), zeng.guoqiang5@gmail.com (G.-Q. Zeng).

https://doi.org/10.1016/j.swevo.2019.06.005
Received 17 July 2018; Received in revised form 24 April 2019; Accepted 17 June 2019
Available online 25 June 2019

adaptively assess the Pareto front maintained in an external archive. In Ref. [7], an enhanced ABC algorithm was designed with a series of modifications to treat the order acceptance and scheduling (OAS) problem. In Ref. [8], ABC algorithm was modified with multiple update rules and K-opt operation to solve the Traveling Salesman Problem. A modified ABC algorithm, called MABC, was proposed to deal with the multiband cooperative spectrum sensing problem in cognitive radio network in Ref. [9]. Saad et al. [10] presented a goal programming-based multi-objective ABC optimization (MOABC) algorithm to solve the problem of topological design of distributed local area networks. A novel co-variance guided ABC algorithm (M-CABC) was introduced for portfolio optimization problem [11]. Zhou et al. [12] proposed a new approach, called multi-population parallel self-adaptive differential ABC (MPsaDABC) algorithm to solve cloud manufacturing. In Ref. [13], Karaboga et al. presented an Adaptive and Hybrid Artificial Bee Colony (aABC) algorithm to train adaptive-network-based fuzzy inference system. In Ref. [14], the authors proposed a multi-objective ABC optimization method called ε-MOABC based on performance indicators to solve multi-objective and many-objective problems. Inspired by the gravity model, Xiang et al. [15] proposed an attractive force model for choosing a better neighbor of a current individual to improve the exploitation ability of ABC. In Ref. [16], Gao et al. presented a novel ABC method called EABC to improve the performance of ABC. They also presented a new algorithm, i.e., DGABC, which combines DE with gbest-guided ABC (GABC) by an evaluation strategy with an attempt to utilize more prior information of the previous search experience to speed up the convergence [17]. To obtain a good balance between exploration and exploitation of ABC, in Ref. [18], Cui et al. mainly introduced an adaptive method into the ABC for the population size. Cui et al. [19] proposed a general dual-population framework (DPF) for ABC from the perspective of resource allocation to enhance its convergence speed. In Ref. [20], Cui et al. proposed a novel ranking-based adaptive ABC algorithm, called ARABC. The idea behind the algorithm is that the higher ranking a solution obtains, the more opportunities it will have in being chosen to generate new solutions. Inspired by the natural phenomenon that honey bees follow the elite group in the foraging process, Kong et al. [21] proposed a novel ABC algorithm named ECABC based on elite group guidance and the combined breadth-depth search strategy. In order to improve the exploitation ability of ABC and accelerate its convergence, Liang et al. [22] proposed an enhanced ABC algorithm named ABCADE, which remedies the limitation of ABC by exploiting the advantage of differential operators. Ding et al. [23] presented an approach for structural damage detection using the ABC algorithm with hybrid search strategy based on modal data. In order to improve the convergence speed of ABC, Lin et al. [24] proposed a novel ABC algorithm with local and global information interaction. It is worth mentioning that, in 2012, an improved ABC algorithm, called IABC, was proposed by Gao et al. [25], in which the search way of employed bees is changed. According to the experimental results in Ref. [25], IABC algorithm is superior to the standard ABC in terms of convergence speed and global-search capability.

As a novel meta-heuristic optimization algorithm originally inspired by far-from-equilibrium dynamics of Self-Organized Criticality (SOC) [26,27], Extremal Optimization (EO) [28,29] provides a novel insight into optimization domain due to the fact that only those worse or the worst variables in the sub-optimal solutions are selected to be mutated, instead of favoring the good. Furthermore, there are many advanced advantages in EO, e.g., only selection and mutation operators, and fewer parameters to be tuned. The mechanism of EO can be characterized from the perspectives of statistical physics, biological co-evolution and ecosystem [30]. In Refs. [31,32], the EO mechanism is extended to solve the multi-objective optimization problems. To overcome the weakness of a single individual, the authors in Refs. [33,34] used the population-based EO (PEO) algorithm to handle the single-objective optimization and multi-objective optimization problems for some benchmark functions, respectively. Li et al. [35] improved the shuffled frog-leaping algorithm with the merits of search ability of EO procedure. Based on the algorithm in Ref. [34], Zeng et al. [36] proposed an improved multi-objective PEO with the polynomial mutation for multi-objective optimization problems. In addition, the EO mechanism is applied to designing the fractional-order PID controller for automatic voltage regulator system [37]. Overall, the basic EO and its modified versions have been successfully applied to a variety of continuous and discrete optimization problems, and the interested readers can obtain more details from this literature [38], which systematically introduced the EO and its application.

It is worth mentioning that the PSO-EO algorithm [39] by introducing EO to PSO has overcome the limitation of PSO in premature convergence for complex multimodal functions due to the strong local-search ability of EO. For the purpose of overcoming the shortcomings of the standard ABC, inspired by the hybrid PSO-EO algorithm [39], an idea of combining ABC with EO was addressed in our previous work published in CEC 2014 [40], in which we developed two hybrid optimization methods, called ABC-EO and IABC-EO, through introducing EO to ABC by updating the positions of all food sources when the global optimal position is not improved for several iterations. The hybrid approaches elegantly combine the exploration ability of ABC or IABC with the strong exploitation ability of EO. In other words, ABC or IABC contributes to the hybrid approaches in a way to ensure that the search converges faster, while EO makes the search to jump out of local optima due to its strong local-search ability.

However, in ABC-EO and IABC-EO, EO is introduced to update the positions of all food sources when the global optimal position is unchanged for several iterations. Although the introducing of EO to ABC or IABC can enhance the local-search ability of ABC, this will result in high computational cost. Moreover, the EO procedures in ABC-EO and IABC-EO adopt Gaussian-Cauchy mutation (G-C mutation), which needs to determine when to exchange each other between Gaussian and Cauchy mutations. The exchanging between Gaussian and Cauchy mutations will also cause higher computational cost. In order to decrease the computational cost of ABC-EO and IABC-EO, in this paper, we present two improved versions of ABC-EO and IABC-EO, called ABC-EO II and IABC-EO II, where EO is introduced to ABC and IABC in different ways.

In brief, the main contributions of this work are summarized as follows:

(1) Unlike ABC-EO and IABC-EO which introduce EO to ABC for all food sources every few iterations, in this paper, EO is used to change the position of only one food source each iteration. Therefore, the computational costs of our algorithms are much lower than those of ABC-EO and IABC-EO.
(2) Since there is merely mutation operator in EO, the mutation plays a crucial role in EO procedure and it will influence the performance of EO significantly. In this work, through selecting neighbor solutions randomly, a novel mutation operator is employed, which is capable of increasing the diversity of the new offspring and hence helps our algorithms jump out of local optima. Furthermore, compared with G-C mutation [39] in ABC-EO and IABC-EO, this mutation operator in our work doesn't need to consider when to exchange each other between Gaussian and Cauchy mutations, and therefore it is much easier to operate.
(3) It is well-known that an appropriate selection pressure plays an important role in the search abilities of an optimization algorithm. Great selection pressure can strengthen the fast convergence ability of the algorithm. However, it may mislead the algorithm to be trapped into local optima. On the contrary, small selection pressure helps to enhance the global-search ability of an algorithm, but too small selection pressure will result in slow convergence speed. In order to overcome this contradiction, we adopt Boltzmann selection probability [41] as the basis for an artificial onlooker bee to choose one food source, and as a

consequence, the selection pressure of our proposed algorithms can be dynamically adjusted in the process of evolution.

(4) Finally, we propose a novel selection probability. The smaller the fitness value of the solution is, the more chance it has to be changed by EO. As a result, those worse solutions have more opportunities to be mutated. This mechanism is more in line with the essence of EO which always chooses the worst or worse solutions to perform mutation. This is another advanced characteristic different from ABC-EO and IABC-EO.

To validate the performance of the proposed approaches, we considered three groups of unimodal/multimodal benchmark functions, and compared with other 15 state-of-the-art optimization algorithms, including three EO variants, i.e., Population-based EO (PEO) [34], Particle Swarm Optimizer Hybridized with Extremal Optimization (PSO-EO) [39], Real-coded Population-based EO Algorithm with Polynomial Mutation (RPEO-PLM) [42], and six ABC-based algorithms, i.e., Artificial Bee Colony Algorithm (ABC) [40], Gbest-guided Artificial Bee Colony Algorithm (GABC) [43], Artificial Bee Colony Algorithm with Powell's Method (PABC) [44], Inspired Artificial Bee Colony Algorithm (IABC) [25], Inspired Artificial Bee Colony Algorithm with Integration of Extremal Optimization (IABC-EO) [40] and Artificial Bee Colony Algorithm with Integration of Extremal Optimization (ABC-EO) [40]. Furthermore, the proposed algorithms are also compared with five well-known algorithms, i.e., Real-coded Genetic Algorithm with Adaptive Directed Mutation (RCGA-ADM) [45], Intelligent Evolutionary Algorithms (IEA) [46], Differential Evolution Algorithm (DE) [47], Particle Swarm Optimizer (PSO) [39], and Genetic Algorithm (GA) [39]. Besides, we consider three well-recognized swarm intelligence algorithms including two new variants of DE as the competitors to further demonstrate the performance of the proposed algorithms. These competitors are Weighted Differential Evolution Algorithm (WDE) [48], Ensemble of Differential Evolution Variants (EDEV) [49], and Improved JAYA (IJAYA) [50]. The experimental results show that the proposed algorithms are good as or better than the other algorithms in terms of solution accuracy, convergence speed, successful rate and statistical tests. Finally, to demonstrate the feasibilities of the proposed methods in solving the real word problems, two kinds of parameters identification of photovoltaic models are used. Two recently presented DE variants, i.e., WDE [48], EDEV [49], one PSO variant, i.e., Particle Swarm Optimizer Hybridized with Differential Evolution (PSO-DE) [51], and IJAYA [50], are selected as the competitors. The simulation results indicate that IABC-EO II and ABC-EO II perform well or better in comparison with other four algorithms when solving the above real life problems. More specifically, IABC-EO II shows the best performance among the six algorithms while the ABC-EO II performs better than other three algorithms and competitive with EDEV. As a consequence, our algorithms can be regarded as good alternatives for solving complex unconstrained continuous optimization problems.

This paper is organized as follows. In Section 2, the standard ABC and EO algorithms are briefly introduced. In Section 3, we present ABC-EO II and IABC-EO II algorithms and describe them in detail. In Section 4, the proposed approaches are used to solve three groups of unconstrained continuous benchmark functions from the usual literature. Furthermore, to demonstrate the performance of the proposed methods, we select two kinds of parameters identification of photovoltaic models as real life problems and compare our proposed algorithms with two DE variants, one PSO variant and IJAYA. Finally, the conclusion and future work are given in Section 5.

2. Artificial bee colony and extremal optimization

2.1. The standard ABC algorithm

ABC algorithm is a meta-heuristic optimization algorithm based on swarm intelligence, and it is motivated by the intelligent foraging behavior of a bee colony [1]. The artificial bees of the swarm are categorized into three groups: employed bees, onlooker bees and scout bees. Employed bees are responsible for seeking food sources, while in the hive, onlooker bees are waiting for the information shared by employed bees and then make a decision to choose which food sources for further exploitation. If the employed bee is unable to find a better food source for a predefined number of trials, then the food source will be abandoned and the corresponding employed bee becomes a scout bee. Exploration search is performed by scout bees, while employed bees and onlooker bees are responsible for the exploitation of food sources. Half of the colony is composed of employed bees and the rest consists of onlooker bees. That is to say, the population size of employed bees is equal to the number of food sources, and also equal to that of onlooker bees.

The standard ABC algorithm consists of four phases, named Initialization Phase, Employed Bees Phase, Onlooker Bees Phase and Scout Bees Phase. The pseudo-code of the standard ABC algorithm is described in Fig. 1 [52].

(1) Initialization Phase

A group of $SN/2$ food sources representing possible solutions (where SN is the population size of the colony) is randomly produced by the following expression:

$$X_{i,j} = x_j^{\max} + rand(0,1)\left(x_j^{\max} - x_j^{\min}\right) \tag{1}$$

where $i = 1, 2, \ldots, SN/2$ and D denotes the number of problem dimension, $rand(0,1)$ is a uniformly distributed random number in [0,1], $x_j^{\min}$ and $x_j^{\max}$ represent the lower bound and upper bound of the j-th variable, respectively.

After all the food sources are assigned to the employed bees, the fitness of the solutions is calculated by the following equation [52], which is proportional to the nectar amount of that food source.

$$fit_i = \begin{cases} 1/(1+OBJ(X_i)), & \text{if} \quad OBJ(X_i) \geq 0 \\ 1+abs(OBJ(X_i)), & \text{if} \quad OBJ(X_i) < 0 \end{cases} \tag{2}$$

where fit_i and $OBJ(X_i)$ are the fitness value and the objective value of the solution X_i respectively, and abs(OBJ (X_i)) is the absolute value of $OBJ(X_i)$.

(2) Employed Bees Phase

```
1. Initialize food sources by Equation (1);
2. Apply fitness evaluation on all food sources and store the global best food source in the memory;
3. Set iteration =1;
4. Employed Bees Phase;
5. Onlooker Bees Phase;
6. Scout Bees Phase;
7. Update the global best food source;
8. If the terminal condition is not satisfied, then set iteration = iteration +1, and go to Step 4; otherwise, go to the next step;
9. Output the optimal solution and the optimal objective value.
```

Fig. 1. Pseudo-code of the standard ABC algorithm.

In this phase, employed bees will search the whole space for food sources. Each food source (X_i) is assigned to only one employed bee, and then a new food source ($X_i^{'}$) is produced in the neighborhood of the food sources X_i and X_k according to the following expression [52]:

$$X_{i,j}^{'} = X_{i,j} + \varphi_{i,j}\left(X_{i,j} - X_{k,j}\right) \tag{3}$$

where $k \in \{1, 2, \ldots, SN/2\}$ and $j \in \{1, 2, \ldots, D\}$ are the randomly chosen indexes, k has to be different from i, and $\varphi_{i,j}$ is a uniformly distributed random number between [-1,1]. After that, a greedy selection is performed between X_i and X_k according to their fitness values. After all employ bees complete their search, they will share their information about the nectar amount of food sources with onlooker bees.

The pseudo-code of Employed Bees Phase of ABC algorithm is described as Fig. 2 [52].

(3) Onlooker Bees Phase

Through the evaluation of the nectar information received from all employed bees, onlooker bees will select a food source with a probability (denoted as P) proportionate to its nectar amount (i.e., the fitness value of that food source). The selection probability P_i of the i-th food source is determined by the following equation [4]:

$$P_i = \frac{fit_i}{\sum_{n=1}^{SN/2} fit_n} \tag{4}$$

where fit_i is the fitness value of the solution X_i. As a result, the larger nectar amount one food source contains, the larger probability it will be chosen by onlooker bees. Then, a new food source $X_i^{'}$ is generated based on the selected food source X_i and its neighbor X_k according to Equation (3). Similar to Employed Bees Phase, a greedy selection is also carried out between X_i and X_k.

The pseudo-code of Onlooker Bees Phase of ABC algorithm is shown as Fig. 3 [52]. In this paper, the parameter "*trial*" is used to record the non-improvement number of one food source.

