Computers & Industrial Engineering 135 (2019) 299–313

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Ensemble of cuckoo search variants

Jiatang Cheng[a,b], Lei Wang[a,*], Yan Xiong[b]

[a] The Faculty of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China
[b] The Engineering College, Honghe University, Mengzi 661199, China

ARTICLE INFO

Keywords:
Cuckoo search
Ensemble
Selection probability
External archive
Numerical optimization

ABSTRACT

Cuckoo search is a simple yet effective evolutionary algorithm for solving numerical optimization problems. Recently, many variants of cuckoo search have been developed to further enhance the performance. These improved versions have different capabilities in tackling the optimization problems with different properties, so it is difficult to determine which algorithm is best for all problems. To address this issue, we present a new cuckoo search algorithm named the ensemble cuckoo search variant. In this developed version, a candidate pool consisting of three different cuckoo search algorithms is first constructed. According to the previous experiences in producing promising solutions, an adaptive scheme is then used to determine the probability that each algorithm can be assigned to distinct individuals in the current population. Also, an external archive is embedded to further discourage premature convergence. To assess the performance of this ensemble algorithm, 42 test problems derived from CEC 2005 and CEC 2013 are employed. Experimental results indicate that the proposed algorithm is a competitive method compared with seven well-established cuckoo search variants and several other well-known evolutionary algorithms.

1. Introduction

Cuckoo search (CS) (Yang & Deb, 2014) algorithm is a meta-heuristic optimization technique inspired by the brood parasitism behavior of some cuckoos. In CS, new individuals can be generated by using Levy flight and biased random walk, corresponding to exploration and exploitation respectively (Salgotra, Singh, & Saha, 2018). Because of its simple concept and effectiveness, CS has attracted wide attention. So far, CS has been successfully applied in various fields, such as mining industry (Fouladgar, Hasanipanah, & Amnieh, 2017), reliability redundancy allocation problems (Valian & Valian, 2013), scheduling (Laha & Gupta, 2018; Majumder, Laha, & Suganthan, 2018), dynamic optimization (Kordestani, Firouzjaee, & Meybodi, 2018), fault diagnosis (Cheng, Wang, & Xiong, 2019), damage detection (Samir, Brahim, Capozucca, & Wahab, 2018), machining process (Yildiz, 2013), hydropower station operation (Meng, Chang, Wang, & Wang, 2019) and solar photovoltaic system (Chen & Yu, 2019).

Similar to other optimization techniques (Goli, Tirkolaee, Malmir, Bian, & Sangaiah, 2019; Li & Cheng, 2017; MiarNaeimi, Azizyan, & Rashki, 2018; Rao, Savsani, & Vakharia, 2012; Salza & Ferrucci, 2019; Storn & Price, 1997; Tirkolaee, Goli, Hematian, Sangaiah, & Han, 2019; Vafashoar & Meybodi, 2018), the search capability of CS algorithm depends heavily on the trade-off between exploration and exploitation. As a result, many CS variants have been developed to enhance the convergence performance. There is no doubt that these CS algorithms exhibit their respective advantages in different aspects. However, different CS variants may have different search characteristics and are suitable for solving different types of optimization problems. For a specific problem, the best generation strategies and parameter settings may vary at different phases of the search process. Therefore, it is often very time-consuming and inefficient to use traditional trial-and-error methods to determine appropriate strategies and parameters, especially when tackling optimization problems with different properties. Further, No free lunch theorem has proved that no algorithm is suitable for solving all optimization problems (Wolpert & Macready, 1997). In response to these challenges, the ensemble of multiple strategies or variants of an algorithm has attracted more and more attention, and has become a promising method to enhance the performance of the algorithm (Wu et al., 2018).

Motivated by these observations, we develop a novel CS algorithm named the ensemble CS variant (ECSV), in which an adaptive selection scheme is employed to exploit the merits of different competitive algorithms. In the presented ECSV, three different CS algorithms, namely chaos-enhanced cuckoo search (CCS) (Huang, Ding, Yu, Wang, & Lu, 2016), nearest neighbour cuckoo search (NNCS) (Wang, Zhong, & Yin, 2016) and peer-learning cuckoo search (PLCS) (Yang, Gao, & Zhang,

* Corresponding author.
E-mail address: leeiwang@163.com (L. Wang).

https://doi.org/10.1016/j.cie.2019.06.015
Received 13 January 2019; Received in revised form 25 May 2019; Accepted 8 June 2019
Available online 10 June 2019
0360-8352/

2017), are integrated to complement each other. To achieve the effective ensemble of these algorithms, the selection probability is defined according to the success rate of the promising solutions generated by each CS variant in a certain number of previous generations. It should be noted that each constituent algorithm has the same selection probability at the beginning of the iteration. As the iteration proceeds, the constituent algorithm that produces better solutions has larger probability of being selected. Next, these algorithms are assigned to distinct individuals in the population on the basis of the selection probabilities. Besides, a new parameter named learning period (*LP*) is predefined to represent a fixed number of iterations. After *LP* generations, the success rate of each CS variant is updated and the probability selection operation is then triggered periodically. The higher the success rate of the constituent algorithm, the greater the probability of being selected. To further strengthen the exploration capability, an external archive is introduced into the biased random walk strategy to store these discarded individuals. In this modified scheme, the random solution is not selected from the current population, but from the union of the current population and external archive.

Obviously, unlike previous hybrid CS algorithms, ECSV focuses on the ensemble of different CS variants. By combining the merits of these constituent algorithms to maintain population diversity and delay convergence, ECSV can exhibit better performance in solving optimization problems with different characteristics. To investigate the advantages of the presented ECSV algorithm, extensive experiments are conducted on 42 test problems derived from CEC 2005 (Suganthan, Hansen, Liang, Deb, Chen, Auger, & Tiwari, 2005) and CEC 2013 (Liang, Qu, Suganthan, & Hernández-Díaz, 2013). For comparative purpose, seven recently-developed CS algorithms are employed. Moreover, the developed ECSV is compared with seven well-known variants of particle swarm optimization (PSO) and differential evolution (DE).

In summary, the main contributions of this paper are as follows.

(1) Considering that the search ability of an algorithm may vary significantly with different optimization problems, a candidate pool composed of three different CS variants is constructed, and a new CS algorithm called ECSV is then developed.
(2) According to the previous experiences in producing improved solutions, an adaptive scheme is employed to determine the probability of each CS algorithm being selected. Also, a fixed number of iterations named learning period is predefined and the probability matching operation is executed periodically.
(3) To further maintain population diversity and delay premature convergence, an external archive is embedded to store the discarded solutions in the selection phase.

The rest of this paper is organized as follows. Section 2 introduces the related works, and Section 3 presents the ensemble CS variant. In Section 4, the experimental results are reported and discussed. Section 5 summarizes the experimental results. Finally, Section 6 concludes this paper.

2. Related works

2.1. Cuckoo search algorithm

Cuckoo search (CS) algorithm is a newly developed global optimizer, which mimics the obligate brood parasitic behavior of some cuckoo species. For the sake of finding potential candidate solutions, Levy flight and biased random walk are used in CS algorithm. Levy flight is a global random walk which the step-length obeys a heavy-tailed probability distribution. Because of the application of Levy flight, CS is more effective in exploring solution space (Cheng, Wang, Jiang, Cao, & Xiong, 2018).

In the initialization phase, an initial population is randomly produced in the solution space of the problem being solved. The process is described using the following equation:

$$x_{i,j}^{0} = a_j + rand\cdot(b_j - a_j) \quad (i = 1, 2, \cdots, N, \quad j = 1, 2, \cdots, D) \tag{1}$$

where $x_{i,j}^{0}$ is the initialization solution, a_j and b_j stand for the lower and upper bounds of solution space, respectively. N is the population size, D is the dimension of solution space, and *rand* is a random number uniformly distributed within the range [0, 1].