(4) Scout Bees Phase

The number of scout bees is not defined beforehand in the colony. A scout bee is generated according to the situation of a food source whether it is abandoned or not. If the employed bee is unable to find a better food source for a predetermined number of trials (which is defined by a control parameter called "*limit*"), then the food source is abandoned and the employed bee associated with it becomes a scout bee. Subsequently, the scout bee randomly searches a new solution in the whole space by the following equation [4] and it turns into an employed bee again.

```
1. For $i$=1 to $SN/2$ do
2. One dimension $j$ of the current food source and one neighbor of the current food source (assume that $k$ is the index of that neighbor, $k$ is not equal to $i$) are randomly selected, respectively, where $k \in \{1, 2, ..., SN/2\}$ and $j \in \{1, 2, ..., D\}$;
3. A candidate solution $X_i^{'}$ for the employed bee of the solution $X_i$ is generated by Equation (3);
4. Evaluate the fitness of $X_i^{'}$;
5. If the fitness of $X_i^{'}$ is better than that of $X_i$, let $X_i = X_i^{'}$ and $trial$=0,otherwise set $trial$= $trial$+1 ;
6. End If
7. End For
```

Fig. 2. Pseudo-code of employed bees phase.

$$X_{i,j}^{'} = x_j^{\min} + rand(0,1)\left(x_j^{\max} - x_j^{\min}\right) \tag{5}$$

where i is the index of the employed bees whose "*trail*" value reaches the "*limit*" value firstly, $j \in \{1, 2, \ldots, D\}$, $x_j^{\min}$ and $x_j^{\max}$ are the lower bound and the upper bound of the j-th variable respectively, and $rand$(0,1) is a random number between [0,1].

The pseudo-code of Scout Bees Phase of ABC algorithm is described as Fig. 4 [52].

2.2. IABC algorithm

In 2012, Gao et al. [25] have proposed an improved ABC algorithm, named IABC, in which the search way of employed bees is changed to the following equation [25]:

$$X_{i,j}^{'} = X_{best,j} + \varphi_{i,j}\left(X_{best,j} - X_{k,j}\right) \tag{6}$$

where $k \in \{1, 2, \ldots, SN/2\}$ and $j \in \{1, 2, \ldots, D\}$ are randomly chosen indexes; k has to be different from i; D is the number of problem dimension; $\varphi_{i,j}$ is a random number between [-1,1] and X_{best} is the global optimal solution.

Through introducing the above Equation (6) to change the updating method of the candidate food position $X_i^{'}$ in Employed Bees Phase, IABC algorithm has fast convergence ability. At the same time, Equation (3) is still employed in Onlooker Bees Phase, and thus the IABC algorithm can also keep its strong global-search capability. According to the

```
1. Calculate the selection probabilities $P_i$ ($i$=1, 2, ..., $SN/2$) for all the solutions by Equation (4);
2. Assume that $t$ is the index of the current onlooker bee, $i$ is the index of the current food source. We set $t$=1and $i$=1;
3. If $rnd < P_i$ (here $rnd$ is an uniformly distributed random number between 0 and 1), then $t$=$t$+1 and go to next step; otherwise go to Step 9;
4. Randomly select one dimension $j$ of the current food source and one neighbor of the current food source (assume that $k$ is the index of that neighbor, which is not equal to $i$), respectively, where $k \in \{1, 2, ..., SN/2\}$ and $j \in \{1, 2, ..., D\}$;
5. Generate a candidate solution $X_i^{'}$ for the onlooker bee of the solution $X_i$ by Equation (3);
6. If the fitness of $X_i^{'}$ is better than that of $X_i$, let $X_i = X_i^{'}$ and $trial$=0,otherwise $trial$= $trial$+1 ;
7. End If
8. End If
9. Set $i$=$i$+1;
10. If $i$ =$SN$/2+1, then set $i$ = 1;
11. If $t$ = $SN$/2+1, terminate the iteration; otherwise, go to Step 3.
```

Fig. 3. Pseudo-code of onlooker bees phase.

```
1. Assume that $i$ is the index of the current employed bee, $trial_i$ ($i$=1,2,..., $SN/2$) is the "$trail$" value of the $i$-th employed bee, and max($trial$) is the maximal value of all the "$trail$" values;
2. If max($trial$)>$limit$, then
3. Replace $X_i$ with a new randomly generated solution $X_i^{'}$ by Equation (5);
4. End if
```

Fig. 4. Pseudo-code of scout bees phase.

```
1. Initialize a solution $X=(x_1,x_2,\ldots,x_D)$ randomly and set $X_{best}=X$ and
   $OBJ(X_{best})=OBJ(X)$, where $X_{best}$ is the global best solution found so far and
   $OBJ(X_{best})$ is the objective function value of $X_{best}$;
2. For the current solution $X$,
   (a) Evaluate the local fitness $\lambda_i$ for each variable $x_i$ , $i=1, 2, \ldots, D$,
   (b) Perform ranking on all the variables depending on their local fitness and
       find the variable $x_j$ with the worst local fitness, i.e., $\lambda_j \le \lambda_i$ for all $i$,
   (c) Produce a new solution $X'$ through changing the state of $x_j$ according to a
       certain mutation rule,
   (d) Accept $X=X'$ unconditionally,
   (e) If $OBJ(X) < OBJ(X_{best})$ , then set $X_{best}=X$ and $OBJ(X_{best})=OBJ(X)$;
3. If the terminal condition is satisfied, go to the next step. Otherwise, go to
   Step 2;
4. Output $X_{best}$ and $OBJ(X_{best})$.
```

Fig. 5. General framework of EO algorithm.

experimental results in Ref. [25], IABC is better than the standard ABC in terms of efficiency and robustness.

2.3. Extremal optimization (EO)

Unlike the GA's framework of population reproduction, there is only one individual during the evolutionary process of EO. Thus, there are only selection and mutation operators in EO. In general, EO algorithm [39] for continuous optimization problem is composed of the following parts, i.e., initialization of a random solution, performing local fitness evaluation on each variable of the current solution, generation of a new solution through mutating on the variable with the worst fitness, and replacing the current solution with the new one unconditionally. The general framework of EO algorithm for a minimization optimization problem is described as Fig. 5 [39].

2.4. ABC-EO and IABC-EO algorithms

In order to improve the efficiency and accuracy of the standard ABC, in our previous work [40], we have presented two improved versions of ABC-EO, called ABC-EO and IABC-EO, through introducing EO to ABC and IABC [25], respectively. The pseudo-code of ABC-EO and IABC-EO for a minimization optimization problem with D dimensions is described as Fig. 6 [40].

The EO procedure is shown as Fig. 7. As can be observed from Fig. 7, in order to find out the worst variable, a local fitness value should be assigned to each variable of a solution. The local fitness of each variable of a solution for an unconstrained minimization problem is defined as follows. For the i-th solution, the local fitness $l_{i,k}$ of the k-th variable is defined as the mutation cost, i.e., $OBJ(X'_{i,k})$-$OBJ(X_{best})$, where $X'_{i,k}$ is the new solution generated through mutating only on the k-th variable of the i-th solution and keeping all other variables unchanged, $OBJ(X'_{i,k})$ is the objective value of $X'_{i,k}$, and $OBJ(X_{best})$ is the objective value of X_{best}.

In ABC-EO and IABC-EO, the G-C mutation is adopted, which combines the coarse search and grained search perfectly. It should be pointed out that there is one defect in G-C mutation, i.e., we should determine when to exchange each other between Gaussian and Cauchy mutations.

According to our previous work [40], through introducing EO to ABC and IABC, respectively, the presented approaches can combine the exploration ability of ABC (or IABC) with the exploitation ability of EO. Experimental results indicate that the ABC-EO and IABC-EO approaches owns superior performance to the standard ABC or IABC.

3. The proposed approaches

Note that ABC has great global-search ability, while EO has strong local-search capability. In our previous work [40], we have proposed two hybrid algorithms, i.e., ABC-EO and IABC-EO, where EO is introduced to update all the solutions when the global optimum is not getting better for several iterations. However, this updating operation on all solutions through introducing EO procedure will result in high computational cost. Especially while solving those complex optimization problems with many local optima, EO procedure will be introduced to ABC with high frequency, and thus the computational complexity will increase greatly. Moreover, the exchanging between Gaussian and Cauchy mutations in ABC-EO and IABC-EO will prolong the searching process and thus the

```
1. Initialize food sources by Equation (1);
2. Apply fitness evaluation on all food sources and store the global best food
   source in the memory;
3. Set $iteration$ =1;
4. Employed Bees Phase. Different from the standard ABC, the search way of
   employed bees in ABC-EO and IABC-EO is changed according to Equation (3)
   (for ABC-EO algorithm) or Equation (6) (for IABC-EO algorithm);
5. Onlooker Bees Phase;
6. Scout Bees Phase;
7. Update the global best food source. If the global best solution $X_{best}$ keeps
   unchanged for "INV" iterations (the value of the parameter "INV" is predefined by
   users according to the complexity of problems, and the interested readers are
   referred to the literature [40] for more details), then each solution is changed
   through introducing the EO procedure (see Fig. 7). Otherwise, go to the next step.
8. If the terminal condition is not met, then set $iteration = iteration + 1$, and go to
   Step 4; otherwise, go to the next step;
9. Output the optimal solution and the optimal objective value.
```

Fig. 6. Pseudo-code of ABC-EO and IABC-EO algorithms.

```
1. For the current solution $X_i=(X_{i,1}, X_{i,2}, \ldots, X_{i,D})$, $i\in\{1, 2, ..., SN/2\}$;
2. Mutation on each variable of $X_i$ is performed one by one, while leaving other variables unchanged. Thus, $D$ new solutions $X'_{i,k}$ ($k$=1, 2, …, $D$) can be generated;
3. Perform evaluations on the local fitness $l_{i,k}=OBJ(X'_{i,k})-OBJ(X_{best})$ of each variable $X_{i,k}$ $k\in\{1, 2, ..., D\}$, and rank all the new variables $X'_{i,k}$ ($k$=1, 2, …, $D$) depending on their local fitness values;
4. Assume that the variable $X_{i,w}, w\in\{1, 2, ..., D\}$is the one with the worst local fitness, and then $X'_{i,w}$ is the new generated variable coming from $X_{i,w}$ by mutation.
5.If $OBJ(X'_{i,w})<OBJ(X_i)$, then set $X_i=X'_{i,w}$ and $OBJ(X_i)=OBJ(X'_{i,w})$;Otherwise, $X_i$ is unchanged;
6. If $X_i$ is better than the global best solution $X_{best}$, then $X_{best}$ can be replaced with $X_i$; otherwise, $X_{best}$ keeps fixed.
```

Fig. 7. Pseudo-code of EO procedure.

time complexity will increase greatly. In order to decrease the computational cost of ABC-EO and IABC-EO, in this paper, we present two improved ABC-EO algorithms, called ABC-EO II and IABC-EO II, where EO is introduced into ABC and IABC in different way from those in ABC-EO and IABC-EO.

3.1. The description of the proposed algorithms

The detailed steps of ABC-EO II and IABC-EO II for a minimization optimization problem with D dimensions are described as follows.

```
Input: A continuous optimization problem and the control parameters of ABC-EO II and IABC-EO II, including the size of population ($SN$), maximum number of iteration ($MaxIter$ for short), the initial value of the control parameter "limit" in Scout Bees Phase, parameters $T_0$ and $\beta$ in Boltzmann selection.
Output: The global best solution $X_{best}$ for the continuous optimization problem and the corresponding optimal objective function value $OBJ(X_{best})$.
Step 1: Randomly generate the initial food source positions with the size $SN/2$ by Equation (1). Set $iteration = 1$.
Step 2: Evaluate the fitness of each food source according to Equation (2) and store the global best food source as $X_{best}$.
Step 3: Employed Bees Phase (see Fig. 2), in which the search way of employed bees is changed according to Equation (3) (for ABC-EO II algorithm) or Equation (6) (for IABC-EO II algorithm).
Step 4: Onlooker Bees Phase (see Fig. 3), in which the selection probability $P_i$ for the $i$-th food source ($i = 1, 2, \ldots, SN/2$) is replaced with $PB_i$ calculated by Equation (8).
Step 5: Calculate the probability $PE$ of each food source chosen by EO according to Equation (10).
Step 6: For $i = 1$ to $SN/2$ do
    An uniformly distributed random number $rnd$ ($rnd\in[0,1]$) is generated.
    If $rnd < PE_i$, then
        EO procedure (see Fig. 7) is introduced to change the current solution $X_i$
        Go to Step 7.
    Else if $i = = SN/2$, then
        $i = 1$.
    End if
    End for
Step 7: Scout Bees Phase.
Step 8: Update the global best solution $X_{best}$ found so far.
Step 9: If the maximum number of iteration is reached, go to the next step; otherwise, set $iteration = iteration + 1$, and go to Step 3.
```

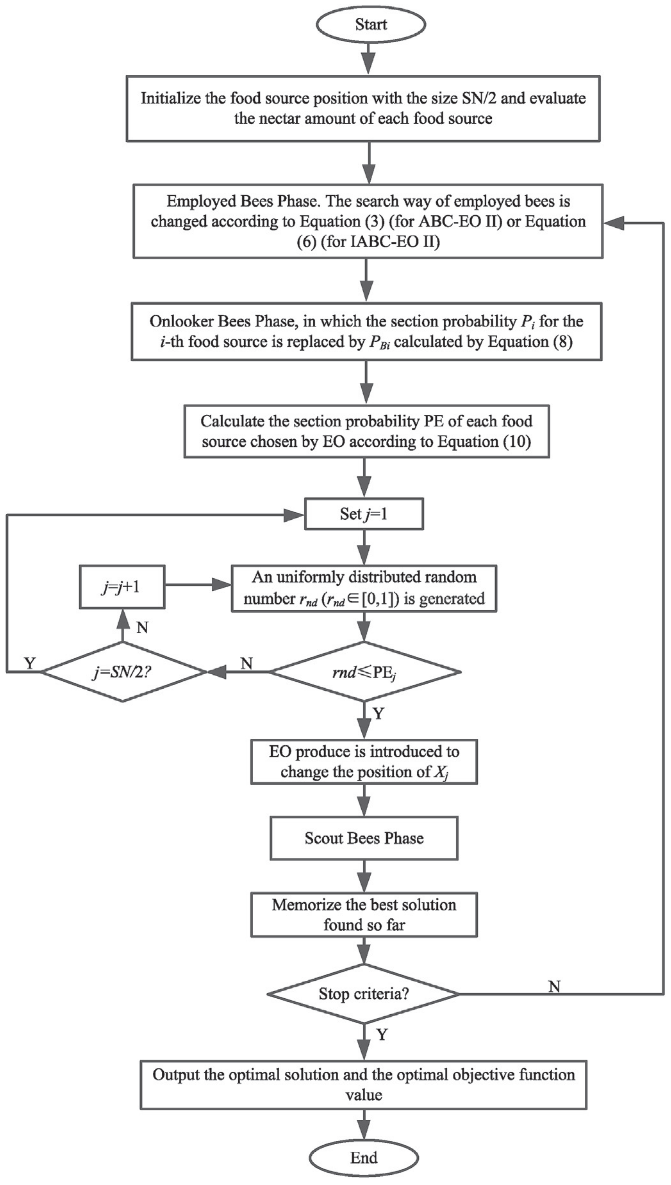

Start
Initialize the food source position with the size SN/2 and evaluate the nectar amount of each food source
Employed Bees Phase. The search way of employed bees is changed according to Equation (3) (for ABC-EO II) or Equation (6) (for IABC-EO II)
Onlooker Bees Phase, in which the section probability P_i for the i-th food source is replaced by P_{Bi} calculated by Equation (8)
Calculate the section probability PE of each food source chosen by EO according to Equation (10)
Set j=1
An uniformly distributed random number r_{nd} ($r_{nd}\in$[0,1]) is generated
j=j+1
N
Y
j=$SN/2$?
N
$rnd\leqslant$PE$_j$
Y
EO produce is introduced to change the position of X_j
Scout Bees Phase
Memorize the best solution found so far
Stop criteria?
N
Y
Output the optimal solution and the optimal objective function value
End

Fig. 8. The flowchart of ABC-EO II and IABC-EO II algorithms for continuous optimization problems.

Step 10: Output the global best solution X_{best} and the corresponding objective function value $OBJ(X_{\text{best}})$.

In the main procedure of ABC-EO II and IABC-EO II, the fitness of each solution is evaluated by Equation (2). However, in the EO procedure, in order to find out the worst variable, each variable of the chosen solution should be also assigned a local fitness value. It must be pointed out that the EO procedure in the proposed algorithms is different from that in ABC-EO and IABC-EO in terms of mutation operator. That is, the proposed approaches use the mutation operator as Equation (7), while ABC-EO and IABC-EO adopt Gaussian-Cauchy mutation operator. The flowchart of the proposed algorithms is shown in Fig. 8.

3.2. Main contributions in the proposed algorithms

There are four main contributions in our proposed algorithms as follows:

(1) Since there is only mutation operator in EO, the mutation plays a crucial role in EO procedure and it will influence the performance of EO significantly. In this work, we adopt the following expression as the mutation operator, which guides the searching process moving from the current solution X_i to its neighbor solution X_t.

$$X'_{i,k} = X_{i,k} + \varphi_{i,k}(X_{i,k} - X_{t,k}) \tag{7}$$

where i is the index of the solution chosen to be mutated, $k = 1, 2, \ldots, D$ and D is the number of problem dimension, $t \in \{1, 2, \ldots, SN/2\}$ is a randomly chosen index different from i, SN is the population size, and $\varphi_{i,k}$ is a random number between [-1,1].

As can be seen from Equation (7), due to the randomness of the neighbor solution chosen, this mutation operator can increase the diversity of new generated solutions, which helps our algorithms avoid getting trapped into local optima. Moreover, compared with G-C mutation in ABC-EO and IABC-EO, the mutation in our work doesn't need to decide when to exchange each other between Gaussian and Cauchy mutations, and thus this mutation is much easier to operate than those in ABC-EO and IABC-EO.

(2) In Onlooker Bees Phase, Boltzmann selection method [41] is introduced to choose the food source by the onlooker bees, where the selection pressure can be dynamically adjusted in the process of evolution. At each iteration, the proposed algorithms adopt the Boltzmann Selection method to choose only one food source updated by EO operation with certain probability.

In general, the larger nectar amount one food source has, the larger probability that food source is chosen by the onlooker bees. That is to say, a few food sources with the greatest nectar amount will attract most of onlooker bees. As we know, the selection pressure has a significant impact on the search abilities of an algorithm. Great selection pressure can strengthen the fast convergence ability of the algorithm. However, it will mislead the algorithm to be trapped into local optima. On the contrary, if the selection pressure is small, these poor food sources will have similar probability as those best ones to attract the onlooker bees, which is helpful to enhance the ability of global searching. However, too small selection pressure will result in slow convergence speed. Thus, an appropriate selection pressure plays an important role in the search abilities of an optimization algorithm.

In order to resolve the above contradiction, we introduce Boltzmann selection to the proposed algorithms in this paper, which can dynamically adjust selection pressure in the process of evolution. In our work, an artificial onlooker bee depends on a selection probability (denoted by PB) to choose one food source. The selection probability PB_i of the solution X_i is calculated as follows [41]:

$$PB_i = \frac{\exp(fit_i/T)}{\sum_{n=1}^{SN/2} \exp(fit_n/T)} \tag{8}$$

$$T = T_0(\beta^c) \tag{9}$$

where fit_i is the fitness value of the solution X_i, which is proportional to the nectar amount of the food source in the position X_i, $SN/2$ is the number of food sources which is equal to the number of employed bees or onlooker bees, T is the temperature, T_0 is the initial temperature which decides the initial selection pressure, β is an adjustable parameter which is used to control the changing speed of selection pressure and it should be set smaller than one, and c is the iteration number.

As can be observed from Equations (8) and (9), Boltzmann selection can dynamically adjust the selection pressure in different stages of evolution. In the early stage, the selection probability is so small that the selection pressure is also very small. This will strengthen the exploration ability of our proposed algorithms. As the iteration number increases, the selection probability goes up gradually, which causes the selection pressure to increase, and thus the exploitation capability of the proposed algorithms will be improved.

However, if β is set below 0.99, then the value of T will become very small as iteration number c goes up in the later stage. Thus, the value of fit_i/T will be very large, which will result in the computational cost of exp(fit_i/T) increasing sharply. Through some experiments, we find that the proper value of β can be set between 0.991 and 0.999, and T_0 can be set between 100 and 200, which will not cause too small selection probability and be able to control the changing speed of selection pressure.

(3) Through introducing EO to ABC at each iteration, it is not easy for ABC to get trapped into local optima. However, if EO is introduced to ABC for all solutions each iteration or every few iterations (just like ABC-EO or IABC-EO), the computational cost will increase sharply. In order to decrease the computational cost brought by integrating ABC with EO, in this work, EO only changes one solution at every iteration. Therefore, the computational costs of both proposed algorithms are much lower than those of ABC-EO and IABC-EO when solving complex hard optimization problems. For more details about the computational complexity of the above algorithms, readers can be referred to Section 3.4.

(4) In this work, we adopt the following selection probability which is related to Boltzmann selection to choose one solution mutated by EO:

$$PE_i = 1 - PB_i = 1 - \frac{\exp(fit_i/T)}{\sum_{n=1}^{SN/2} \exp(fit_n/T)} \tag{10}$$

where PB_i is calculated by Equation (8).

As can be seen from Equation (10), the smaller the fitness value of one solution is, the more chance it has to be chosen. This mechanism is more in line with the essence of EO which always chooses the worst or worse solutions to perform mutation. However, in ABC-EO and IABC-EO, all the solutions will be mutated by EO every few iterations. This is another advanced characteristic of our approaches different from ABC-EO and IABC-EO.

3.3. Differences between EABC, ABC-EO, IABC-EO and our proposed algorithms

Azadehgan et al. [53] have proposed a hybrid optimization method called EABC in 2011, in which EO is also introduced to ABC. In the EABC algorithm, EO procedure is used to determine how to choose the neighbor of employed bees or onlooker bees. However, this paper did not discuss the mechanism of EABC in detail and as can be seen from the

Table 1
Summary of experimental conditions.

Experiment	Test problem	Dimension	Algorithm	*SN*	*MaxIter*	Number of runs
Exp. 1	h_1-h_{10}	100	ABC-EO II IABC-EO II PEO [34] PSO-EO [39] RPEO-PLM [42]	30	6000 (ABC-EO II and IABC-EO II) 12000 (Others)	30
Exp. 2	f_1-f_{10}	30	ABC-EO II IABC-EO II GABC [43] PABC [44] IABC [25]	100	50000 (ABC-EO II and IABC-EO II) 100000 (Others)	30
Exp. 3	h_1-h_{10}	100	ABC-EO II IABC-EO II RCGA-ADM [45] IEA [46] DE [47] PSO [39]	30	6000 (ABC-EO II and IABC-EO II) 12000 (Others)	30
Exp. 4	F_1–F_6	10 (F_1) 30 (Others)	IABC-EO II IABC-EO [40] ABC-EO II ABC-EO [40] IABC [25] PSO-EO [39] GA [39] ABC [40]	10 (F_1, F_4) 30 (Others)	20000 (F_1– F_4) 10000 (F_5) 100000 (F_6)	30
Exp. 5	h_1-h_{10} and f_1-f_{10}	100 (h_1-h_{10}) and 30 (f_1-f_{10})	ABC-EO II IABC-EO II WDE [48] EDEV [49] IJAYA [50]	30 (h_1-h_{10}) and 100 (f_1-f_{10})	6000 (ABC-EO II and IABC-EO II for h_1-h_{10}) 12000 (Others for h_1-h_{10}) 50000 (ABC-EO II and IABC-EO II for f_1-f_{10}) 100000 (Others for f_1-f_{10})	30
Exp. 6	Real life problems (parameters identification of photovoltaic models)	5 (single diode model) 7 (double diode model)	ABC-EO II IABC-EO II IJAYA [50] PSO-DE [51] EDEV [49] WDE [48]	45 (ABC-EO II) and IABC-EO II) 20 (IJAYA) 40 (PSO-DE) 100(EDVE) 50 (WDE)	240 (ABC-EO II and IABC-EO II for single diode model) 230 (ABC-EO II and ABC-EO II for double diode model) 11080 (IJAYA) 300 (PSO-DE) 100 (EDEV) 50 (WDE)	30

experimental results [53], this algorithm is week in solving some continuous optimization problems. Therefore, we do not compare EABC with our proposed algorithms here and the interested readers may get more details of EABC from the literature [53].

Compared with ABC-EO and IABC-EO (so-called the old versions) in our previous work [40], there are the following differences between the proposed algorithms and the old versions:

(1) In the old versions, EO is introduced to update the positions of all food sources when the global optimal position is unchanged for several iterations. While in this work, EO is only introduced to change the position of only one food source at each iteration. Therefore, while dealing with those problems with large dimension, the computational cost of our proposed algorithms will be much smaller than those of the old versions.
(2) Moreover, the EO procedures in the old versions adopt G-C mutation, which needs to determine when to exchange each other between Gaussian and Cauchy mutations, while our proposed algorithms in this work use Equation (8) to perform mutation in EO procedure and thus it is much easier to operate than G-C mutation in the old versions.
(3) Furthermore, in the proposed algorithms, by means of introducing Boltzmann selection method, the selection pressure can be dynamically adjusted in the process of evolution, while the selection pressure cannot be tuned in the evolutionary procedure in the old versions.
(4) In our algorithms, only one solution will be chosen to be mutated by EO according to the selection probability related to Boltzmann selection. As can be seen from Equation (10), the smaller the fitness value of one solution is, the more chance it has to be selected by EO. This mechanism is more in line with the essence of EO which always performs mutation on the worst or worse solutions. However, in the old versions, EO puts mutation on all the solutions. This is another advanced characteristic of our algorithms different from the old versions.

3.4. *Computational complexity of our algorithms*

Procedures are the same in each iteration, so we only look at the computational complexity of one iteration of the entire algorithm. For the proposed ABC-EO II algorithm, the worst-case complexity of the basic operations is as follows:

- The complexity of the ABC procedure is $O(5SN/2)$, where SN is the population size.
- Because EO is introduced to change the position of one food source at each iteration, the complexity of EO procedure is $O(D)$, where D is the number of decision variables.

Thus, the overall complexity of ABC-EO II is $O(5SN/2 + D)$, while the whole complexity of the standard ABC algorithm is $O(5SN/2)$. Because we only introduce Equation (7) to replace Equation (4) in Employed Bees Phase of IABC-EO II, the overall complexity of IABC-EO II algorithm is also $O(5SN/2 + D)$. Obviously, the computational complexity of ABC-EO II and IABC-EO II is mostly influenced by the problem dimension D and the population size SN. As we can see, when EO is introduced to ABC at each iteration, the computational complexity of ABC–EO II or IABC-EO II is a little higher than that of the standard ABC. When the number of decision variables (i.e., D) is small, our proposed approaches in this paper will have almost the same complexity as that of the standard ABC, due to the low computational complexity of EO procedure (i.e., $O(D)$).

According to the pseudo-code of ABC-EO and IABC-EO [40], it can be found that the computational complexities of ABC-EO and IABC-EO are both $O\ (5SN/2 + D \times SN/2INV)$. Here, INV represents for the number of iterations for which the global best solution X_{best} keeps unchanged in ABC-EO or IABC-EO (please see Fig. 6, where INV appears for the first time). If the problem to be optimized is easy to find the optima, then the EO procedure can be introduced to the algorithm with low frequency (i.e., the value of INV is large). In this case, $SN/2$ will be smaller than the value of INV, and thus the computational complexities of ABC-EO and IABC-EO are almost the same as those of our approaches in this work. Otherwise, if the problem to be solved is hard to obtain the optima, then the EO procedure should be included to the algorithm with high frequency (i.e., the value of INV is small). Under this situation, $SN/2$ will be larger than the value of INV, and therefore the computational complexities of ABC-EO and IABC-EO are larger than those of our approaches when dealing with complex hard optimization problems.

From the above analysis, it can be concluded that, compared with the standard ABC, ABC-EO and IABC-EO, our algorithms have nearly the same or lower computational complexity, and therefore our approaches in this work are especially suitable for handling those complex hard optimization problems.

4. Experimental results and discussion

4.1. Summary of experimental conditions

This section aims to investigate the effectiveness of the proposed algorithms, i.e., ABC-EO II and IABC-EO II, through six groups of experiments shown as Table 1. In the first experiment, i.e., Exp. 1, our proposed algorithms are compared with three EO variants, i.e., PEO [34], PSO-EO [39] and RPEO-PLM [42]. In the second experiment, i.e., Exp. 2, our methods are used to compete with three ABC variants, i.e., GABC [43], PABC [44], and IABC [25]. In the third experiment, i.e., Exp. 3, comparisons are performed among our algorithms and four well-recognized evolutionary algorithms, i.e., RCGA-ADM [45], IEA [46], DE [47], and PSO [39]. To further demonstrate the performance of our algorithms, the fourth experiment, i.e., Exp. 4 is designed to compare the performance of our algorithms with six state-of-the-art evolutionary algorithms, i.e., IABC-EO [40], ABC-EO [40], IABC [25], PSO-EO [39], GA [39], and ABC [40]. Besides, the three recently published state-of-the-art EAs including WDE [48], EDEV [49], and IJAYA [50] are considered as the competitors for the fifth experiment, i.e., Exp. 5. In all the following experiments, the parameters β and T_0 used in our algorithms are set as 0.995 and 200, respectively. To compare the performance of all the algorithms fairly, all the algorithms are performed under the same fitness function evaluations (FFEs for short), except the FFEs of proposed algorithms are set smaller than those of compared algorithms in some experiments to highlight their superiority. Table 2, Table 3 and Table 4 show three groups of well-known benchmark functions chosen from the literature [40–46]. Note that the benchmark functions in Exp. 4 come from the literature [40] published in CEC 2014. All the problems are to be minimized. Besides, to testify the feasibilities of our algorithms for the real world scenario, two kinds of parameters identification of photovoltaic models are adopted as the sixth experiment, i.e., Exp. 6 in Section 4.6. All the experiments are performed in JAVA software on a 2.20 GHz computer with processor i5-5200U and 8 GB RAM, except the

Table 2
Benchmark functions h_1-h_{10} with the dimension $D = 100$.

Problem	Function expression	Search space	Global minimum
h_1	$h_1 = \sum_{i=1}^{D} \left\lvert \frac{\sin(10x_i\pi)}{10x_i\pi} \right\rvert$	(-0.5,0.5)	0
h_2	$h_2 = \sum_{i=1}^{D} [x_i + 0.5]^2$	(-100,100)	0
h_3	$h_3 = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	(-5.12,5.12)	0
h_4	$h_4 = \sum_{i=1}^{D} x_i^2$	(-5.12,5.12)	0
h_5	$h_5 = -\sum_{i=1}^{D} [x_i \sin(10\pi x_i)]$	(-1,2)	≈-1.85n
h_6	$h_6 = \sum_{i=1}^{D} \left[\sin(x_i) + \sin\left(\frac{2x_i}{3}\right)\right]$	(3,13)	≈-1.21598n
h_7	$h_7 = -20\exp\left(-0.02\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	(-30,30)	0
h_8	$h_8 = 418.9828D - \sum_{i=1}^{D} x_i \sin(\sqrt{\lvert x_i \rvert})$	(-500,500)	0
h_9	$h_9 = 6D + \sum_{i=1}^{D} \lfloor x_i \rfloor$	(-5.12,5.12)	0
h_{10}	$h_{10} = 1 + \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	(-600,600)	0

Table 3
Benchmark functions F_1–F_6.