To generate new candidate solutions, Levy flight can be expressed as follows:

$$x_i^{t+1} = x_i^t + step\cdot Levy(\lambda) \tag{2}$$

$$step = \alpha\cdot(x_i^t - x_{best}) \tag{3}$$

where x_i^t is the current solution, x_i^{t+1} stands for the new solution, λ denotes the power coefficient, α is the step size factor, depending on the scale of the problem being solved, and *step* stands for the random search factor, which utilizes the information of the current best solution x_{best}. $Levy(\lambda)$ represents a random number drawn from the Levy distribution and is given as:

$$Levy(\lambda) = \frac{m}{|n|^{1/\beta}} \tag{4}$$

where m and n stand for two random numbers following the normal distribution, β represents a distribution parameter.

In biased random walk, a fraction of new solutions are generated by using the following equation:

$$x_i^{t+1} = x_i^t + r\cdot(x_{r1}^t - x_{r2}^t) \tag{5}$$

where r is the scaling factor bounded in the range [0, 1], x_{r1} and x_{r2} are two randomly chosen solutions in the current population.

2.2. Some variants of CS

Practice has proved that CS algorithm is prone to premature convergence in addressing optimization problems with complex landscapes. For the purpose of enhancing the convergence performance, many attempts have been made in recent years and some promising CS variants have been developed. Generally speaking, these improved CS algorithms can be divided into two categories: parameter control and hybridization (Abdel-Basset, Hessin, & Abdel-Fatah, 2018).

In terms of parameter control, some work has been done to improve CS. Walton, Hassan, Morgan, and Brown (2011) proposed a modified CS algorithm while maintaining the attractiveness of the original method. For this modified version, the step size of Levy flight varied with the number of generations. Also, the information exchange between individuals was introduced to accelerate convergence. Then, 7 test problems were considered to investigate the performance of the proposed algorithm. Jia, Yu, Wu, Wei, and Law (2016) developed an improved CS to address the affinity propagation model. With respect to this variant, a parameter was added to regulate the step size and discovery probability. Next, the performance of this improved algorithm was tested on several benchmark problems. It was seen that the proposed method is superior to CS and other two evolutionary algorithms. Li and Yin (2015) presented a modified CS with adaptive parameters. First, two mutation rules were designed to balance the exploitation and exploration capacities. Next, a linear decreasing probability scheme was employed to control these mutation rules. Finally, according to the relative success number of the presented two new parameters, an adaptive parameter setting strategy was introduced. To investigate the performance of the proposed algorithm, 16 benchmark test problems were employed. Ma, Li, Li, Lv, and Wang (2018) developed a dynamic self-adaption CS for numerical optimization. In this scheme, the population was divided into two subgroups, each of which used different strategy to generate candidate solutions. Also, the step size was adaptively controlled to further alleviate premature convergence. The

simulation experiments were conducted on 9 benchmark test functions. It was found that the proposed algorithm has better optimization performance.

Additionally, some work has focused on the research of hybrid schemes. Kanagaraj, Ponnambalam, Jawahar, and Mukund (2014) developed a hybrid CS and genetic algorithm (GA) for solving engineering design optimization problems. With regard to this hybrid scheme, genetic operators were used for exploitation, while Levy flight was used for exploration. After that, this hybrid algorithm was tested on 13 benchmark constrained functions and 3 engineering design problems. Liu and Fu (2015) introduced frog leaping algorithm (FLA) and chaos theory into CS to enhance the performance. First, chaos theory was used to initialize the population. Then, the inertia weight was embedded in Levy flight strategy to enhance the exploration capability. To balance the tradeoff between exploration and exploitation, the local search mechanism of FLA was employed to further strengthen the exploitation performance. To evaluate the effectiveness of the proposed algorithm, 14 benchmark problems were considered. It was seen that this hybrid technique displays higher optimization accuracy and faster convergence speed. Wang, Gandomi, Zhao, and Chu (2016) proposed a hybrid harmony search (HS) and CS for global numerical optimization. In this hybrid algorithm, the pitch adjustment operation in HS was introduced into CS as the mutation operator to accelerate convergence. Next, the performance of this hybrid scheme was tested on 14 benchmark problems. Further, the influence of parameter settings on the algorithm performance was also investigated.

With respect to these improved CS algorithms, all the work is to enhance the ability of CS to handle complex problems. It is widely accepted that the search capability of an algorithm may vary significantly from problem to problem. To cope with this issue, it is a beneficial attempt to explore an ensemble method to further strengthen the performance of CS.

2.3. Ensemble methods used in evolutionary algorithms

It is commonly believed that the best search strategy and parameter setting of the evolutionary algorithm are often different in handling distinct optimization problems. To enhance the universality and robustness of the algorithm, some studies have been done on the ensemble method.

In terms of the ensemble of differential evolution, some attempts have been made to enhance the performance. Mallipeddi, Suganthan, Pan, and Tasgetiren (2011) presented a modified DE algorithm with ensemble of parameters and mutation strategies. During the evolution process, the combination of successful strategies and parameters was selected with higher probability. Besides, the authors investigated the performance of the presented algorithm on 14 test problems. Tong, Dong, and Jing (2018) developed an improved multi-population ensemble differential evolution. To improve the convergence accuracy and speed, the authors designed a new mutation strategy, which made full use of the information of good and bad solutions to balance exploration and exploitation. Moreover, the weighted Lehmer mean scheme was embedded to alleviate premature convergence. The performance of the proposed method was tested on CEC 2005 and CEC 2017 benchmark functions. Awad, Ali, and Suganthan (2018) proposed a new variant called the ensemble sinusoidal DE with niching reduction. In this version, the sinusoidal formula and Cauchy distribution were used to strengthen the convergence performance. Then, a restart approach was employed to further improve the solution quality. Also, a niching-based reduction method was introduced to adjust the population size. Li et al. (2016) developed a hybrid framework based on CoDE and JADE, which were executed alternately on the basis of the improvement rate of the fitness. The performance of the proposed algorithm was evaluated on CEC 2014 benchmark problems. It was found that this scheme performs well on most of test functions.

Furthermore, some work has been done to apply the concept of ensemble to other evolutionary algorithms. Lynn and Suganthan (2017) proposed an ensemble particle swarm optimizer to address real-parameter optimization problems. In this scheme, a self-adaptive strategy was adopted to identify the best algorithm based on the previous experience of producing promising individuals. After that, the proposed ensemble PSO was evaluated on CEC 2005 benchmark problems and compared with other state-of-the-art algorithms. Wang et al. (2014) developed a multi-strategy ensemble artificial bee colony (ABC) algorithm. For this version, three different search strategies coexisted throughout the evolution process and competed to generate new individuals. Also, experiments were conducted on 40 test functions. It was seen that this scheme performs better than some well-established evolutionary algorithms. Xiong, Shi, and Duan (2013) introduced a multi-strategy ensemble biogeography-based optimization method for economic dispatch. To balance exploration and exploitation, the authors extended three components of biogeography-based optimization (BBO). First, the migration model on the basis of sinusoidal curve was adopted. Next, a migration operator which combined perturb operator with blended operator was developed to further enhance the exploitation capability. Finally, a mutation operator combining differential mutation and Levy local search was designed by using the backup mechanism. Vrugt, Robinson, and Hyman (2009) designed a multi-algorithm genetically adaptive scheme based on the concept of self-adaptive multiple approaches. This scheme merged the advantages of covariance matrix adaptation evolutionary strategy, GA and PSO. During the search process, a self-adaptive learning method was employed to produce more offspring.

3. Ensemble of cuckoo search variants

It is generally accepted that the search ability of an algorithm may vary significantly as the problem changes. Therefore, for a given problem, it is difficult to determine the appropriate algorithm. Practice has proved that the ensemble method is efficient and effective for online adjustment of distinct strategies or control parameters (Mallipeddi & Suganthan, 2010). As a result, to solve different types of optimization problems, this paper proposes an ensemble of multiple CS variants named the ensemble CS variant (ECSV), in which several different CS algorithms coexist throughout the evolution process and compete to generate better offspring.

3.1. Constituent algorithms

In this work, we choose three efficient CS variants as the constituent algorithms, namely CCS (Huang et al., 2016), NNCS (Wang et al., 2016) and PLCS (Yang et al., 2017). The reason is that these algorithms have diverse properties and can display different performance characteristics when dealing with different types of optimization problems. Specifically, CCS shows extraordinary performance in solving some unimodal and basic multimodal problems, NNCS works well on some basic multimodal problems, and PLCS is very effective in handling some unimodal and composition problems. Therefore, these constituent algorithms can support each other during the search process, not just compete for resources. The three CS variants are briefly described as below.