Problem	Function expression	Search space	Global minimum
F_1	$F_1 = -\sum_{i=1}^{D} \sin(x_i)\sin^{2m}\left\{\frac{(i+1)x_i^2}{\pi}\right\}, m = 10$	(0π)	-9.66
F_2	$F_2 = -\sum_{i=1}^{D} (x_i \sin(\sqrt{\lvert x_i \rvert}))$	(-500,500)	$-12{,}569.5$
F_3	$F_3 = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	(-600,600)	0
F_4	$F_4 = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	(-5.12,5.12)	0
F_5	$F_5 = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\sum_{i=1}^{D}\frac{\cos(2\pi x_i)}{D}\right)$	(-32.768,32.768)	0
F_6	$F_6 = \sum_{i=1}^{D} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	(-30,30)	0

Table 4
Benchmark functions f_1-f_{10} with the dimension $D = 30$.

Problem	Function expression	Search space	Global minimum
f_1	$f_1 = \sum_{i=1}^{D} x_i^2$	(-100,100)	0
f_2	$f_2 = \sum_{i=1}^{D} \lvert x_i \rvert^{(i+1)}$	(-1,1)	0
f_3	$f_3 = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	(-10,10)	0
f_4	$f_4 = \max\{\lvert x_i \rvert, 1 \le i \le D\}$	(-100,100)	0
f_5	$f_5 = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	(-100,100)	0
f_6	$f_6 = \exp(0.5 * \sum_{i=1}^{D} x_i) - 1$	(-1.28,1.28)	0
f_7	$f_7 = \sum_{i=1}^{D} i x_i^4 + random[0, 1)$	(-1.28,1.28)	0
f_8	$f_8 = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	(-5,10)	0
f_9	$f_9 = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\sum_{i=1}^{D}\frac{\cos(2\pi x_i)}{D}\right)$	(-32,32)	0
f_{10}	$f_{10} = \sum_{i=1}^{D} \lvert x_i \sin(x_i) + 0.1x_i \rvert$	(-10,10)	0

Table 5
Comparative performances for test functions h1-h10 with the dimension D = 100.

Problem	ABC-EO II		IABC-EO II		PEO [34]		PSO-EO [39]		RPEO-PLM [42]	
	fm	rank	*fm*	rank	*fm*	rank	*fm*	rank	*fm*	rank
h_1	**0.000E + 0**	1	**0.000E + 0**	1	2.618E-3	5	**0.000E + 0**	1	1.890E-6	4
h_2	**0.000E + 0**	1	**0.000E + 0**	1	1.621E-6	4	**0.000E + 0**	1	1.661E-5	5
h_3	**0.000E + 0**	1	4.112E-7	3	12.61	5	**0.000E + 0**	1	2.761E-6	4
h_4	**0.000E + 0**	1	**0.000E + 0**	1	3.603E-10	4	2.153E-11	3	1.392E-8	5
h_5	**−1.850E + 2**	1	**−1.850E + 2**	1	−1.423E+2	5	−1.631E+2	4	**−1.850E + 2**	1
h_6	**−1.216E + 2**	1	**−1.216E + 2**	1	−1.201E+2	5	**−1.216E + 2**	1	**−1.216E + 2**	1
h_7	**1.048E-9**	1	1.384E-6	3	20.41	5	5.028E-5	4	6.985E-8	2
h_8	7.904E+0	3	**6.607E-9**	1	7.520E+4	5	8.624E+0	4	1.380E-3	2
h_9	**0.000E + 0**	1	**0.000E + 0**	1	3.020E+0	5	**0.000E + 0**	1	**0.000E + 0**	1
h_{10}	**0.000E + 0**	1	6.572E-4	4	**0.000E + 0**	1	9.272E-4	5	5.005E-6	3
Average rank	**1.2**		1.5		4.4		2.5		2.8	
Final rank	1		2		5		3		4	

Table 6
Ranks, the statistics, and related *p*-value achieved by Friedman and Quade tests for Exp. 1

Algorithm	Friedman test	Quade test
ABC-EO II	**2**	**2.0636**
IABC-EO II	2.35	2.2909
PEO [34]	4.5	4.7818
PSO-EO [39]	3.05	3.2
RPEO-PLM [42]	3.1	2.6636
Statistic	14.74	5.283064
p-value	0.005272	0.001877

Table 8
Ranks, the statistics, and related *p*-value achieved by Friedman and Quade tests for Exp. 2

Algorithm	Friedman	Quade
GABC [43]	4.25	4.5727
PABC [44]	3.2	3.5545
IABC [25]	2.95	2.6545
ABC-EO II	2.7	2.6636
IABC-EO II	**1.9**	**1.5545**
Statistic	11.62	6.023413
p-value	0.020412	0.000811399174

WDE [48], PSO-DE [51], EDVE [49], and IJAYA [50] which are performed in MATLAB software on a 2.5 GHz PC with processor I7-6500 and 8 GB RAM.

4.2. Exp. 1: comparison with PEO, PSO-EO and RPEO-PLM

In recent years, several EO-based evolutionary algorithms, e.g., PEO [34], PSO-EO [39] and RPEO-PLM [42] have been presented and testified their effectiveness by a large number of experiments. In order to further investigate the superiority of our proposed algorithms to PEO, PSO-EO and RPEO-PLM, Exp. 1 is carried out. The experimental results of these five algorithms, i.e., ABC-EO II, IABC-EO II, PEO, PSO-EO and RPEO-PLM for 10 benchmark test problems are shown in Table 5. Note that *fm* represents the average results. The experimental results of algorithms RPEO-PLM is excerpted from the literature [42]. Table 5 indicates that our algorithms, i.e., ABC-EO II and IABC-EO II, perform the best in 6 out of 10 test functions h_1-h_{10}, the same as PSO-EO for functions h_1, h_2, h_6 and h_9, and the same as RPEO-PLM for function h_5, h_6 and h_9. Therefore, compared with other three algorithms, ABC-EO II and IABC-EO II have almost the same or superior performance for test functions h_1-h_{10}.

Additionally, to detect the statistical differences systematically in Exp. 1, Friedman and Quade tests [54,55] are carried out by making use of keel software [56]. Ranks, the statistics and *p*-value computed by the Friedman and Quade tests for all the algorithms in Exp. 1 are shown in Table 6. Clearly, ABC-EO II achieves the best rank both in Friedman and Quade test, while IABC-EO II is the second-best one with a level of significance $\alpha = 0.01$.

4.3. Exp. 2: comparison with GABC, PABC and IABC

Recently, some ABC variants, e.g., GABC [43], PABC [44], and IABC [25] have been proposed and demonstrated by experimental results on a large number of experiments. In Exp. 2, our proposed algorithms are compared with these three ABC-based algorithms, i.e., GABC, PABC and IABC on 10 well-known benchmark functions f_1-f_{10}. Specifically speaking, f_1-f_4 and f_6 are continuous unimodal functions, f_5 is a discontinuous step function and f_7 is a noisy quartic function. The Rosenbrock function f_8 with high dimension has multiple minima. f_9 and f_{10} are multimodal functions and as the dimension increases, the number of the local minima will also increase exponentially.

The experimental results of these algorithms on test functions f_1-f_{10} are shown in Table 7. Note that *fm* represents the average results, and *SD* means the standard deviation. The experimental results of GABC and PABC are excerpted from the literature [44]. As can be seen in Table 7,

Table 7
Comparative performances for test functions f1-f10 with the dimension D = 30.

Problem	GABC [43]		PABC [44]		IABC [25]		ABC-EO II		IABC-EO II	
	fm	SD	*fm*	SD	*fm*	SD	*fm*	SD	*fm*	SD
f_1	**0**	**0**	**0**	**0**	1.41E-15	**0**	**0**	**0**	**0**	**0**
f_2	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_3	3.23E-12	9.27E-13	**0**	**0**	1.27E-15	**0**	**0**	**0**	**0**	**0**
f_4	7.00E-00	1.01E-00	9.17E-00	1.38E-01	6.74E-00	1.01E-00	2.99E-00	1.41E-01	**1.68E-00**	**1.25E-01**
f_5	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_6	7.35E-09	3.29E-08	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_7	8.94E-02	2.42E-02	4.68E-02	9.42E-03	8.90E-4	2.35E-04	4.21E-02	8.42E-03	**4.47E-04**	**1.53E-04**
f_8	1.15E-00	2.81E-00	6.69E-02	1.28E-01	7.32E-13	2.45E-12	1.89E-03	1.46E-03	**0**	**0**
f_9	2.15E-11	1.06E-11	1.94E-14	**4.81E-15**	3.95E-12	5.08E-13	2.08E-14	**4.06E-15**	2.95E-14	9.15E-15
f_{10}	5.99E-06	9.71E-06	**3.61E-14**	**4.28E-14**	**1.72E-15**	**1.54E-15**	3.40E-14	4.83E-14	6.17E-15	3.15E-15

Table 9
Comparative performances for test functions h_1-h_{10} with the dimension D = 100.

problem	ABC-EO II		IABC-EO II		RCGA-ADM [45]		IEA [46]		DE [47]		PSO [39]	
	fm	rank	*fm*	rank	*fm*	rank	*fm*	rank	*fm*	rank	*fm*	rank
h_1	**0.000E + 0**	**1**	**0.000E + 0**	**1**	1.490E-5	3	6.500E-1	5	1.366E-0	6	5.548E-1	4
h_2	**0.000E + 0**	**1**	**0.000E + 0**	**1**	3.333E-2	5	6.210E+2	6	9.703E-8	4	**0.000E + 0**	**1**
h_3	**0.000E + 0**	**1**	4.112E-7	3	7.988E+1	4	2.315E+2	6	2.123E+2	5	**0.000E + 0**	**1**
h_4	**0.000E + 0**	**1**	**0.000E + 0**	**1**	**0.000E + 0**	**1**	1.600E+0	6	9.546E-8	4	7.130E-4	5
h_5	**−1.850E + 2**	**1**	**−1.850E + 2**	**1**	−1.325E+2	4	−1.313E+2	5	−1.486E+2	3	−1.301E+2	6
h_6	**−1.216E + 2**	**1**	**−1.216E + 2**	**1**	**−1.216E + 2**	**1**	−1.204E+2	5	−1.199E+2	6	**−1.216E + 2**	**1**
h_7	**1.048E-9**	**1**	1.384E-6	3	3.239E+0	5	3.690E+0	6	1.492E-0	4	4.138E-9	2
h_8	7.904E+0	2	**6.607E-9**	**1**	1.463E+4	6	8.011E+3	4	1.394E+4	5	2.104E+3	3
h_9	**0.000E + 0**	**1**	**0.000E + 0**	**1**	5.647E+1	4	4.394E+1	3	2.270E+2	6	6.513E+1	5
h_{10}	**0.000E + 0**	**1**	6.572E-4	3	2.629E-3	4	3.286E+1	6	4.728E-2	5	3.562E-6	2
Average rank	**1.1**		1.6		3.7		5.2		4.8		3.0	
Final rank	1		2		4		6		5		3	

ABC-EO II and IABC-EO II perform better than other algorithms on function f_4 and the same as other algorithms on functions f_1, f_2 and f_5. Moreover, IABC-EO II ranks the first on two functions, i.e., f_7 and f_8, and our algorithms obtained almost the same performance as PABC and IABC on functions f_9 and f_{10}. Thus, it can be concluded that our proposed algorithms own nearly the same or better rank than other algorithms on functions f_1-f_{10}, and our proposed algorithms may be good alternatives for solving those continuous/discontinuous, unimodal/multimodal or noisy quartic functions with high dimension or multiple local optima.

Additionally, ranks, the statistics and *p*-value computed by the Friedman and Quade tests for the algorithms in Exp. 2 are given in Table 8. It is obvious that IABC-EO II ranks the first both in Friedman and Quade tests, while ABC-EO II is almost the second-best one with a level of significance $\alpha = 0.05$.

4.4. Exp. 3: comparison with RCGA-ADM, IEA, DE, and PSO

In Exp. 3, the superiority of the proposed algorithms is compared with three well-known and successful algorithms, i.e., RCGA-ADM [45], IEA [46], DE [47] and PSO [39] which have been demonstrated by the experimental results on a large number of benchmark functions. The comparative performances of these algorithms for test functions h_1-h_{10} are shown in Table 9. Note that *fm* represents the average results. The experimental data of RCGA-ADM and IEA are excerpted from the literature [42]. As can be found in Table 9, our algorithms, i.e., ABC-EO II and IABC-EO II, perform better than RCGA-ADM, IEA and PSO in 5 out of 10 test functions h_1-h_{10} respectively, the same as RAGA-ADM for 2 functions and the same as PSO for two functions h_2 and h_6. Besides, ABC-EO II outperforms IABC-EO II in 3 out of 10 test functions and IABC-EO II only performs better than ABC-EO II in function h_8. Hence, compared with other algorithms, our proposed algorithms, i.e., ABC-EO II and IABC-EO II, obtain nearly the same or superior performance for test functions h_1-h_{10}.

Furthermore, the results of ranks, the statistics and *p*-value computed by the Friedman and Quade tests for the algorithms in Exp. 3 are shown in Table 10. Obviously, ABC-EO II achieves the best rank both in Friedman and Quade test, while IABC-EO II is the second-best one with a level of significance $\alpha = 0.01$.

Table 10
Ranks, the statistics, and related *p*-value achieved by Friedman and Quade tests for Exp. 3

Algorithm	Friedman	Quade
RCGA-ADM [45]	3.95	4.4273
IEA [46]	5.2	5.0727
DE [47]	4.8	4.7091
PSO [39]	3.25	3.0545
ABC-EO II	**1.7**	**1.7091**
IABC-EO II	2.1	2.0273
Statistic	28.7	10.270901
p-value	0.000027	0.000001300349

Table 11
Comparison results for function F_1 ($F_1{}^* = -9.66$).

Algorithm	Success (%)	Time (ms)	*fm*	*fb*	*fw*	*SD*
IABC-EO II	**100**	264.76	**−9.660**	**−9.660**	**−9.659**	9.476E-5
IABC-EO [40]	**100**	181.00	**−9.660**	**−9.660**	**−9.659**	8.657E-5
ABC-EO II	**100**	270.86	**−9.660**	**−9.660**	**−9.659**	7.659E-5
ABC-EO [40]	**100**	**135.3**	**−9.660**	**−9.660**	**−9.659**	9.061E-5
IABC [25]	**100**	156.03	**−9.660**	**−9.660**	**−9.659**	9.536E-5
PSO-EO [39]	**100**	7452.8	**−9.660**	**−9.660**	**−9.659**	**7.561E-5**
GA [39]	43.3	524.64	−9.617	−9.659	−9.45	0.08
ABC [40]	**100**	161.00	**−9.660**	**−9.660**	**−9.659**	7.642E-5

4.5. Exp. 4: comparison with IABC-EO, ABC-EO, IABC, PSO-EO, GA and ABC

In order to compare the computational efficiency and convergence speed of IABC-EO II and ABC-EO II with other algorithms, Exp. 4 is designed in this section. Different from Exp. 1, Exp. 2 and Exp. 3 which will stop running when the maximum number of iteration is reached, all algorithms in Exp. 4 will stop if the solutions found by them reach the near-global minimum, i.e., *F* satisfies that $|(F^*\text{-}F)/F^*|<10^{-5}$ (for the case $F^*\neq 0$) or $|F^*\text{-}F|<10^{-5}$ (for the case $F^* = 0$), or the maximum number of iteration is reached. Here, *F* indicates the results found by algorithms and F^* represents global minimum of the functions. In Tables 11–16, "Success" indicates the successful rate, and "Time" is the average runtime of 30 independent runs. Meanwhile, the best results *fb*, the average results *fm*, the worst results *fw* and the standard deviations (*SD* for short) found by each algorithm are also listed in these tables.

As can be seen from Table 11, for function F_1 which is highly multimodal, the successful rate of IABC-EO II, ABC-EO II, IABC-EO [40], ABC-EO [40], IABC [25], PSO-EO [39], and ABC [40] are 100%. It is clearly that each algorithm except GA has nearly the same performance in terms of stability. ABC-EO has the fastest speed to find the global minimum while IABC-EO, IABC and ABC converge to the global minimum almost as quickly as ABC-EO.

From Table 12, which shows the simulation results of each algorithm on function F_2, it can be seen that IABC-EO could find the global minimum with the 100% successful rate and the fastest convergence speed, and ABC-EO II and IABC converge to the global minimum almost as quickly as IABC-EO. Besides, IABC-EO II and ABC-EO are a little slower than IABC-EO with respect to convergence speed, but better than PSO-EO

Table 12
Comparison results for function F_2 ($F_2{}^* = -12569.5$).

Algorithm	Success (%)	Time (ms)	*fm*	*fb*	*fw*	*SD*
IABC-EO II	**100**	106.63	**−12569.441**	**−12569.500**	**−12569.379**	0.033
IABC-EO [40]	**100**	**82.3**	**−12569.441**	**−12569.500**	−12569.377	0.037
ABC-EO II	**100**	82.93	−12569.435	**−12569.500**	−12569.376	0.037
ABC-EO [40]	**100**	148.36	−12569.431	**−12569.500**	−12569.378	0.033
IABC [25]	**100**	87.86	−12569.432	**−12569.500**	−12569.375	**0.029**
PSO-EO [39]	68	80642.36	−12531.543	−12569.471	−12451.057	54.188
GA [39]	0	2531.63	−8881.299	−9904.510	−8088.527	374.539
ABC [40]	**100**	180.86	−12569.434	**−12569.500**	−12569.375	0.034

Table 13
Comparison results for function F_3 ($F_3{}^* = 0$).

Algorithm	Success (%)	Time (ms)	*fm*	*fb*	*fw*	*SD*
IABC-EO II	**100**	28.00	9.341E-6	7.125E-6	9.962E-6	**5.891E-7**
IABC-EO [40]	**100**	28.50	8.142E-6	4.322E-6	**9.955E-6**	1.679E-6
ABC-EO II	**100**	32.16	8.782E-6	5.184E-6	9.971E-6	1.233E-6
ABC-EO [40]	**100**	45.06	7.441E-6	**1.709E-6**	9.998E-6	2.101E-6
IABC [25]	**100**	**24.03**	7.881E-6	2.748E-6	9.996E-6	1.807E-6
PSO-EO [39]	**100**	40.20	**3.812E-6**	2.125E-6	9.976E-6	1.923E-6
GA [39]	0	2849.32	0.060	0.005436	0.201	0.036
ABC [40]	**100**	39.16	7.905E-6	2.443E-6	9.988E-6	1.912E-6

Table 14
Comparison results for function F_4 ($F_4{}^* = 0$).

Algorithm	Success (%)	Time (ms)	*fm*	*fb*	*fw*	*SD*
IABC-EO II	**100**	**70.40**	7.206E-6	5.217E-7	9.959E-6	2.914E-6
IABC-EO [40]	**100**	103.40	6.485E-6	9.537E-7	9.881E-6	2.574E-6
ABC-EO II	**100**	71.70	7.112E-6	1.011E-7	9.986E-6	3.154E-6
ABC-EO [40]	**100**	123.93	6.528E-6	4.666E-7	9.942E-6	2.786E-6
IABC [25]	**100**	98.30	6.529E-6	1.913E-7	9.991E-6	2.936E-6
PSO-EO [39]	**100**	94.633	**0**	**0**	**0**	**0**
GA [39]	0	293.333	0.023	7.333E-5	0.569	0.075
ABC [40]	100	123.60	6.716E-6	1.016E-6	9.974E-6	2.704E-6

Table 15
Comparison results for function F_5 ($F_5{}^* = 0$).

Algorithm	Success (%)	Time (ms)	*fm*	*fb*	*fw*	*SD*
IABC-EO II	**100**	70.50	7.788E-6	7.508E-7	9.999E-6	2.654E-6
IABC-EO [40]	**100**	56.80	8.186E-6	2.559E-6	9.609E-6	1.485E-6
ABC-EO II	**100**	89.70	9.202E-6	5.917E-6	9.998E-6	9.755E-7
ABC-EO [40]	**100**	84.20	8.005E-6	2.851E-6	9.983E-6	1.993E-6
IABC [25]	**100**	49.40	8.598E-6	4.711E-6	9.949E-6	1.431E-6
PSO-EO [39]	**100**	**29.36**	**8.88E-16**	**8.88E-16**	**8.88E-16**	**9.96E-32**
GA [39]	0	734.97	0.051	0.023	0.116	0.019
ABC [40]	100	78.35	8.304E-6	4.395E-6	9.961E-6	1.776E-6

in terms of successful rate, convergence speed and solution accuracy.

Table 13 shows the simulation results of all these algorithms on functions F_3. As can be observed from Table 13, each algorithm except GA could find the optimal solution with 100% successful rate. IABC can find the minimum in the shortest time and IABC-EO II, ABC-EO II, IABC-EO and ABC-EO has nearly the same convergence speed with IABC.

From Table 14, which gives the experimental results of all the algorithms on function F_4, it can be seen that IABC-EO II and ABC-EO II converge to the optima more quickly than other algorithms, and PSO-EO is the best performer in terms of solution accuracy.

From Table 15 on function F_5, it is obvious that PSO-EO is the best performer with respect to convergence speed and solution accuracy. All the algorithms except GA can find the optima with 100% successful rate.

It must be pointed out that the global minimum point of function F_6 is inside a long, narrow, parabolic shaped flat valley. Even though this valley is easy to find, it's difficult to approach to the minimum. As can be seen from Table 16, IABC-EO II is the winner which could find the global minimum with 100% successful rate, the fastest convergence speed and best solution quality. Moreover, PSO-EO can find the optimum with 100% successful rate but need more running time than IABC-EO II.

In general, the above simulation results confirm that the proposed algorithms, i.e., ABC-EO II and IABC-EO II perform nearly as well as or superiorly in terms of successful rate, convergence speed and solution quality, as compared to IABC, IABC-EO, ABC-EO, PSO-EO, GA and ABC. Therefore, our algorithms can be considered as good performers for solving complex continuous optimization problems.

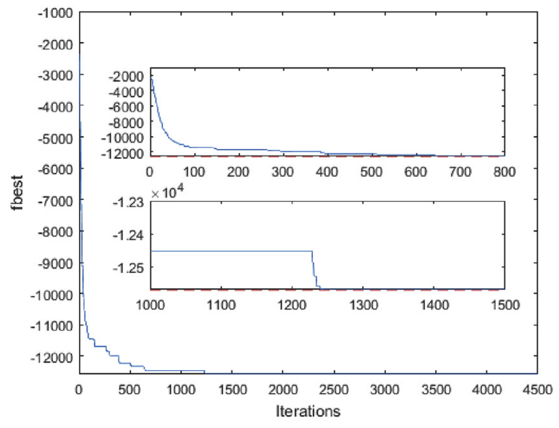
Furthermore, the convergence curves of the ABC-EO II, IABC-EO II, ABC, ABC-EO and IABC-EO for two benchmark functions F_2 and F_4 are illustrated as Fig. 9 and Fig. 10. From Fig. 9, we can see that ABC-EO II, IABC-EO II and IABC-EO converge to the near-global minimum of test function F_2 at about 1000 iterations, while ABC and ABC-EO converge to the near-global minimum at approximate 2000 iterations. Thus, for function F_2, our proposed algorithms perform nearly as well as IABC-EO and better than ABC and ABC-EO in terms of convergence speed. It can be found from Fig. 10 that ABC-EO II and IABC-EO II are able to converge to the global minimum at about 600 iterations and 960 iterations respectively, while ABC, ABC-EO and IABC-EO find the global minimum at more than 6300 iterations. Therefore, for function F_4, our proposed algorithms have the fastest convergence speed.

4.6. Exp. 5: comparison with WDE, EDEV and IJAYA

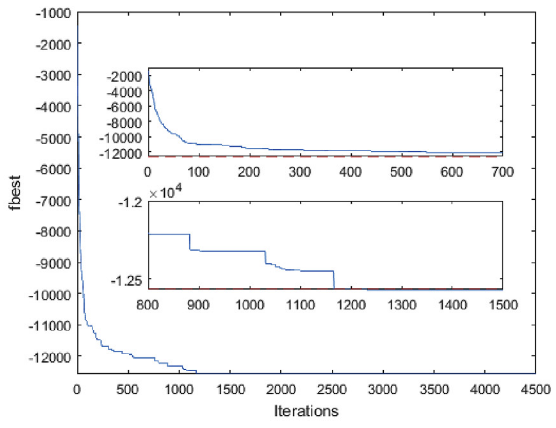
In the former experiments, we have demonstrated the performance of IABC-EO II and ABC-EO II by comparison with some popular evolutionary algorithms. This subsection is devoted to further investigating the

Table 16
Comparison results for function F_6 ($F_6{}^* = 0$).

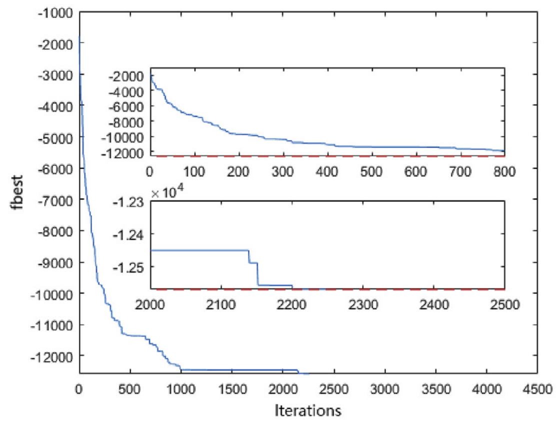
Algorithm	Success (%)	Time (ms)	*fm*	*fb*	*fw*	*SD*
IABC-EO II	**100**	**285.33**	**8.916E-6**	4.131E-6	**9.997E-6**	**1.251E-6**
IABC-EO [40]	90.00	833.36	3.677E-5	**1.602E-6**	7.723E-4	1.371E-4
ABC-EO II	0	1350.18	0.009	1.418E-4	0.066	0.012
ABC-EO [40]	0	1604.23	0.011	7.624E-4	0.052	0.011
IABC [25]	86.66	558.20	1.298E-4	6.554E-6	0.002	4.468E-4
PSO-EO [39]	**100**	8031.45	9.901E-6	9.607E-6	9.999E-6	**1.1518E-04**
GA [39]	0	4936.63	29.321	20.650	52.004	6.2983
ABC [40]	0	1131.90	0.014	3.587E-4	0.047	0.013



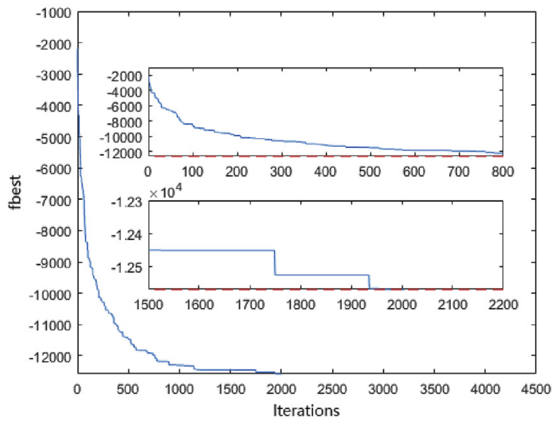
(a) ABC-EO II



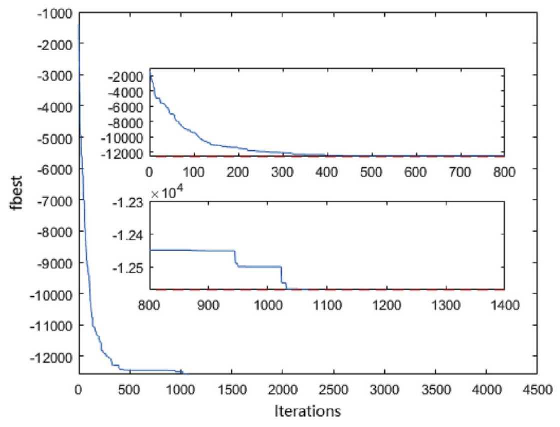
(b) IABC-EO II



(c) ABC

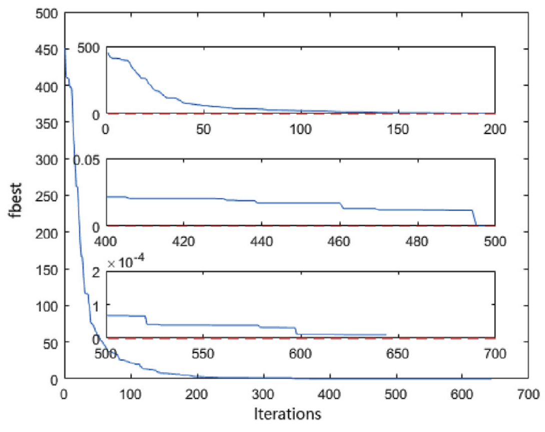


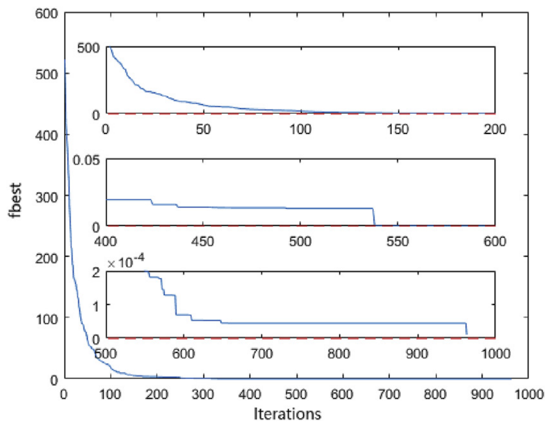
(d) ABC-EO

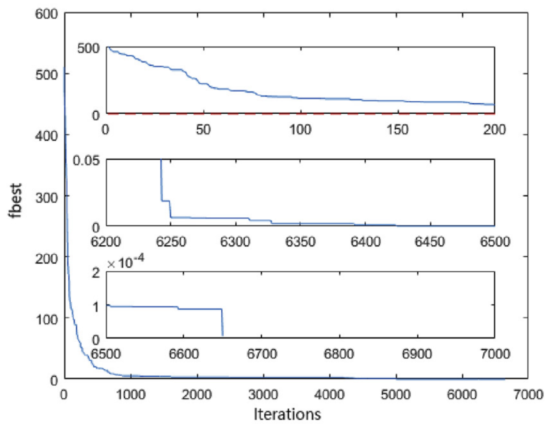


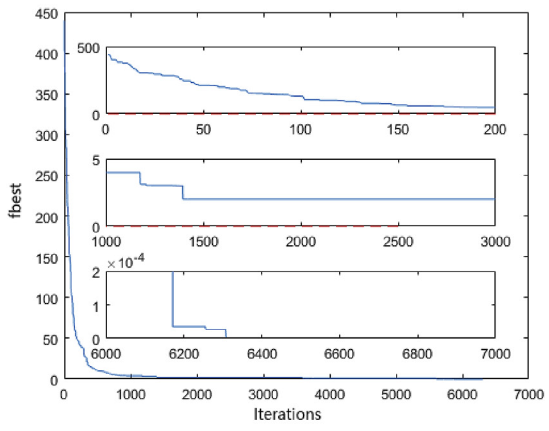
(e) IABC-EO

Fig. 9. The convergence curves of algorithms (a) ABC-EOII, (b) IABC-EO II, (c) ABC, (d) ABC-EO and (e) IABC-EO for test function F_2.