3.1.1. Chaos-enhanced cuckoo search (CCS)

CCS, developed by Huang et al., is an efficient CS variant. In CCS, chaos theory is mainly used in three aspects, namely the initial population, parameter adjustment and boundary handling.

For population initialization, five different chaotic equations such as logistic map, tent map, gauss map, sinusoidal iterator and circle map are employed to enhance the quality of the initial population distribution.

To strengthen the ability of CS algorithm to deal with different optimization problems, chaotic sequence is used to adjust the step size factor α. In case the chaotic sequence cc is larger than 0.3, the step size

Table 1
Comparison results of ECSV with different *LP* values.

Value	Term	f1	f3	f4	f6	f7	f9	f10	f14
5	Mean	**0.00E+00**	5.41E+04	2.06E−01	**6.04E−01**	3.23E−04	**9.95E−02**	6.44E+01	**1.26E+01**
10	Mean	**0.00E+00**	5.54E+04	2.47E−01	1.25E+00	8.22E−04	2.66E−01	**6.22E+01**	**1.26E+01**
15	Mean	**0.00E+00**	5.47E+04	**1.47E−01**	1.16E+00	**2.80E−04**	2.99E−01	6.39E+01	1.27E+01
20	Mean	**0.00E+00**	4.71E+04	1.49E−01	2.57E+00	1.10E−03	2.67E−01	6.47E+01	1.27E+01
25	Mean	**0.00E+00**	6.65E+04	1.63E−01	2.63E+00	4.93E−04	1.66E−01	6.97E+01	1.27E+01
30	Mean	**0.00E+00**	6.79E+04	6.36E−01	2.09E+00	7.51E−04	3.67E−01	6.53E+01	1.27E+01
35	Mean	**0.00E+00**	7.30E+04	6.21E−01	9.56E−01	4.93E−04	3.32E−01	7.31E+01	1.27E+01
40	Mean	**0.00E+00**	5.61E+04	2.81E−01	1.25E+00	1.10E−03	3.65E−01	6.25E+01	1.27E+01
45	Mean	**0.00E+00**	6.84E+04	2.55E−01	3.59E+00	5.53E−04	3.01E−01	6.72E+01	1.27E+01
50	Mean	**0.00E+00**	**3.39E+04**	4.64E−01	1.23E+00	4.93E−04	2.66E−01	7.25E+01	1.28E+01

Table 2
Comparison results of ECSV0 and ECSV on the first test suite with different variables.

Fun	Term	30 variables		50 variables	
		ECSV0	ECSV	ECSV0	ECSV
f1	Mean	**0.00E+000**	**0.00E+000**	6.73E−030	**0.00E+000**
f2	Mean	6.65E−010	**2.86E−011**	3.25E−004	**2.60E−005**
f3	Mean	1.83E+005	**5.60E+004**	1.37E+007	**9.99E+006**
f4	Mean	2.80E+000	**2.14E−001**	4.01E+003	**2.20E+003**
f5	Mean	1.19E+003	**1.14E+003**	5.79E+003	**4.93E+003**
f6	Mean	**1.21E+000**	3.50E+000	**2.61E+000**	5.25E+000
f7	Mean	**1.30E−003**	1.70E−003	5.50E−003	**8.22E−004**
f8	Mean	**2.09E+001**	**2.09E+001**	**2.11E+001**	**2.11E+001**
f9	Mean	3.66E−001	**2.66E−001**	5.73E+000	**4.20E+000**
f10	Mean	8.12E+001	**6.88E+001**	1.77E+002	**1.54E+002**
f11	Mean	**2.34E+001**	2.42E+001	**4.50E+001**	4.57E+001
f12	Mean	4.11E+003	**3.31E+003**	2.82E+004	**2.77E+004**
f13	Mean	**2.76E+000**	**2.76E+000**	5.69E+000	**5.68E+000**
f14	Mean	**1.27E+001**	**1.27E+001**	**2.22E+001**	**2.22E+001**

factor α is set to 0.3. If *cc* is less than 0.1, α is then set as 0.1. Otherwise, the step size factor is set to the chaotic sequence.

Also, in CS algorithm, the solution flying out of the boundary is set on the boundary, which is not suitable for solving the optimization problem that the optimal solution is located on the boundary. As a result, the boundary handling scheme is defined as:

$$x_{i,j}^{t+1} = Lb_j + cc\cdot(Ub_j - Lb_j) \tag{6}$$

where *cc* is the chaotic sequence, *Ub* and *Lb* are the upper and lower bounds, respectively.

3.1.2. Nearest neighbour cuckoo search (NNCS)

In the conventional CS, each individual interacts with the best solution obtained so far in the population, which accelerates the convergence rate, but may lead to the individual getting trapped in the local optimum. NNCS, proposed by Wang et al., is a nearest neighbour CS algorithm. With regard to this improved version, the nearest neighbor solution, rather than the best solution found so far, is used to guide individual search behavior.

Besides, a probabilistic mutation scheme is employed to allow individuals to learn from their nearest neighbors only in specific dimensions. The probabilistic mutation scheme is described as follows:

$$x_i^{t+1} = \begin{cases} x_i^t + r\cdot(x_i^t - x_{i,nearest}^t)\cdot Levy(\lambda) & rand < p \\ x_i^t & otherwise \end{cases} \tag{7}$$

where r and *rand* are two uniformly distributed random numbers in the range [0, 1], p represents the conversion probability, $x_{i,nearest}^t$ stands for the nearest neighbor of x_i^t, which can be selected by the solution-based and fitness-based indicators.

Table 3
Parameter settings of these algorithms.

Algorithm	Parameter settings
ACS	$p_a = 0.25$.
CCS	$\alpha = 0.1\ 0.3$, $p_a = 0.25$.
CSPSO	$\alpha_{max} = 0.5$, $\alpha_{min} = 0.01$, $p_a = 0.25$.
NNCS	$p = 0.25$, $p_a = 0.25$.
OCS	$\alpha = 0.01$, $p_a = 0.25$.
PLCS	$\alpha = 0.01$, $p_a = 0.25$, $s_p = 0.13$.
QCCS	$\alpha = 1$, $\delta = 1.6$, $p_a = 0.25$.
ECSV	$LP = 5$, $p_a = 0.25$.

3.1.3. Peer-learning cuckoo search (PLCS)

PLCS, initially presented by Yang et al., is a CS algorithm based on peer learning. In PLCS, each individual is allowed to learn from an exemplar randomly chosen among its peers. For minimization problems, the exemplar with smaller fitness is more likely to be selected. That is, individuals tend to learn from exemplars with more useful information. The selection probability is defined as:

$$p_{i,e_i} = \frac{f_{worst} - f(x_{e_i})}{f_{worst} - f_{best}} \tag{8}$$

where f_{best} and f_{worst} stand for the best and worst fitness of the exemplar candidates, respectively. $f(x_{e_i})$ is the fitness of the exemplar x_{e_i}, which is distinct from x_i.

To take advantage of the useful information of the exemplar, the peer-learning scheme is given as:

$$x_i^{t+1} = x_i^t + r\cdot\left(x_{e_i}^t - x_i^t\right) \tag{9}$$

where r is a random number between 0 and 1.

Additionally, a handover probability S_p is set to control the combination of peer-learning and Levy flight strategies. The combination scheme is given as follows:

$$x_i^{t+1} = \begin{cases} x_i^t + \alpha\cdot(x_i^t - x_{best})\cdot Levy(\lambda) & rand > S_p \\ x_i^t + r\cdot(x_{e_i}^t - x_i^t) & otherwise \end{cases} \tag{10}$$

3.2. Ensemble algorithm

As mentioned above, CCS, NNCS and PLCS are used as the constituent methods of the developed ECSV. For CCS, chaotic sequences are introduced into CS, but neither Levy flight nor biased random walk is modified. For NNCS and PLCS, the main difference between them and CS lies in Levy flight component. In ECSV, these three constituent algorithms coexist in the entire search process and compete to produce promising solutions.