(a) ABC-EO II


(b) IABC-EO II


(c) ABC


(d) ABC-EO

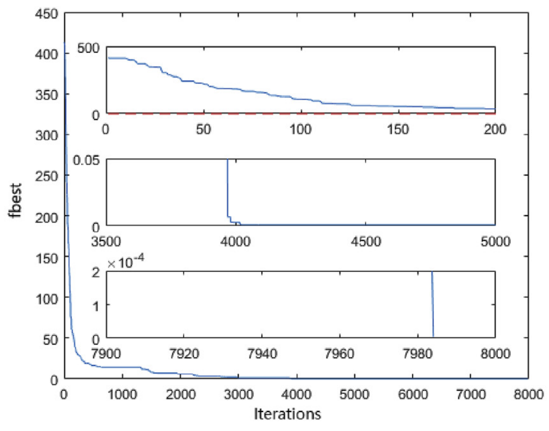

(e) IABC-EO

Fig. 10. The convergence curves of algorithms (a) ABC-EO II, (b) IABC-EO II, (c) ABC, (d) ABC-EO and (e) IABC-EO for test function F_4.

Table 17
Comparison of mean and SD values obtained by IABC-EOII, ABC-EOII, WDE, EDEV, and IJAYA for h_1-h_{10} and f_1-f_{10} problems.

Problem		IABC-EO II	ABC-EO II	WDE [48]	EDEV [49]	IJAYA [50]
h_1	Mean	**3.89817E-15**	**3.89817E-15** (≈)	6.44939E-01 (+) (+)	2.94908E-03 (+) (+)	1.66665E-10 (≈) (≈)
	SD	**1.60469E-30**	**1.60469E-30**	3.41280E-02	3.29994E-03	9.12841E-10
h_2	Mean	2.49169E-09	9.36732E-16 (≈)	9.13030E-02 (+) (+)	**1.07338E-31** (−) (−)	4.97014E+01 (+) (+)
	SD	9.98216E-09	6.64229E-16	2.69040E-02	**8.80847E-32**	8.01495E+00
h_3	Mean	3.97390E-07	**1.00088E-08** (≈)	1.45333E+02 (+) (+)	8.85513E+00 (+) (+)	7.90141E+02 (+) (+)
	SD	2.17628E-06	**5.47450E-08**	1.59497E+01	3.05105E+00	6.91842E+01
h_4	Mean	1.68704E-10	9.57162E-16 (≈)	2.66237E-04 (+) (+)	**1.99156E-51** (−) (−)	1.11846E+01 (+) (+)
	SD	8.86457E-10	5.70068E-16	6.25108E-05	**5.23965E-51**	1.72892E+00
h_5	Mean	−1.85027E+02	**−1.85027E + 02** (≈)	−1.21400E+02 (+) (+)	−1.82201E+02 (+) (+)	−3.57477E+01 (+) (+)
	SD	2.83696E-04	**4.16813E-07**	2.26725E+00	1.31348E+00	7.01053E+00
h_6	Mean	−1.21598E+02	**−1.21598E + 02** (≈)	−1.13566E+02 (+) (+)	−1.14716E+02 (+) (+)	−1.15179E+02 (+) (+)
	SD	1.76553E-11	**6.56548E-14**	1.26839E+00	2.31601E+00	3.03706E+00
h_7	Mean	2.45779E-05	**1.25995E-07** (≈)	8.73234E-01 (+) (+)	2.02649E-01 (+) (+)	2.78158E+00 (+) (+)
	SD	6.77556E-05	**1.70953E-07**	4.38994E-02	5.86449E-02	2.36442E-01
h_8	Mean	**1.70966E-06**	6.93137E-03 (+)	7.93966E+03 (+) (+)	6.03369E+02 (+) (+)	2.60571E+04 (+) (+)
	SD	**7.33016E-06**	2.82488E-02	4.43961E+02	2.35265E+02	1.98350E+03
h_9	Mean	**0.00000E + 00**	**0.00000E + 00** (≈)	1.68167E+02 (+) (+)	**0.00000E + 00** (≈) (≈)	**0.00000E + 00** (≈) (≈)
	SD	**0.00000E + 00**	**0.00000E + 00**	3.46493E+00	**0.00000E + 00**	**0.00000E + 00**
h_{10}	Mean	2.46545E-04	**1.79726E-15** (≈)	1.20282E-01 (+) (+)	2.12531E-02 (+) (+)	3.00082E+02 (+) (+)
	SD	1.35038E-03	**2.49045E-15**	5.40904E-02	4.00429E-02	8.65373E+01
f_1	Mean	**0.00000E + 00**	**0.00000E + 00** (≈)	3.67125E-94 (+) (+)	**0.00000E + 00** (≈) (≈)	2.85302E-45 (+) (+)
	SD	**0.00000E + 00**	**0.00000E + 00**	5.14546E-94	**0.00000E + 00**	1.08970E-45
f_2	Mean	**0.00000E + 00**	**0.00000E + 00** (≈)	1.04700E-107 (+) (+)	**0.00000E + 00** (≈) (≈)	1.13172E-32 (+) (+)
	SD	**0.00000E + 00**	**0.00000E + 00**	3.92074E-107	**0.00000E + 00**	4.84601E-32
f_3	Mean	**0.00000E + 00**	**0.00000E + 00** (≈)	3.88758E-22 (+) (+)	**0.00000E + 00** (≈) (≈)	1.11984E-13 (+) (+)
	SD	**0.00000E + 00**	**0.00000E + 00**	1.25974E-21	**0.00000E + 00**	8.15794E-14
f_4	Mean	3.28417E+00	1.78790E+00 (−)	3.67861E-10 (−) (−)	**4.90000e-324** (−) (−)	1.50812E+00 (−) (−)
	SD	5.32794E-01	3.01939E-01	1.06918E-10	**0.00000E + 00**	2.61320E-01
f_5	Mean	**0.00000E + 00**	**0.00000E + 00** (≈)	**0.00000E + 00** (≈) (≈)	**0.00000E + 00** (≈) (≈)	**0.00000E + 00** (≈) (≈)
	SD	**0.00000E + 00**	**0.00000E + 00**	**0.00000E + 00**	**0.00000E + 00**	**0.00000E + 00**
f_6	Mean	**0.00000E + 00**	**0.00000E + 00** (≈)	2.07242E-16 (+) (+)	1.48030E-16 (+) (+)	1.48992E-14 (+) (+)
	SD	**0.00000E + 00**	**0.00000E + 00**	5.63345E-17	1.06462E-16	9.21794E-15
f_7	Mean	5.13977E-04	4.23518E-02 (+)	6.35689E-169 (−) (−)	**0.00000E + 00** (−) (−)	9.78340E-52 (−) (−)
	SD	1.95201E-04	6.06720E-03	0.00000E+00	**0.00000E + 00**	1.20433E-51
f_8	Mean	5.13977E-04	4.23518E-02 (+)	3.65247E-02 (+) (−)	**1.86393E-29** (−) (−)	1.39334E+00 (+) (+)
	SD	1.95201E-04	6.06720E-03	8.00388E-02	**1.34000E-29**	1.66582E+00
f_9	Mean	5.13977E-04	4.23518E-02 (+)	4.00518E-06 (−) (−)	**4.44089E-15** (−) (−)	1.49408E+00 (+) (+)
	SD	1.95201E-04	6.06720E-03	1.18908E-06	**0.00000E + 00**	3.02285E-01
f_{10}	Mean	7.19500E-15	1.12784E-13 (+)	5.24586E-05 (+) (+)	**0.00000E + 00** (−) (−)	1.66684E+01 (+) (+)
	SD	3.93476E-15	1.10487E-13	9.83892E-06	**0.00000E + 00**	3.94485E+00
IABC-EO II: Better/Similar/Worse		–	4/15/1	16/1/3	8/5/7	15/3/2
ABC-EO II: Better/Similar/Worse		1/15/4	–	15/1/4	8/5/7	15/3/2

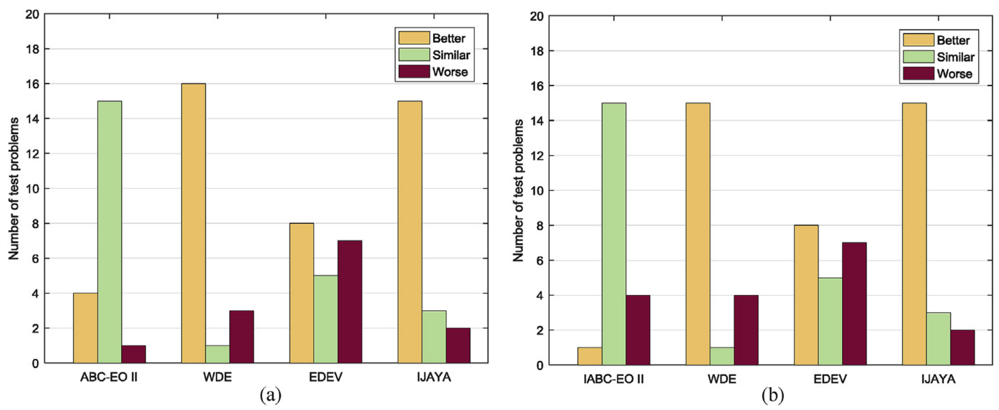


Fig. 11. The significance test of difference between two compared algorithms (a): IABC-EO II and its comparison (b): ABC-EO II and its comparison.

proposed algorithms by comparing with very recently published algorithms including new variants of DE. These compared algorithms are the weighted differential evolution algorithm (WDE) [48], ensemble of differential evolution variants (EDEV) [49] and improved JAYA (IJAYA) [50]. Here, we use 20 benchmark functions (i.e., h_1-h_{10} and f_1-f_{10}) to test the performance of IABC-EOII and ABC-EOII by comparison with WDE, EDEV and IJAYA. The common parameters of IABC-EOII and ABC-EOII and other competitors are the same with Table 1. The specific parameters of EDVE are set as: $\lambda_1 = \lambda_2 = \lambda_3 = 0.1$, $\lambda_4 = 0.7$, and $ng = 20$. Note that these related source codes[1] of the compared methods are all from the authors. Table 17 gives the comparison of mean and *SD* values obtained by five different algorithms for h_1-h_{10} problems and f_1-f_{10} problems,

[1] The WDE can be downloaded: https://www.mathworks.com/matlabcentral/fileexchange/68370-weighted-differential-evolution-algorithm -wde, The EDEV can be downloaded: http://www.ntu.edu.sg/home/epnsugan/, The IJAYA can be downloaded: http://www5.zzu.edu.cn/cilab/Code.htm.

Table 18
Ranks, the statistics, and related *p*-value achieved by Friedman and Quade tests for Exp. 5

Algorithm	Friedman	Quade
WDE [48]	3.55	3.8143
EDEV [49]	2.50	2.4595
IJAYA [50]	3.95	4.3119
ABC-EO II	**2.45**	**2.0905**
IABC-EO II	2.55	2.3238
Statistic	15.68	8.310643
p-value	0.00348	0.0000129

where the symbol "+", "≈", and "-" indicate that result of proposed algorithm is better, similar, worse to that of compared method with Wilcoxon signed-rank test at a 5% significance. Note that the there are two symbol in WDE, EDEV and IJAYA, the first symbol represents the significance test of IABC-EOII and its comparison, and the second symbol means the significance test of ABC-EOII and its comparison. The significance test of IABC-EOII and ABC-EOII is marked in row of ABC-EOII. Fig. 11 plots the significance test of difference between two compared algorithms with Wilcoxon signed-rank test at a 5% significance. From Table 17 and Fig. 11, we can see that IABC-EOII is significance better than WDE and IJAYA in most considered problems except the f_4, f_7, f_9 of WDE and f_4, f_7 of IJAYA. Compared to EDEV, the IABC-EOII has slightly better performance than EDEV. Similar with IABC-EOII, ABC-EOII is also significance better than WDE and IJAYA, while it has competitive performance with EDEV. In addition, the statistics and *p*-value obtained by the Friedman and Quade tests for the compared algorithms in Exp. 5 are listed in Table 18. Clearly, ABC-EO II achieves the best rank both in Friedman and Quade tests. As for IABC-EO II, it is the third-best one slightly worse than EDEV with a level of significance $\alpha = 0.01$ in Friedman test while IABC-EO II achieves the second-best in Quade test. Overall, the extensive experimental results of almost all benchmark problems considered in this paper show that IABC-EO II and ABC-EO II perform better than or at least competitive with other new EA variants including WDE, EDEV and IJAYA.

4.7. Exp. 6: parameters identification of photovoltaic models (real life problems)

In the former sections, we have investigated the superiority of the ABC-EO II and IABC-EO II over other competitors on the considered benchmarks. The feasibility of two proposed algorithms on the real life problems remain unknown. Here, we use two kinds of parameters identification of photovoltaic models as the real life problems to evaluate the performance of the proposed methods. By photovoltaic (PV) systems, the solar energy can be converted into the electrical energy, which is one of promising renewable energies to tackle the problems of climate change and global warming. The accuracy of PV systems largely depends on model parameters. Unfortunately, it is a challenging task to obtain these parameters because of aging, faults, or volatile operating conditions. Therefore, it is of great importance to evaluate the actual behavior of PV systems according to the measured current-voltage data [50]. As reported in Ref. [57], the most common models are the single diode model and double diode model to describe the performance and nonlinear behavior of PV systems, and the parameters identification problems of PV models are the optimization problems in essence. Fig. 12(a) gives the equivalent circuit single diode model of PV, which is widely employed to denote the static characteristics of solar cell due to its simplicity and accuracy. As shown in Fig. 12(a), this model contains a diode, shunt resistor and a series resistor, and the mathematical formula for output current can be given as follows [50]:

$$I_L = I_{ph} - I_d - I_{sh} \tag{11}$$

$$I_d = I_{sd} \cdot \left[\exp\left(\frac{q \cdot (V_L + R_s \cdot I_L)}{n \cdot k \cdot T}\right) - 1\right] \tag{12}$$

$$I_{sh} = \frac{V_L + R_S \cdot I_L}{R_{sh}} \tag{13}$$

where I_L, I_{ph}, I_d and I_{sh} are the solar cell output current, total current, diode current and shunt current, respectively. R_s represents the series, and R_{sh} denotes the shunt resistances. In addition, the V_L means the cell output voltage, n is the ideal factor of diode. k represents the Boltzmann constant, which is set as $1.3806503 \times 10^{-23}$ J/K, q is set as $1.60217646 \times 10^{-19}$ C, which is the electron charge, and T means the cell absolute temperature.