According to the previous success and failure memories of the promising solutions found by each constituent algorithm, an adaptive selection scheme is employed to determine the probability of each algorithm being selected. With respect to this selection scheme, a fixed

Table 4
Comparison results of ACS, CCS, CSPSO, NNCS, OCS, PLCS, QCCS and ECSV on the first test suite with 30 variables.

Fun	Term	ACS	CCS	CSPSO	NNCS	OCS	PLCS	QCCS	ECSV
f1	Mean	6.57E−022	**0.00E+000**	7.06E−028	**0.00E+000**	1.01E−029	**0.00E+000**	**0.00E+000**	**0.00E+000**
	SD	6.52E−022	**0.00E+000**	5.62E−028	**0.00E+000**	3.84E−029	**0.00E+000**	**0.00E+000**	**0.00E+000**
f2	Mean	1.24E−004	7.90E−007	2.06E−005	2.67E−008	**2.97E−013**	2.69E−012	2.36E−011	2.86E−011
	SD	1.78E−004	1.38E−006	9.69E−005	3.81E−008	**7.54E−013**	3.36E−012	3.39E−011	4.42E−011
f3	Mean	2.08E+005	8.52E+004	1.80E+005	1.25E+005	2.55E+005	8.20E+004	1.05E+006	**5.60E+004**
	SD	2.01E+005	1.63E+005	1.94E+005	2.84E+005	3.08E+005	1.14E+005	9.79E+005	**9.47E+004**
f4	Mean	1.66E+002	7.83E+000	1.01E+001	8.96E+000	1.60E+002	4.43E−001	3.76E+002	**2.14E−001**
	SD	1.79E+002	1.68E+001	2.05E+001	1.92E+001	2.08E+002	4.58E−001	4.26E+002	**2.97E−001**
f5	Mean	2.03E+003	**6.32E+002**	1.07E+003	1.59E+003	4.09E+003	1.39E+003	3.66E+003	1.14E+003
	SD	7.76E+002	5.39E+002	**4.52E+002**	4.69E+002	1.84E+003	5.49E+002	1.10E+003	5.29E+002
f6	Mean	2.57E+001	2.50E+001	2.82E+000	1.47E+001	**6.65E−001**	2.70E+000	3.53E+000	3.50E+000
	SD	2.38E+001	2.93E+001	3.64E+000	1.59E+001	**1.51E+000**	2.52E+000	1.34E+001	1.26E+001
f7	Mean	2.20E−003	4.40E−003	1.43E−002	**1.89E−007**	1.35E−002	8.30E−003	1.58E−002	1.70E−003
	SD	4.50E−003	8.10E−003	1.35E−002	**9.51E−007**	1.48E−002	9.30E−003	1.35E−002	5.60E−003
f8	Mean	2.10E+001	2.09E+001	2.09E+001	2.09E+001	**2.08E+001**	2.09E+001	2.09E+001	2.09E+001
	SD	5.20E−002	7.02E−002	**4.08E−002**	4.79E−002	7.09E−002	5.61E−002	5.05E−002	5.93E−002
f9	Mean	3.15E+001	3.68E+001	4.16E+001	1.09E+001	2.07E+001	5.07E+000	4.37E+001	**2.66E−001**
	SD	8.30E+000	1.17E+001	1.07E+001	3.47E+000	6.96E+000	2.56E+000	1.16E+001	**4.48E−001**
f10	Mean	**6.32E+001**	8.78E+001	8.18E+001	7.66E+001	1.08E+002	9.86E+001	7.03E+001	6.88E+001
	SD	1.74E+001	2.55E+001	1.78E+001	1.57E+001	2.86E+001	1.90E+001	1.91E+001	**1.42E+001**
f11	Mean	2.42E+001	**2.11E+001**	2.57E+001	2.41E+001	2.46E+001	2.42E+001	2.66E+001	2.42E+001
	SD	3.60E+000	4.69E+000	2.94E+000	**2.46E+000**	3.30E+000	2.74E+000	3.98E+000	2.89E+000
f12	Mean	4.86E+003	7.19E+003	4.38E+003	3.41E+003	5.96E+003	3.70E+003	7.75E+003	**3.31E+003**
	SD	4.39E+003	6.33E+003	4.29E+003	3.87E+003	5.81E+003	3.47E+003	9.11E+003	**3.25E+003**
f13	Mean	4.90E+000	4.88E+000	3.55E+000	3.50E+000	3.59E+000	**2.49E+000**	3.04E+000	2.76E+000
	SD	2.30E+000	2.05E+000	1.22E+000	7.90E−001	9.68E−001	**4.33E−001**	7.13E−001	6.38E−001
f14	Mean	1.30E+001	1.28E+001	**1.24E+001**	1.26E+001	1.26E+001	1.26E+001	1.25E+001	1.27E+001
	SD	2.31E−001	3.53E−001	4.41E−001	2.62E−001	3.59E−001	2.71E−001	4.00E−001	**2.06E−001**
	ARV	5.8214	4.7857	4.7857	3.7857	5.0000	3.3929	5.6786	**2.7500**
	Rank	8	4	4	3	6	2	7	**1**

The best results, i.e., the lowest mean and standard deviation (SD), are shown in bold.

number of iterations named learning period (*LP*) is defined in advance, and each algorithm has the same probability of being selected at the beginning of the iteration. As the iteration proceeds, the success and failure memories are separately recorded in each learning period. Then, for each constituent algorithm, the success rate is computed to update the selection probability in the following generations. Also, the constituent algorithm is selected in terms of probability matching to generate new candidate solutions. In other words, at subsequent generations, the algorithm with higher success rate is considered to be more suitable for finding the global optimal solution, so it has a higher probability of being selected to produce more offspring. It should be noted that the process is performed periodically with *LP* being the period.

After *LP* generations, the probability of selecting each constituent algorithm is defined as:

$$p_k^t = \frac{SR_k^t}{\sum_{k=1}^{K} SR_k^t} \tag{11}$$

$$SR_k^t = \frac{\sum_{l=t-LP}^{t-1} sn_k^l}{\sum_{l=t-LP}^{t-1} sn_k^l + \sum_{l=t-LP}^{t-1} fn_k^l} + \varepsilon \tag{12}$$

where $t > LP$, p_k^t represents the selection probability of the kth constituent algorithm, K is the number of constituent algorithms, SR_k^t represents the success rate of the kth constituent algorithm, sn stands for the number of success memories, fn denotes the number of failure memories, ε is set as 0.01 to avoid the null success rate during the evolutionary process.

Moreover, for these constituent CS algorithms, the biased random walk strategy is not modified. In the original scheme, two solutions are randomly selected from the current population. Due to the stochastic nature, the wrong moving direction may be provided in some cases. To further enhance the convergence performance, an external archive A is introduced into ECSV to store the current solution defeated by the new solution in the selection phase. Note that, the external archive is initiated to be empty. Let P represent the current population. The biased random walk is modified as follows:

$$x_i^{t+1} = x_i^t + r \cdot (x_{r1}^t - \hat{x}_{r2}^t) \tag{13}$$

where x_{r1}^t is a random solution chosen from the current population, $\hat{x}_{r2}^t$ represents a random solution chosen from the union of P and A.

With the increasing of generations, more and more individuals will be stored in the external archive. In general, the larger the archive size, the better the population diversity. Nevertheless, too large archive size may reduce the convergence rate of the algorithm. Therefore, the maximum size of the external archive is set as the population size N. In case the archive size exceeds N, the redundant solutions are randomly removed from A.

Table 5
Comparison results of ACS, CCS, CSPSO, NNCS, OCS, PLCS, QCCS and ECSV on the first test suite with 50 variables.

Fun	Term	ACS	CCS	CSPSO	NNCS	OCS	PLCS	QCCS	ECSV
f1	Mean	1.56E−018	**0.00E+000**	2.07E−027	**0.00E+000**	1.78E−028	3.20E−029	6.73E−030	**0.00E+000**
	SD	1.30E−018	**0.00E+000**	1.23E−027	**0.00E+000**	1.55E−028	7.44E−029	3.69E−029	**0.00E+000**
f2	Mean	1.98E+002	5.65E−002	1.40E−003	7.10E−003	6.94E−005	**1.50E−005**	1.78E−004	2.60E−005
	SD	1.07E+003	6.23E−002	2.50E−003	6.80E−003	9.60E−005	**1.90E−005**	2.60E−004	2.27E−005
f3	Mean	1.64E+007	**1.33E+006**	1.41E+007	1.06E+007	1.99E+007	7.85E+006	2.03E+007	9.99E+006
	SD	6.89E+006	**6.75E+005**	7.65E+006	4.83E+006	1.16E+007	4.41E+006	2.97E+007	3.95E+006
f4	Mean	1.86E+004	7.92E+003	1.02E+004	4.09E+003	1.32E+004	2.27E+003	1.75E+004	**2.20E+003**
	SD	6.17E+003	4.36E+003	5.08E+003	1.71E+003	5.75E+003	1.29E+003	5.51E+003	**1.24E+003**
f5	Mean	7.44E+003	**4.00E+003**	**4.00E+003**	5.66E+003	9.20E+003	5.24E+003	1.17E+004	4.93E+003
	SD	1.38E+003	9.60E+002	**7.78E+002**	1.12E+003	1.94E+003	9.17E+002	2.56E+003	1.18E+003
f6	Mean	4.62E+001	5.43E+001	1.05E+001	4.24E+001	7.69E+000	9.42E+000	1.53E+001	**5.25E+000**
	SD	2.10E+001	3.14E+001	2.38E+001	2.62E+001	**1.36E+001**	1.50E+001	4.11E+001	1.41E+001
f7	Mean	4.70E−003	1.80E−003	1.43E−002	2.50E−003	1.14E−002	9.70E−003	8.30E−003	**8.22E−004**
	SD	8.10E−003	6.40E−003	1.24E−002	6.00E−003	1.03E−002	1.40E−002	1.12E−002	**2.50E−003**
f8	Mean	2.11E+001	2.11E+001	2.11E+001	2.11E+001	**2.10E+001**	2.11E+001	2.11E+001	2.11E+001
	SD	4.95E−002	**2.64E−002**	4.32E−002	3.66E−002	9.39E−002	3.75E−002	4.34E−002	3.74E−002
f9	Mean	8.11E+001	7.69E+001	9.96E+001	3.91E+001	3.89E+001	2.95E+001	9.83E+001	**4.20E+000**
	SD	2.34E+001	1.98E+001	2.34E+001	1.08E+001	1.21E+001	7.92E+000	2.70E+001	**1.93E+000**
f10	Mean	**1.24E+002**	1.93E+002	1.81E+002	1.79E+002	2.95E+002	2.54E+002	2.13E+002	1.54E+002
	SD	2.46E+001	4.37E+001	4.51E+001	3.08E+001	5.32E+001	4.19E+001	3.94E+001	**2.22E+001**
f11	Mean	4.26E+001	**3.61E+001**	5.17E+001	4.52E+001	5.20E+001	4.72E+001	5.22E+001	4.57E+001
	SD	9.09E+000	6.59E+000	5.12E+000	5.38E+000	5.32E+000	**4.40E+000**	6.83E+000	5.95E+000
f12	Mean	8.70E+004	4.38E+004	7.37E+004	3.04E+004	1.35E+005	**1.97E+004**	4.75E+004	2.77E+004
	SD	4.65E+004	3.19E+004	4.55E+004	2.71E+004	1.52E+005	**1.53E+004**	8.68E+004	2.53E+004
f13	Mean	7.35E+000	7.29E+000	7.29E+000	6.92E+000	6.64E+000	**5.66E+000**	7.45E+000	5.68E+000
	SD	2.51E+000	1.92E+000	1.70E+000	1.94E+000	1.28E+000	**1.16E+000**	2.49E+000	1.28E+000
f14	Mean	2.24E+001	2.25E+001	2.19E+001	2.22E+001	2.18E+001	2.22E+001	**2.17E+001**	2.22E+001
	SD	4.12E−001	2.99E−001	4.42E−001	3.61E−001	5.84E−001	**2.44E−001**	7.09E−001	3.25E−001
	ARV	5.6071	4.3929	5.0357	4.1786	4.8929	3.4643	5.8214	**2.6071**
	Rank	7	4	6	3	5	2	8	**1**

The best results, i.e., the lowest mean and standard deviation (SD), are shown in bold.

The framework of the proposed ECSV algorithm is given in Algorithm 1.

Algorithm 1 (*The proposed ECSV algorithm*).

Initialize the necessary parameters, including the population size N problem dimension D, maximum number of generations t_{max}, discovery probability p_a and learning period LP;
Initialize the parameters of CCS, NNCS and PLCS;
Randomly initialize the solution x_i and evaluate its fitness $f(x_i)$;
$t = 0$, $A = \phi$;
while $t \le t_{max}$ **do**
  $t = t + 1$;
  for $i = 1$ to N **do**
    if $t \le 1$ **then**
      $p = [1/3, 1/3, 1/3]$, $rk = [0, 1/3, 2/3, 1]$, $sn = 0$ and $fn = 0$;
    else if $mod(t, LP) = =0$ **then**
      Calculate the selection probability p using Eq. (11);
      $rk = [0, cumsum(p)]$, $sn = 0$ and $fn = 0$;
    end if
    Generate a random number $rand$, and $prob = rand$;
    if $prob \ge rk(1)\&prob < rk(2)$ **then**
      Produce the new solution $x_{i,new}$ by executing CCS, and $k = 1$;
    else if $prob \ge rk(2)\&prob < rk(3)$ **then**
      Produce the new solution $x_{i,new}$ by executing NNCS, and $k = 2$;
    else if $prob \ge rk(3)\&prob < rk(4)$ **then**
      Produce the new solution $x_{i,new}$ by executing PLCS, and $k = 3$;
    end if
    Calculate the fitness $f(x_{i,new})$;
    if $f(x_{i,new}) < f(x_i)$ **then**
      $x_i = x_{i,new}$, $f(x_i) = f(x_{i,new})$ and $sn_k = sn_k + 1$;
    else
      $fn_k = fn_k + 1$;
    end if
    if $rand(0, 1) > p_a$ **then**
      Produce the new solution $x_{i,new}$ using Eq. (13) and calculate its fitness $f(x_{i,new})$;
      if $f(x_{i,new}) < f(x_i)$ **then**
        $x_i \to A$, $x_i = x_{i,new}$ and $f(x_i) = f(x_{i,new})$;
      end if
    end if
  end for
  Record the best solution;
  if $size(A) > N$ **then**
    $M = \text{size}(A) - N$;
    Remove M solutions from A randomly;
  end if
end while

4. Experimental results

4.1. Test problems

To fully investigate the efficiency of the proposed ECSV algorithm, 42 popular benchmark problems from two different test suites are considered. The first test suite consists of the front 14 problems in CEC 2005 competition on real-parameter optimization, and the second test suite includes 28 problems proposed in CEC 2013. These test functions span a diverse set of properties, such as unimodality, multimodality, separation and non-separation. Specifically, with respect to the first test suite, f1 – f5 are unimodal problems, f6 – f12 are basic multimodal