By substituting Equation (12) and Equation (13) into Equation (11), the following output current can be obtained [50]:

$$I_L = I_{ph} - I_{sd} \cdot \left[\exp\left(\frac{q \cdot (V_L + R_S \cdot I_L)}{n \cdot k \cdot T}\right) - 1\right] - \frac{V_L + R_s \cdot I_L}{R_{sh}} \tag{14}$$

There are five unknown parameters (i.e., I_{ph}, I_{sd}, R_s, R_{sh}, and n) as the decision variables needed to be identified. The fitness function is to minimize the root mean square error (RMSE for short), in other words, we need to estimate the related parameters to obtain the minimization difference between the actual experimental values and the simulated values. For the single diode model, the fitness function can be described as follows [50]:

$$\begin{cases} \mathrm{RMSE}(\mathbf{X}) = \sqrt{\dfrac{1}{N}\sum_{k=1}^{N} f_k(V_L, I_L, \mathbf{X})^2} \\ f_k(V_L, I_L, \mathbf{X}) = I_{ph} - I_{sd} \cdot \left[\exp\left(\dfrac{q \cdot (V_L + R_S \cdot I_L)}{n \cdot k \cdot T}\right) - 1\right] - \dfrac{V_L + R_S \cdot I_L}{R_{Sh}} - I_L \\ \mathbf{X} = \{I_{ph}, I_{sd}, R_S, R_{Sh}, n\} \\ s.t. \quad 0 \le I_{ph}(A) \le 1 \\ \quad 0 \le I_{sd}(A) \le 1 \\ \quad 0 \le R_s(\Omega) \le 0.5 \\ \quad 0 \le R_{sh}(\Omega) \le 100 \\ \quad 0 \le n \le 1 \end{cases} \tag{15}$$

As another widely used model, the double diode model is considered, whose equivalent circuit is presented in Fig. 12(b) and the corresponding mathematical formula of the output current can be shown as follows [50]:

$$\begin{aligned} I_L = I_{ph} - I_{d1} - I_{d2} - I_{sh} = I_{ph} - I_{sd1} \cdot \left[\exp\left(\frac{q \cdot (V_L + R_s \cdot I_L)}{n_1 \cdot k \cdot T}\right) - 1\right] \\ - I_{sd2}\left[\exp\left(\frac{q \cdot (V_L + R_S \cdot I_L)}{n_2 \cdot k \cdot T}\right) - 1\right] - \frac{V_L + R_S \cdot I_L}{R_{sh}} \end{aligned} \tag{16}$$

where I_{sd1} and I_{sd2} represent the diffusion and saturation currents, while n_1 and n_2 represent the ideal factors of diffusion and recombination diode. And the other parameters have the same meaning with Equation (14). Therefore, there are seven decide variables (i.e., I_{ph}, I_{sd1}, I_{sd2}, R_s, R_{sh}, n_1 and n_2) needed to be identified. Similar with single diode model, the fitness function of double diode model can be given as follows [50]:

$$\begin{cases} \mathrm{RMSE}(\mathbf{X}) = \sqrt{\frac{1}{N}\sum_{k=1}^{N} f_k(V_L, I_L, \mathbf{X})^2} \\ f_k(V_L, I_L, \mathbf{X}) = I_{ph} - I_{sd1}\cdot\left[\exp\left(\frac{q\cdot(V_L + R_S\cdot I_L)}{n_1\cdot k\cdot T}\right) - 1\right] - I_{sd2}\cdot\left[\exp\left(\frac{q\cdot(V_L + R_S\cdot I_L)}{n_2\cdot k\cdot T}\right) - 1\right] - \frac{VL + R_S I_L}{R_{sh}} - I_L \\ \mathbf{X} = \{I_{ph}, I_{sd1}, I_{sd2}, R_S, R_{sh}, n_1, n_2\} \\ s.t. \quad 0 \le I_{ph}(A) \le 1 \\ 0 \le I_{sd1}(A) \le 1 \\ 0 \le I_{sd2}(A) \le 1 \\ 0 \le R_s(\Omega) \le 0.5 \\ 0 \le R_{sh}(\Omega) \le 100 \\ 0 \le n_1 \le 1 \\ 0 \le n_2 \le 1 \end{cases} \tag{17}$$

To demonstrate the performance of the proposed methods, we select two recently proposed well-recognized DE variants, i.e., WDE [48], and EDEV [49], one PSO variant, i.e., PSO-DE [51] and one swarm intelligence algorithm, i.e., IJAYA [50], as the competitors. Based on the recommendations in the corresponding references, the adjustable parameters of PSO-DE are set as: $w_s = 0.9$,$w_f = 0.4$, $c_{1s} = c_{2f} = 2$, $c_{1f} = c_{2s} = 0.1$. As for the EDVE, the parameters are set as: $\lambda_1 = \lambda_2 = \lambda_3 = 0.1$, $\lambda_4 = 0.7$, and $ng = 20$. For fair comparison, the same maximum number of function evaluations is set as 12000 for each method and each problem. And other common parameters have been listed in Table 1.

Table 19 gives the statistical results of RMSE obtained by six compared algorithms, where the minimum RMSE value (Min), maximum RMSE value (Max), mean RMSE value (Mean) and standard deviation (SD) of all involved methods in 30 independent results. The best term is highlighted in bold. More specifically, the minimum RMSE implies the identified accuracy, and mean value reflects the average accuracy. In addition, the SD value indicates the robustness and stability of the identified PV model parameters. From Table 19, for the single diode model, IABC-EO II ranks the first in terms of all the statistical results. Compared with other four methods, ABC-EO II achieves better performance than WDE, PSO-DE and EDVE, and it obtains the better minimum value than that of IJAYA although other terms are worse than IJAYA. Similar to the single diode model, for the double diode model, IABC-EO II achieves the best performance among all the algorithms. As for ABC-EO II, its performance is better than other four compared algorithms, except the minimum RMSE value where PSO-DE is better than ABC-EO II. Overall, IABC-EO II has stronger ability than other counterparts in the parameter identification problems of PV model, and ABC-EO II outperforms WDE, PSO-DE and EDVE and obtains the similar performance with IJAYA.

Besides, some statistical tests [55,58] for different algorithms are adopted to compare the performance of IABC-EO II, ABC-EO II, IJAYA, WDE, PSO-DE and EDVE for two considered problems of PV models. For example, Fig. 13 plots the analysis of the variance (ANOVA) test of RMSE values achieved by six different algorithms. Clearly, IABC-EO II, ABC-EO

Table 19
Statistical results of RMSE of different algorithms for two models.

Model	Algorithm	RMSE			
		Min	Mean	Max	SD
Single diode model	IABC-EO II	**9.98518E-04**	**1.18679E-03**	**1.47536E-03**	**1.40224E-04**
	ABC-EO II	9.98540E-04	1.28810E-03	1.75662E-03	1.66900E-04
	IJAYA [50]	1.02041E-03	1.22391E-03	1.50175E-03	1.45876E-04
	WDE [48]	2.09764E-03	4.51837E-03	7.62968E-03	1.39303E-03
	PSO-DE [51]	1.02025E-03	2.10476E-03	7.70745E-03	1.14993E-03
	EDVE [49]	1.74303E-03	2.70520E-03	4.13930E-03	5.29477E-04
Double diode model	IABC-EO II	**9.89900E-04**	**1.14791E-03**	**1.49748E-03**	**1.21023E-04**
	ABC-EO II	9.94826E-04	1.21924E-03	1.57275E-03	1.39181E-04
	IJAYA [50]	9.96330E-04	1.23167E-03	1.58804E-03	1.85824E-04
	WDE [48]	2.06197E-03	5.07873E-03	7.73191E-03	1.82377E-03
	PSO-DE [51]	9.94074E-04	2.11314E-03	3.74598E-03	8.48271E-04
	EDVE [49]	1.04526E-03	1.46801E-03	1.88118E-03	1.98605E-04

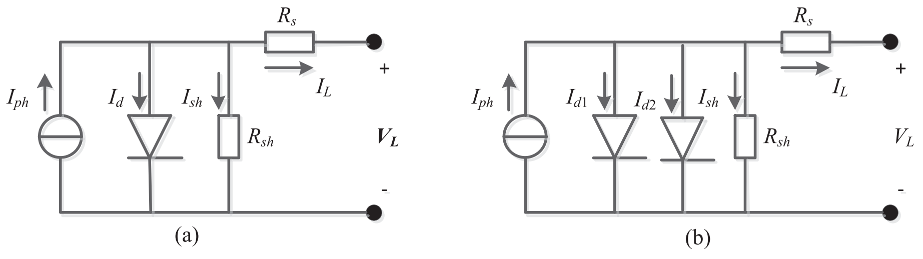


Fig. 12. Schematics of equivalent circuit for PV model (a): single diode model; (b) double diode model.

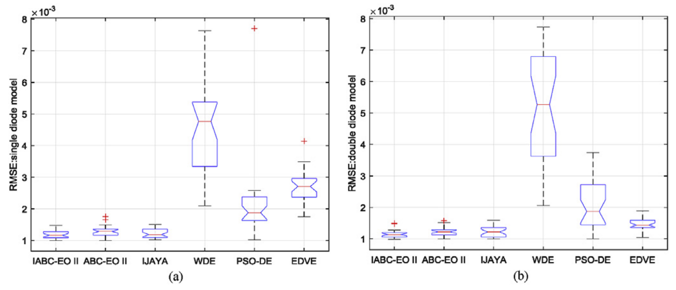


Fig. 13. ANOVA tests of fitness obtained by different algorithms for parameters identification (a): single diode model (b): double diode model.

Table 20
The significance test of difference between IABC-EO II and its competitors.

IABC-EO II vs.	Single diode model		Double diode model	
	Sig.	*p*-value	Sig.	*p*-value
ABC-EO II	+	0.02183	+	0.04492
IJAYA [50]	≈	0.2452	≈	0.06268
WDE [48]	+	1.734e-06	+	1.734e-06
PSO-DE [51]	+	2.603e-06	+	8.466e-06
EDVE [49]	+	1.734e-06	+	5.752e-06

Table 21
The significance test of difference between ABC-EO II and its competitors.

ABC-EO II vs.	Single diode model		Double diode model	
	Sig.	p-value	Sig.	p-value
IABC-EO II	–	0.02183	–	0.04492
IJAYA [50]	≈	0.09368	≈	0.8130
WDE [48]	+	1.734e-06	+	1.734e-06
PSO-DE [51]	+	6.339e-06	+	2.597e-05
EDVE [49]	+	1.734e-06	+	1.057e-04

II and IJAYA are much better than WDE, PSO-DE and EDVE. Specifically, for the double diode model, IABC-EO II shows better performance than ABC-EO II and IJAYA. Table 20 presents the significance test of difference between IABC-EO II and its counterparts, where "+", "≈", and "-" indicate that the results of IABC-EO II are better than, similar to, and worse than those of its compared method with Wilcoxon signed-rank test at a 5% significance respectively. For both PV models, IABC-EO II significantly outperforms ABC-EO II, WDE, PSO-DE and EDVE. It should be noted that although the results of IABC-EO II have no significant difference with those of IJAYA, any reduction in the objective function will have great impact because it may cause significant improvement in the knowledge on the actual parameters. Similar to Table 20, Table 21 gives the significance test of difference between ABC-EO II and other methods. From Table 21, it is obvious that ABC-EO II is much better than WDE, PSO-DE and EDVE, and has no significant difference with IJAYA. To further confirm the quality results of IABC-EO II, the best parameters obtained by IABC-EO II are employed to reconstruct the I–V and P–V characteristics, whose curves are given in Figs. 14 and 15 for single diode model and double diode model, respectively. Clearly, we can see that the calculated data are highly coincident with the measured data.

5. Conclusion and future work

Through introducing EO to ABC and IABC respectively, this paper develops two novel hybrid optimization algorithms, called ABC-EO II and IABC-EO II, to solve unconstrained continuous optimization problems. The advanced advantages of our algorithms can be summarized from the following four aspects. Firstly, different from ABC-EO and IABC-EO which introduce EO to ABC for all food sources at several iterations, in this paper, EO is employed to change the position of only one food source at each iteration, and hence the computational costs of our algorithms are lower than those of ABC-EO and IABC-EO when solving complex hard optimization problems. Secondly, through selecting neighbor solutions randomly, a novel mutation operator adopted in this work is able to increase the diversity of the new offspring, which is helpful for our algorithms jumping out of local optima. Moreover, this mutation in our work is much easier to operate in comparison with ABC-EO and IABC-EO. Thirdly, the Boltzmann selection probability is utilized by the artificial onlooker bee to choose one food source, and as a result the selection pressure of the proposed algorithms can be dynamically adjusted in the evolutionary process. Finally, we propose a novel selection probability, according to which those worse solutions have more opportunities to be selected and then mutated by EO. Compared with 15 state-of-the-art optimization algorithms on three groups of well-known benchmark functions, ABC-EO II and IABC-EO II have been testified to perform well or superiorly in terms of successful rate, convergence speed, solution quality and statistical tests. The simulation results indicate that the proposed algorithms have a great potential of solving a variety of complex unimodal/multimodal, continuous/discontinuous or noisy quartic functions with high dimension or multiple local optima. In order to demonstrate the feasibilities of our proposed algorithms on the real life problems, our algorithms are used to deal with two parameters identification problems of photovoltaic models in comparison with two well-recognized DE variants, i.e., WDE [48] and EDEV [49], one PSO variant, i.e., PSO-DE [51], and one swarm intelligence algorithm, i.e., IJAYA [50]. Experimental results have shown that IABC-EO II has stronger ability than other counterparts in the parameter identification problems of PV model, and ABC-EO II outperforms WDE, PSO-DE and EDVE and performs nearly as well as IJAYA.

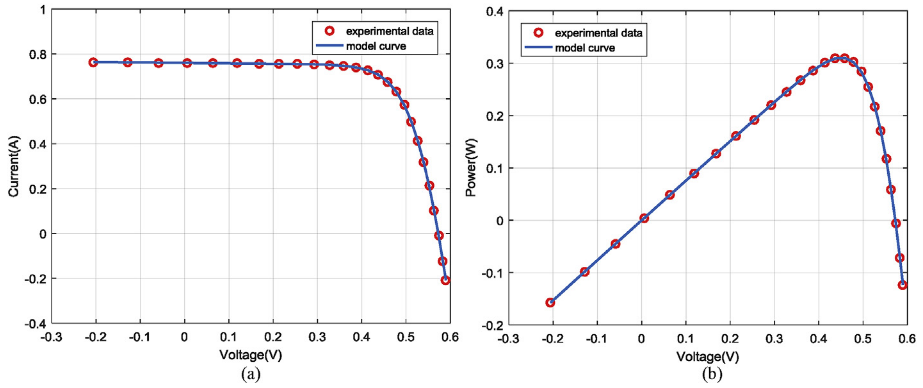


Fig. 14. Comparisons between experimental data and the data achieved by IABC-EO II for parameters identification of single diode model (a) I–V characteristics; (b) P–V characteristics.

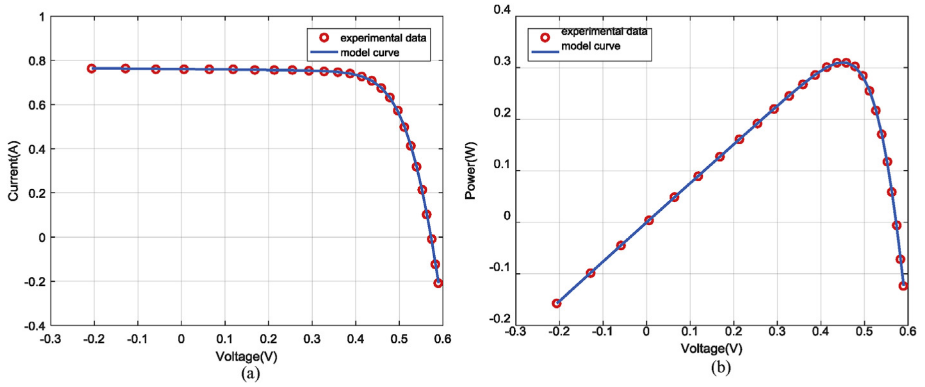


Fig. 15. Comparisons between experimental data and the data achieved by IABC-EO II for parameters identification of double diode model (a) I–V characteristics; (b) P–V characteristics.