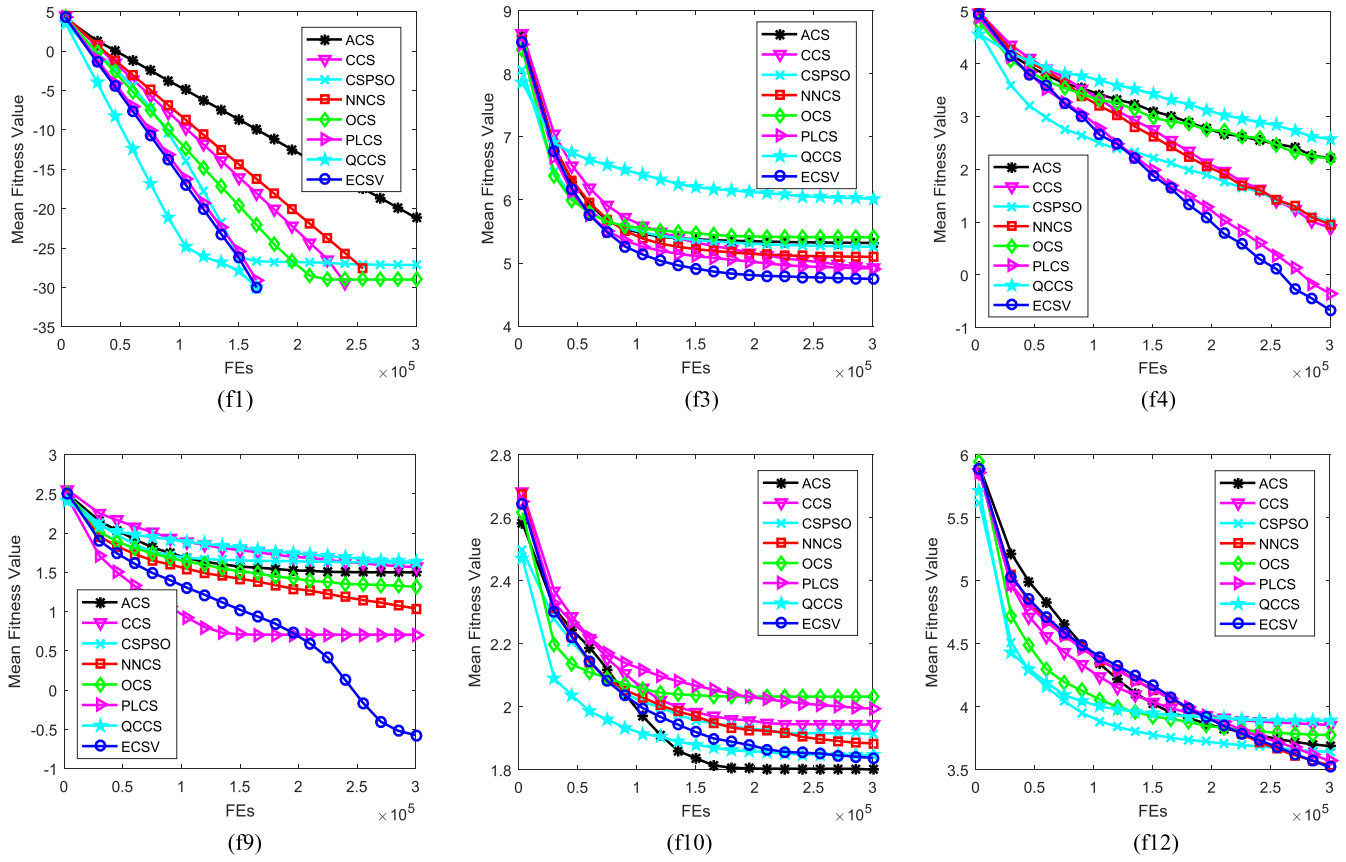

Fig. 1. Convergence graphs of ACS, CCS, CSPSO, NNCS, OCS, PLCS, QCCS and ECSV on the first test suite with 30 variables.

problems and f13 – f14 are expanded multimodal functions. For the second test suite, F1 – F5 are unimodal problems, F6 – F20 are basic multimodal problems and F21 – F28 are composition problems. It should be noted that all functions should be treated as black-box problems. Furthermore, these test functions are complex shifted and rotated problems, which can make the experimental results more convincing.

4.2. Influence of the learning period

In the proposed ECSV algorithm, two important parameters, namely the learning period LP and discovery probability p_a, need to be preset. With respect to the discovery probability, it is set as 0.25 according to the original CS algorithm. To investigate the effect of learning period on the performance of ECSV, some experiments are conducted on several functions in the first test suite with 30 decision variable. In this subsection, the population size N is set to be 50, the learning period LP is set from 5 to 50 with the step size of 5, and the maximal function evaluation is set to 300,000 for each test problem. Moreover, for each function, 30 independent trials are run for each value of the learning period. The mean errors are recorded for comparison, as shown in Table 1, where the best result is highlighted in bold. In this paper, all the experiments are conducted in MATLAB R2016a on a computer (Intel i7-4790 CPU, 8.00 GB RAM and Windows 7 system).

From Table 1, it can be observed that the convergence performance of ECSV is less sensitive to the setting of the learning period LP. In terms of solution quality, $LP = 5$ provides reasonable results on f1, f6, f9 and f14, and it is the second best on f7. $LP = 10$ is competitive in solving f1, f10 and f14, $LP = 15$ does well in tackling f1, f4 and f7, and $LP = 50$ outperforms others on f3. Also, for the problem f1, ECSV can converge to the global optimal solution when using these different LP values. According to the above results, the learning period LP is suggested to set as 5 in the following experiments.

4.3. Performance enhancement using the external archive

As discussed above, the external archive A is embedded into the ensemble algorithm. To illustrate the performance enhancement brought about by the external archive, the ensemble scheme without introducing the archive A is named ECSV0. For the purpose of comparison, experiments are conducted on the first test suite. For these test problems with 30 and 50 decision variables, the maximal function evaluation is set as 300,000 and 500,000, respectively. Also, with respect to the two ensemble algorithms, the population size N is set to 50, the learning period LP is set as 5, and the discovery probability p_a is equal to 0.25. For each algorithm, 30 independent runs are executed for each problem. Table 2 presents the comparison results of these two algorithms, and the lowest mean error is shown in bold.

As shown in Table 2, with respect to these test problems with 30 variables, ECSV surpasses ECSV0 on f2, f3, f4, f5, f9, f10 and f12, and they get the same results on f1, f8, f13 and f14. Similarly, for these problems with 50 variables, ECSV performs best on f1, f2, f3, f4, f5, f7, f9, f10, f12 and f13 in terms of solution quality, and these two algorithms obtain similar solutions on f8 and f14. Also, ECSV is inferior to ECSV0 on f6 and f11 in tackling the first test suite. Regarding the problem f6, there is a very narrow valley from local optimum to global optimum, while f11 is a continuous function, but it is differentiable only on a set of points. Therefore, in solving these two kinds of problems, the algorithm may need to focus on exploitation capability for fine search. In a word, ECSV is superior to ECSV0 on most cases, which means that the introduction of the external archive is conducive to improving the solution quality. In subsequent experiments, we only test the

Table 6
Comparison results of ACS, CCS, CSPSO, NNCS, OCS, PLCS, QCCS and ECSV on the second test suite with 30 variables.