In the recent years, the discrete artificial bee colony algorithm and its improved versions have already been widely used for some important engineering optimization problems, e.g., flow shop and hybrid flow shop scheduling problems [59–66]. In fact, the basic idea behind ABC-EO II and IABC-EO II will be extended to solve those engineering scheduling problems although they were originally developed for the continuous optimization problems in this paper. However, the following modifications and improvements should be generally made in ABC-EO II and IABC-EO II according to the characteristics of flow shop and hybrid flow shop scheduling problems. Firstly, the permutation-based encoding mechanism or some improved ones, e.g., the dynamic encoding mechanism [61], and a flexible decoding mechanism can be introduced for a specific scheduling problem, then the discrete ABC operations, e.g., employed bee strategy and onlooker bee strategy [60] can be designed for the considered problem. Secondly, a discrete framework of extremal optimization should be designed to improve the ability of local search [38,67]. Thirdly, generation of variable discrete neighboring solutions is another important operation to help the algorithms jump out of local optimum regions [59]. In order to handle different constraints of real-world flowshop scheduling problems, some adaptive constraints dealing techniques also need to be adopted in ABC-EO II and IABC-EO II. Additionally, some hybrid multi-objective optimization methods combining multi-objective ABC [65] and other evolutionary algorithms [36,66] can be developed for various complex blocking lot-streaming flow shop scheduling problems with machine breakdown.

Moreover, the proposed algorithms in this work will be applied to solving other more complex engineering optimization problems, such as optimal control issue of industrial processes and energy management problem of smart grid. In order to further improve the performance of the proposed methods, some dynamic multi-swarm PSO [68] and adaptive ABC algorithms [69,70] can also be introduced to our ABC-EO II and IABC-EO II methods.

Acknowledgments

The authors declared that they have no conflicts of interest to this work. We would like to thank all the anonymous reviewers very much for their valuable comments and useful suggestions. This work was supported by National Natural Science Foundation of China (Grant Nos. 61872153, 61373158, 51207112, 61171124, 6130128), Zhejiang Provincial Natural Science Foundation of China (Grant Nos. LY16F030011 and LZ16E050002), and Natural Science Foundation of Guangdong Province (No.2018A030313318).

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.swevo.2019.06.005.

References

[1] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization, Erciyes University, Turkey, 2005.
[2] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Glob. Optim. 3 (2007) 459–471.
[3] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft Comput. 8 (1) (2008) 687–697.
[4] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, Appl. Math. Comput. 214 (1) (2009) 108–132.
[5] A. Rajasekhar, N. Lynn, S. Das, P.N. Suganthan, Computing with the collective intelligence of honeybees – a survey, Swarm and Evolutionary Computation 32 (2017) 25–48.
[6] R. Akbari, R. Hedayatzadeh, K. Ziarati, B. Hassanizadeh, A multi-objective artificial bee colony algorithm, Swarm and Evolutionary Computation 2 (2012) 39–52.
[7] X.Z. Xie, X.L. Wang, An enhanced ABC algorithm for single machine order acceptance and scheduling with class setups, Appl. Soft Comput. 44 (2016) 255–266.
[8] I. Khan, M.K. Maiti, A swap sequence based artificial bee colony algorithm for traveling salesman problem, Swarm and Evolutionary Computation 44 (2019) 428–438.
[9] M. Li, Y.Q. Hei, Z. Qiu, Optimization of multiband cooperative spectrum sensing with modified artificial bee colony algorithm, Appl. Soft Comput. 57 (2017) 751–759.
[10] A. Saad, S.A. Khan, A. Mahmood, A multi-objective evolutionary artificial bee colony algorithm for optimizing network topology design, Swarm and Evolutionary Computation 38 (2018) 187–201.
[11] D. Kumar, K.K. Mishra, Portfolio optimization using novel co-variance guided artificial bee colony, Swarm and Evolutionary Computation 33 (2017) 119–130.
[12] J.J. Zhou, X.F. Yao, Multi-population parallel self-adaptive differential artificial bee colony algorithm with application in large-scale service composition for cloud manufacturing, Appl. Soft Comput. 56 (2017) 379–397.
[13] D. Karaboga, E. Kaya, An adaptive and hybrid artificial bee colony algorithm (aABC) for ANFIS training, Appl. Soft Comput. 49 (2016) 423–436.
[14] J.P. Luo, Q.Q. Liu, Y. Yang, X. Li, M.R. Chen, W.M. Cao, An artificial bee colony algorithm for multi-objective optimisation, Appl. Soft Comput. 50 (2017) 235–251.
[15] W.L. Xiang, X.L. Meng, Y.Z. Li, R.C. He, M.Q. An, An improved artificial bee colony algorithm based on the gravity model, Inf. Sci. 429 (2018) 49–71.
[16] W.F. Gao, S.Y. Liu, L.L. Huang, Enhancing artificial bee colony algorithm using more information-based search equations, Inf. Sci. 270 (2014) 112–133.
[17] W.F. Gao, L.L. Huang, J. Wang, S.Y. Liu, C.D. Qin, Enhanced artificial bee colony algorithm through differential evolution, Appl. Soft Comput. 48 (2016) 137–150.
[18] L.Z. Cui, G. H Li, Z.X. Zhu, Q.Z. Lin, Z.K. Wen, N. Lu, K.C. Wong, J.Y. Chen, A novel artificial bee colony algorithm with an adaptive population size for numerical function optimization, Inf. Sci. 414 (2017) 53–67.
[19] L.Z. Cui, G.H. Li, Y.L. Luo, F. Chen, Z. Ming, N. Lu, J. Lu, An enhanced artificial bee colony algorithm with dual-population framework, Swarm and Evolutionary Computation 43 (2018) 184–206.
[20] L.Z. Cui, G.H. Li, X.Z. Wang, Q.Z. Lin, J. Lu, A ranking-based adaptive artificial bee colony algorithm for global numerical optimization, Inf. Sci. 417 (2017) 169–185.
[21] D.P. Kong, T.Q. Chang, W.J. Dai, Q.D. Wang, H.Z. Sun, An improved artificial bee colony algorithm based on elite group guidance and combined breadth-depth search strategy, Inf. Sci. 442 (2018) 54–71.
[22] Z.P. Liang, K.F. Hu, Q.X. Zhu, Z.X. Zhu, An enhanced artificial bee colony algorithm with adaptive differential operators, Appl. Soft Comput. 58 (2017) 480–494.
[23] Z.H. Ding, M. Huang, Z.R. Lu, Structural damage detection using artificial bee colony algorithm, Swarm and Evolutionary Computation 28 (2016) 1–13.
[24] Q.Z. Lin, M. M Zhu, G.H. Li, W.J. Wang, L.Z. Cui, J.Y. Chen, J. Lu, A novel artificial bee colony algorithm with local and global information interaction, Appl. Soft Comput. 62 (2018) 702–735.
[25] W.F. Gao, S.Y. Liu, L.L. Huang, Inspired artificial bee colony algorithm for global optimization problems, Acta Electron. Sin. 12 (2012) 2396–2403.

[26] P. Bak, K. Sneppen, Punctuated equilibrium and criticality in a simple model of evolution, Phys. Rev. Lett. 71 (1993) 4083–4086.
[27] P. Bak, C. Tang, K. Wiesenfeld, Self-organized criticality, Phys. Rev. Lett. 59 (1987) 381–384.
[28] S. Boettcher, A.G. Percus, Extremal optimization: methods derived from co-evolution, in: Proceedings of the Genetic and Evolutionary Computation Conference, 1999, pp. 825–832.
[29] S. Boettcher, A.G. Percus, Nature's way of optimizing, Artif. Intell. 119 (2000) 275–286.
[30] Y.Z. Lu, M.R. Chen, Y.W. Chen, Studies on extremal optimization and its applications in solving real world optimization problems, in: Proceedings of the 2007 IEEE Symposium on Foundations of Computational Intelligence (FOCI 2007), Hawaii, USA, 2007, pp. 162–168.
[31] M.R. Chen, Y.Z. Lu, G.K. Yang, Multiobjective extremal optimization with applications to engineering design, J. Zhejiang Univ. - Sci. 8 (2007) 1905–1911.
[32] M.R. Chen, Y.Z. Lu, A novel elitist multiobjective optimization algorithm: multiobjective extremal optimisation, Eur. J. Oper. Res. 188 (2008) 637–651.
[33] M.R. Chen, Y.Z. Lu, G.K. Yang, Multiobjective optimization using population-based extremal optimization, Journal of Neural Computing and Applications 7 (2008) 101–109.
[34] M.R. Chen, Y.Z. Lu, G.K. Yang, Population-based extremal optimization with adaptive Lévy mutation for constrained optimization, in: Proceedings of 2006 International Conference on Computational Intelligence and Security (CIS'06), 2006, pp. 258–261.
[35] X. Li, J.P. Luo, M.R. Chen, N. Wang, An improved shuffled frog-leaping algorithm with extremal optimisation for continuous optimization, Inf. Sci. 192 (2012) 143–151.
[36] G.Q. Zeng, J. Chen, L.M. Li, M.R. Chen, L. Wu, Y.X. Dai, C.W. Zheng, An improved multi-objective population-based extremal optimization algorithm with polynomial mutation, Inf. Sci. 330 (2016) 49–73.
[37] G.Q. Zeng, J. Chen, Y.X. Dai, L.M. Li, C.W. Zheng, M.R. Chen, Design of fractional order PID controller for automatic regulator voltage system based on multi-objective extremal optimization, Neurocomputing 160 (2015) 173–184.
[38] Y.Z. Lu, Y.W. Chen, M.R. Chen, P. Chen, G.Q. Zeng, Extremal Optmization: Fundamentals, Algorithms, and Applications, CRC Press & Chemical Industry Press, 2016.
[39] M.R. Chen, X. Li, X. Zhang, Y.Z. Lu, A novel particle swarm optimizer hybridized with extremal optimization, Appl. Soft Comput. 10 (2010) 367–373.
[40] M.R. Chen, W. Zeng, G.Q. Zeng, X. Li, J.P. Luo, A novel artificial bee colony algorithm with integration of extremal optimization for numerical optimization problems, in: Proceedings of the 2014 IEEE Conference on Evolutionary Computation, 2014, pp. 242–249.
[41] T. Back, Selective pressure in evolutionary algorithms: a characterization of selection mechanisms, in: Proceedings of 1st IEEE Conference on Evolutionary Computation, 1994, pp. 57–62.
[42] L.M. Li, K.D. Lu, G.Q. Zeng, L. Wu, M.R. Chen, A novel real-coded population-based extremal optimization algorithm with polynomial mutation: a non-parametric statistical study on continuous optimization problems, Neurocomputing 174 (2016) 577–587.
[43] G.P. Zhu, S. Kwong, Gbest-guided artificial bee colony algorithm for numerical function optimization, Appl. Math. Comput. 217 (2010) 3166–3173.
[44] W.F. Gao, S.Y. Liu, L.L. Huang, A novel artificial bee colony algorithm with Powell's method, Appl. Soft Comput. 13 (2013) 3763–3775.
[45] P.H. Tang, M.H. Tseng, Adaptive directed mutation for real-coded genetic algorithms, Appl. Soft Comput. 13 (2013) 600–614.
[46] S.Y. Ho, L.S. Shu, J.H. Chen, Intelligent evolutionary algorithms for large parameter optimization problems, IEEE Trans. Evol. Comput. 8 (6) (2004) 522–541.
[47] 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.
[48] P. Civicioglu, E. Besdok, M.A. Gunen, U.K. Atasever, Weighted differential evolution algorithm for numerical function optimization: a comparative study with cuckoo search, artificial bee colony, adaptive differential evolution, and backtracking search optimization algorithms, Neural Comput. Appl. (2018), https://doi.org/10.1007/s00521-018-3822-5 (published online).
[49] G.H. Wu, X. Shen, H.F. Li, H.K. Chen, A.P. Lin, P.N. Suganthan, Ensemble of differential evolution variants, Inf. Sci. 423 (2018) 172–186.
[50] K.J. Yu, J. Liang, B.Y. Qu, X. Chen, H.S. Wang, Parameters identification of photovoltaic models using an improved JAYA optimization algorithm, Energy Convers. Manag. 150 (2017) 742–753.
[51] B.W. Tang, K. Xiang, M.Y. Pang, An integrated particle swarm optimization approach hybridizing a new self-adaptive particle swarm optimization with a modified differential evolution, Neural Comput. Appl. (2018), https://doi.org/10.1007/s00521-018-3878-2 (published online).
[52] D. Karaboga, B. Akay, A modified artificial bee colony (ABC) algorithm for constrained optimization, Appl. Soft Comput. 11 (2011) 3021–3031.
[53] V. Azadehgan, N. Jafarian, F. Jafarieh, A novel hybrid artificial bee colony with extremal optimization, in: Proceedings of 4th International Conference on Computer and Electrical Engineering, ICCEE, 2011, pp. 45–49, 2011.
[54] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of nonparametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization, J. Heuristics 15 (6) (2009) 617–644.
[55] 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 Evolutionary Computation 1 (1) (2011) 3–18.
[56] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms to data mining problems, Soft Computing 13 (3) (2009) 307–318.
[57] D.F. Alam, D.A. Yousri, M.B. Eteiba, Flower pollination algorithm based solar PV parameter estimation, Energy Convers. Manag. 101 (2015) 410–422.
[58] G.Q. Zeng, X.Q. Xie, M.R. Chen, J. Weng, Adaptive population extremal optimization-based PID neural network for multivariable nonlinear control systems, Swarm and Evolutionary Computation 44 (2019) 320–334.
[59] Q.K. Pan, L. Wang, J.Q. Li, J.H. Duan, A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation, Omega 45 (2014) 42–56.
[60] J.Q. Li, Q.K. Pan, Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm, Inf. Sci. 316 (2015) 487–502.
[61] K.Z. Gao, P.N. Suganthan, Q.K. Pan, T.J. Chua, C.S. Chong, T.X. Cai, An improved artificial bee colony algorithm for flexible job-shop scheduling problem with fuzzy processing time, Expert Syst. Appl. 65 (2016) 52–67.
[62] J.Q. Li, Q.K. Pan, P.Y. Duan, An improved artificial bee colony algorithm for solving hybrid flexible flowshop with dynamic operation skipping, IEEE Transactions on Cybernetics 46 (6) (2016) 1311–1324.
[63] K.K. Peng, Q.K. Pan, L. Gao, B. Zhang, X.F. Pang, An Improved Artificial Bee Colony algorithm for real-world hybrid flowshop rescheduling in steelmaking-refining-continuous casting process, Comput. Ind. Eng. 122 (2018) 235–250.
[64] T. Meng, Q.K. Pan, H.Y. Sang, A hybrid artificial bee colony algorithm for a flexible job shop scheduling problem with overlapping in operations, Int. J. Prod. Res. 56 (16) (2018) 5278–5292.
[65] D.W. Gong, Y.Y. Han, J.Y. Sun, A novel hybrid multi-objective artificial bee colony algorithm for blocking lot-streaming flow shop scheduling problems, Knowl. Based Syst. 148 (2018) 115–130.
[66] Y.Y. Han, D.W. Gong, Y.C. Jin, Q.K. Pan, Evolutionary multiobjective blocking lot-streaming flow shop scheduling with machine breakdowns, IEEE Transactions on Cybernetics 49 (1) (2019) 184–197.
[67] Y.W. Chen, Y.Z. Lu, M. Ge, C.C. Pan, Development of hybrid evolutionary algorithms for production scheduling of hot strip mill, Comput. Oper. Res. 39 (2) (2012) 339–349.
[68] S.Z. Zhao, P.N. Suganthan, S. Das, Dynamic multi-swarm particle swarm optimizer with sub-regional harmony search, in: Proceedings of IEEE Congress on Evolutionary Computation, 2010, https://doi.org/10.1109/cec.2010.5586323.
[69] S. Das, S. Biswas, S. Kundu, Synergizing fitness learning with proximity-based food source selection in artificial bee colony algorithm for numerical optimization, Appl. Soft Comput. 13 (12) (2013) 4676–4694.
[70] P. Venkatesh, A. Singh, An artificial bee colony algorithm with variable degree of perturbation for the generalized covering traveling salesman problem, Appl. Soft Comput. 78 (2019) 481–495.