Fun	Term	ACS	CCS	CSPSO	NNCS	OCS	PLCS	QCCS	ECSV
F1	Mean	1.14E−013	1.21E−013	3.94E−013	1.36E−013	3.71E−013	2.35E−013	2.88E−013	**6.82E−014**
	SD	1.16E−013	1.15E−013	1.18E−013	1.13E−013	1.14E−011	**4.15E−014**	1.02E−013	1.06E−013
F2	Mean	7.89E+004	9.44E+004	7.83E+004	2.08E+003	5.22E+002	1.12E+003	2.18E+005	**3.30E+002**
	SD	6.88E+004	9.92E+004	7.06E+004	3.00E+003	1.12E+003	3.00E+003	1.40E+005	**1.07E+003**
F3	Mean	1.93E+006	2.57E+004	2.16E+006	6.85E+005	3.65E+005	1.12E+005	1.35E+007	**7.51E+003**
	SD	4.73E+006	1.08E+005	6.27E+006	1.48E+006	1.01E+006	4.26E+005	1.94E+007	**3.63E+004**
F4	Mean	2.53E−002	2.71E−002	1.10E+002	1.11E−001	2.36E−001	2.30E−002	2.14E+001	**8.80E−003**
	SD	3.26E−002	3.58E−002	1.33E+002	9.97E−002	3.14E−001	**2.37E−002**	5.96E+001	4.20E−002
F5	Mean	3.93E−012	1.21E−013	6.03E−013	**1.14E−013**	4.05E−013	2.69E−013	2.99E−013	1.17E−013
	SD	2.95E−012	2.88E−014	1.47E−013	**0.00E+000**	8.80E−014	1.14E−013	8.17E−014	2.08E−014
F6	Mean	1.67E+001	3.61E+000	1.29E+001	2.85E+000	7.23E+000	4.19E+000	4.16E+000	**1.86E+000**
	SD	1.56E+001	4.93E+000	6.75E+000	4.63E+000	1.18E+001	**4.62E+000**	8.91E+000	6.68E+000
F7	Mean	4.27E+001	**2.94E+001**	4.45E+001	7.03E+001	9.97E+001	6.69E+001	8.39E+001	4.95E+001
	SD	1.63E+001	**1.42E+001**	1.99E+001	1.49E+001	3.33E+001	1.91E+001	4.28E+001	1.50E+001
F8	Mean	2.09E+001	2.09E+001	2.09E+001	2.09E+001	2.09E+001	2.09E+001	2.09E+001	2.09E+001
	SD	5.75E−002	4.65E−002	4.65E−002	4.55E−002	**3.74E−002**	5.36E−002	4.23E−002	4.50E−002
F9	Mean	2.37E+001	**2.23E+001**	2.78E+001	2.42E+001	2.55E+001	2.56E+001	2.59E+001	2.56E+001
	SD	3.14E+000	3.87E+000	3.03E+000	2.12E+000	2.68E+000	1.92E+000	3.70E+000	**1.82E+000**
F10	Mean	1.66E−002	1.83E−002	3.00E−002	**6.90E−003**	2.75E−002	1.79E−002	5.75E−002	**6.90E−003**
	SD	1.52E−002	1.59E−002	2.05E−002	9.70E−003	2.05E−002	1.42E−002	4.05E−002	**6.30E−003**
F11	Mean	3.16E+001	3.56E+001	5.02E+001	1.35E+001	1.87E+001	5.07E+000	4.59E+001	**6.63E−002**
	SD	9.65E+000	1.35E+001	1.53E+001	3.18E+000	6.78E+000	3.25E+000	1.69E+001	**2.52E−001**
F12	Mean	**5.01E+001**	7.02E+001	7.94E+001	6.39E+001	1.30E+002	9.61E+001	9.63E+001	5.55E+001
	SD	1.97E+001	1.81E+001	2.14E+001	1.44E+001	4.00E+001	2.08E+001	2.72E+001	**1.35E+001**
F13	Mean	**9.51E+001**	1.19E+002	1.36E+002	1.07E+002	1.91E+002	1.43E+002	1.50E+002	9.76E+001
	SD	**1.87E+001**	2.83E+001	3.09E+001	2.54E+001	4.86E+001	2.53E+001	2.89E+001	2.37E+001
F14	Mean	3.37E+003	2.06E+003	3.71E+003	1.12E+003	1.43E+003	**3.26E+002**	3.71E+003	6.08E+002
	SD	1.17E+003	7.19E+002	1.25E+003	2.21E+002	3.72E+002	**1.26E+002**	7.09E+002	2.27E+002
F15	Mean	4.53E+003	4.43E+003	5.06E+003	4.07E+003	**3.46E+003**	4.16E+003	4.33E+003	4.04E+003
	SD	9.28E+002	6.96E+002	9.64E+002	5.29E+002	**3.97E+002**	4.87E+002	1.08E+003	5.92E+002
F16	Mean	2.09E+000	2.01E+000	2.26E+000	1.57E+000	**1.05E+000**	1.91E+000	2.08E+000	1.78E+000
	SD	4.18E−001	4.31E−001	**2.41E−001**	4.17E−001	2.45E−001	3.12E−001	5.05E−001	2.86E−001
F17	Mean	8.73E+001	1.04E+002	8.32E+001	6.77E+001	9.20E+001	**4.88E+001**	1.27E+002	5.12E+001
	SD	3.04E+001	3.66E+001	1.52E+001	6.56E+000	1.72E+001	**4.75E+000**	2.39E+001	5.49E+000
F18	Mean	1.26E+002	1.27E+002	1.47E+002	**1.07E+002**	1.48E+002	1.47E+002	1.28E+002	1.20E+002
	SD	3.46E+001	4.01E+001	3.61E+001	**1.98E+001**	2.86E+001	2.00E+001	3.20E+001	2.23E+001
F19	Mean	4.47E+000	4.66E+000	3.85E+000	3.63E+000	4.14E+000	4.71E+000	5.77E+000	**3.06E+000**
	SD	1.93E+000	2.02E+000	8.70E−001	**7.24E−001**	1.43E+000	2.06E+000	2.12E+000	7.57E−001
F20	Mean	1.27E+001	1.21E+001	**1.14E+001**	1.21E+001	1.25E+001	1.17E+001	1.21E+001	1.24E+001
	SD	4.13E−001	7.15E−001	5.93E−001	**3.80E−001**	1.14E+000	4.92E−001	7.14E−001	5.65E−001
F21	Mean	3.04E+002	3.71E+002	3.49E+002	**2.84E+002**	3.09E+002	3.02E+002	3.86E+002	3.81E+002
	SD	7.49E+001	8.86E+001	9.03E+001	8.61E+001	7.91E+001	**6.77E+001**	8.75E+001	7.23E+001
F22	Mean	3.36E+003	2.04E+003	2.87E+003	1.27E+003	1.45E+003	**2.37E+002**	3.64E+003	7.55E+002
	SD	9.93E+002	1.12E+003	1.13E+003	3.26E+002	3.86E+002	**1.19E+002**	1.05E+003	2.44E+002
F23	Mean	4.78E+003	4.65E+003	5.54E+003	4.72E+003	**4.20E+003**	4.83E+003	4.92E+003	4.91E+003
	SD	1.07E+003	8.48E+002	7.93E+002	5.53E+002	**4.70E+002**	5.40E+002	9.28E+002	6.52E+002
F24	Mean	2.62E+002	**2.30E+002**	2.90E+002	2.70E+002	2.79E+002	2.59E+002	2.81E+002	2.69E+002
	SD	1.24E+001	1.33E+001	**7.55E+000**	9.81E+000	8.63E+000	1.20E+001	1.38E+001	9.46E+000
F25	Mean	**2.68E+002**	2.76E+002	2.96E+002	2.85E+002	2.89E+002	2.86E+002	2.99E+002	2.81E+002
	SD	7.88E+000	1.09E+001	**4.64E+000**	5.96E+000	6.77E+000	7.44E+000	1.36E+001	7.22E+000
F26	Mean	**2.00E+002**	2.10E+002	2.06E+002	**2.00E+002**	2.10E+002	**2.00E+002**	3.34E+002	**2.00E+002**
	SD	7.90E−003	3.76E+001	3.00E+001	4.43E−004	3.94E+001	6.38E−004	6.16E+001	**2.60E−004**
F27	Mean	8.95E+002	**7.55E+002**	1.10E+003	8.81E+002	1.04E+003	9.37E+002	1.04E+003	9.04E+002
	SD	1.18E+002	1.57E+002	8.44E+001	2.25E+002	**7.41E+001**	9.34E+001	1.48E+002	1.57E+002
F28	Mean	3.00E+002	3.37E+002	**2.60E+002**	3.00E+002	4.17E+002	3.00E+002	5.24E+002	3.00E+002
	SD	3.10E−009	2.03E+002	8.14E+001	**1.79E−013**	3.58E+002	2.11E−013	5.09E+002	2.31E−013
	ARV	4.2321	4.1786	5.8036	3.2500	5.0536	3.8036	6.6964	**2.9821**
	Rank	5	4	7	2	6	3	8	**1**

The best results, i.e., the lowest mean and standard deviation (SD), are shown in bold.

Table 7
Comparison results of ACS, CCS, CSPSO, NNCS, OCS, PLCS, QCCS and ECSV on the second test suite with 50 variables.

Fun	Term	ACS	CCS	CSPSO	NNCS	OCS	PLCS	QCCS	ECSV
F1	Mean	2.80E−013	2.50E−013	8.49E−013	2.58E−013	7.65E−013	5.23E−013	6.90E−013	**2.27E−013**
	SD	9.78E−014	6.94E−014	2.23E−013	7.86E−014	1.26E−013	1.22E−013	1.52E−013	**0.00E+000**
F2	Mean	2.58E+006	1.10E+006	1.73E+006	3.21E+005	**1.38E+005**	1.94E+005	4.99E+005	1.97E+005
	SD	1.83E+006	5.34E+005	7.69E+005	1.34E+005	**6.87E+004**	8.48E+004	1.95E+005	8.69E+004
F3	Mean	9.59E+007	2.53E+006	4.88E+007	3.30E+006	2.94E+007	3.27E+005	1.89E+008	**1.55E+005**
	SD	1.75E+008	3.99E+006	7.10E+007	7.78E+006	7.49E+007	6.67E+005	2.10E+008	**3.23E+005**
F4	Mean	1.42E+000	1.33E+000	1.54E+003	7.61E+000	2.22E+001	2.07E+000	3.79E+001	**7.05E−001**
	SD	9.42E−001	1.32E+000	5.82E+002	4.55E+000	1.56E+001	1.61E+000	4.90E+001	**5.08E−001**
F5	Mean	1.44E−010	2.96E−013	1.55E−012	2.92E−013	7.54E−013	5.99E−013	6.78E−013	**2.31E−013**
	SD	1.12E−010	8.23E−014	5.67E−013	5.73E−014	1.21E−013	1.52E−013	1.45E−013	**5.57E−014**
F6	Mean	4.36E+001	4.18E+001	**3.61E+001**	4.36E+001	4.28E+001	3.82E+001	4.12E+001	4.36E+001
	SD	**1.04E+000**	8.08E+000	1.67E+001	**1.04E+000**	1.64E+001	1.24E+001	1.99E+001	**1.04E+000**
F7	Mean	8.92E+001	6.79E+001	**6.03E+001**	8.97E+001	1.00E+002	8.13E+001	9.74E+001	9.18E+001
	SD	1.83E+001	1.52E+001	1.74E+001	1.22E+001	1.81E+001	**9.70E+000**	1.97E+001	1.09E+001
F8	Mean	2.11E+001	2.11E+001	2.11E+001	2.11E+001	2.11E+001	2.11E+001	2.11E+001	2.11E+001
	SD	3.58E−002	4.98E−002	4.19E−002	**3.11E−002**	3.59E−002	4.52E−002	3.55E−002	5.14E−002
F9	Mean	4.33E+001	**3.85E+001**	5.49E+001	4.81E+001	5.30E+001	4.95E+001	4.73E+001	4.96E+001
	SD	6.73E+000	6.09E+000	4.38E+000	5.83E+000	4.28E+000	**3.79E+000**	5.31E+000	4.51E+000
F10	Mean	3.00E−002	3.00E−002	3.51E−002	2.78E−002	3.78E−002	7.32E−002	5.24E−002	**2.64E−002**
	SD	1.82E−002	2.36E−002	2.65E−002	2.59E−002	2.18E−002	5.19E−002	4.41E−002	**1.44E−002**
F11	Mean	7.37E+001	6.19E+001	1.24E+002	4.82E+001	3.76E+001	3.66E+001	1.24E+002	**2.30E+000**
	SD	1.58E+001	2.04E+001	2.75E+001	1.72E+001	1.47E+001	1.26E+001	3.15E+001	**1.37E+000**
F12	Mean	**1.09E+002**	1.63E+002	1.95E+002	1.60E+002	3.16E+002	2.07E+002	2.71E+002	1.41E+002
	SD	**1.44E+001**	4.41E+001	4.49E+001	2.99E+001	5.03E+001	3.04E+001	5.32E+001	2.83E+001
F13	Mean	**2.24E+002**	2.65E+002	3.01E+002	2.55E+002	4.47E+002	3.21E+002	4.02E+002	2.43E+002
	SD	4.47E+001	4.98E+001	5.53E+001	3.93E+001	6.65E+001	3.82E+001	6.89E+001	**3.70E+001**
F13	Mean	5.68E+003	3.11E+003	7.47E+003	2.69E+003	2.30E+003	**6.48E+002**	6.87E+003	1.39E+003
	SD	1.69E+003	1.59E+003	2.17E+003	8.12E+002	5.70E+002	**2.56E+002**	9.82E+002	5.91E+002
F15	Mean	8.16E+003	7.47E+003	9.39E+003	8.02E+003	**6.99E+003**	8.24E+003	8.55E+003	7.81E+003
	SD	1.69E+003	9.51E+002	1.74E+003	1.01E+003	**7.69E+002**	1.11E+003	1.21E+003	1.18E+003
F16	Mean	2.79E+000	2.62E+000	3.07E+000	2.37E+000	**1.29E+000**	2.70E+000	2.85E+000	2.44E+000
	SD	4.51E−001	5.93E−001	**2.52E−001**	3.26E−001	4.43E−001	3.46E−001	4.57E−001	3.89E−001
F17	Mean	1.56E+002	1.77E+002	1.68E+002	1.52E+002	1.77E+002	1.24E+002	2.65E+002	**1.22E+002**
	SD	4.95E+001	4.96E+001	1.90E+001	2.63E+001	2.62E+001	**1.38E+001**	5.92E+001	1.79E+001
F18	Mean	2.11E+002	**1.92E+002**	2.33E+002	2.00E+002	3.34E+002	3.07E+002	2.98E+002	2.07E+002
	SD	4.61E+001	5.47E+001	7.19E+001	4.63E+001	5.20E+001	**3.84E+001**	7.40E+001	4.63E+001
F19	Mean	9.92E+000	8.91E+000	8.33E+000	7.15E+000	9.14E+000	2.74E+001	1.59E+001	**6.69E+000**
	SD	3.12E+000	2.87E+000	**1.80E+000**	1.85E+000	2.20E+000	6.57E+000	4.79E+000	1.89E+000
F20	Mean	2.23E+001	2.18E+001	**2.04E+001**	2.12E+001	2.23E+001	2.10E+001	2.16E+001	2.18E+001
	SD	8.08E−001	7.55E−001	8.81E−001	9.22E−001	1.23E+000	7.56E−001	1.06E+000	**5.28E−001**
F21	Mean	7.06E+002	6.27E+002	8.94E+002	**2.94E+002**	7.69E+002	7.79E+002	5.12E+002	3.44E+002
	SD	4.32E+002	3.95E+002	3.08E+002	**2.49E+002**	3.97E+002	4.02E+002	3.99E+002	3.31E+002
F22	Mean	5.18E+003	3.54E+003	7.27E+003	3.10E+003	2.53E+003	**2.65E+002**	7.49E+003	1.83E+003
	SD	1.68E+003	1.36E+003	1.91E+003	6.85E+002	5.53E+002	**1.11E+002**	1.26E+003	8.69E+002
F23	Mean	9.03E+003	8.63E+003	1.02E+004	8.71E+003	**8.06E+003**	9.69E+003	8.19E+003	8.95E+003
	SD	1.29E+003	1.32E+003	1.84E+003	1.08E+003	**8.56E+002**	9.28E+002	1.38E+003	1.18E+003
F24	Mean	3.14E+002	**2.74E+002**	3.77E+002	3.28E+002	3.56E+002	3.13E+002	3.57E+002	3.16E+002
	SD	1.71E+001	1.76E+001	**4.39E+000**	1.35E+001	1.21E+001	1.61E+001	2.03E+001	1.62E+001
F25	Mean	**3.35E+002**	3.41E+002	3.81E+002	3.65E+002	3.81E+002	3.64E+002	4.08E+002	3.57E+002
	SD	2.03E+001	1.71E+001	**4.70E+000**	9.62E+000	1.09E+001	1.44E+001	2.34E+001	1.45E+001
F26	Mean	2.64E+002	2.07E+002	2.56E+002	**2.00E+002**	2.41E+002	**2.00E+002**	4.34E+002	**2.00E+002**
	SD	9.96E+001	3.80E+001	1.02E+002	1.41E−002	9.26E+001	**8.50E−003**	1.27E+001	1.46E−002
F27	Mean	1.44E+003	**1.32E+003**	2.06E+003	1.61E+003	1.75E+003	1.52E+003	1.72E+003	1.57E+003
	SD	1.71E+002	1.60E+002	**8.59E+001**	1.22E+002	9.81E+001	1.54E+002	2.02E+002	1.26E+002
F28	Mean	5.04E+002	5.01E+002	1.02E+003	**4.00E+002**	9.44E+002	**4.00E+002**	1.38E+003	**4.00E+002**
	SD	5.69E+002	5.51E+002	1.26E+003	**1.68E−013**	1.24E+003	2.73E−013	1.65E+003	2.35E−013
	ARV	4.6250	3.5714	5.8393	3.6607	5.1071	4.0179	6.1071	**3.0714**
	Rank	5	2	7	3	6	4	8	**1**

The best results, i.e., the lowest mean and standard deviation (SD), are shown in bold.

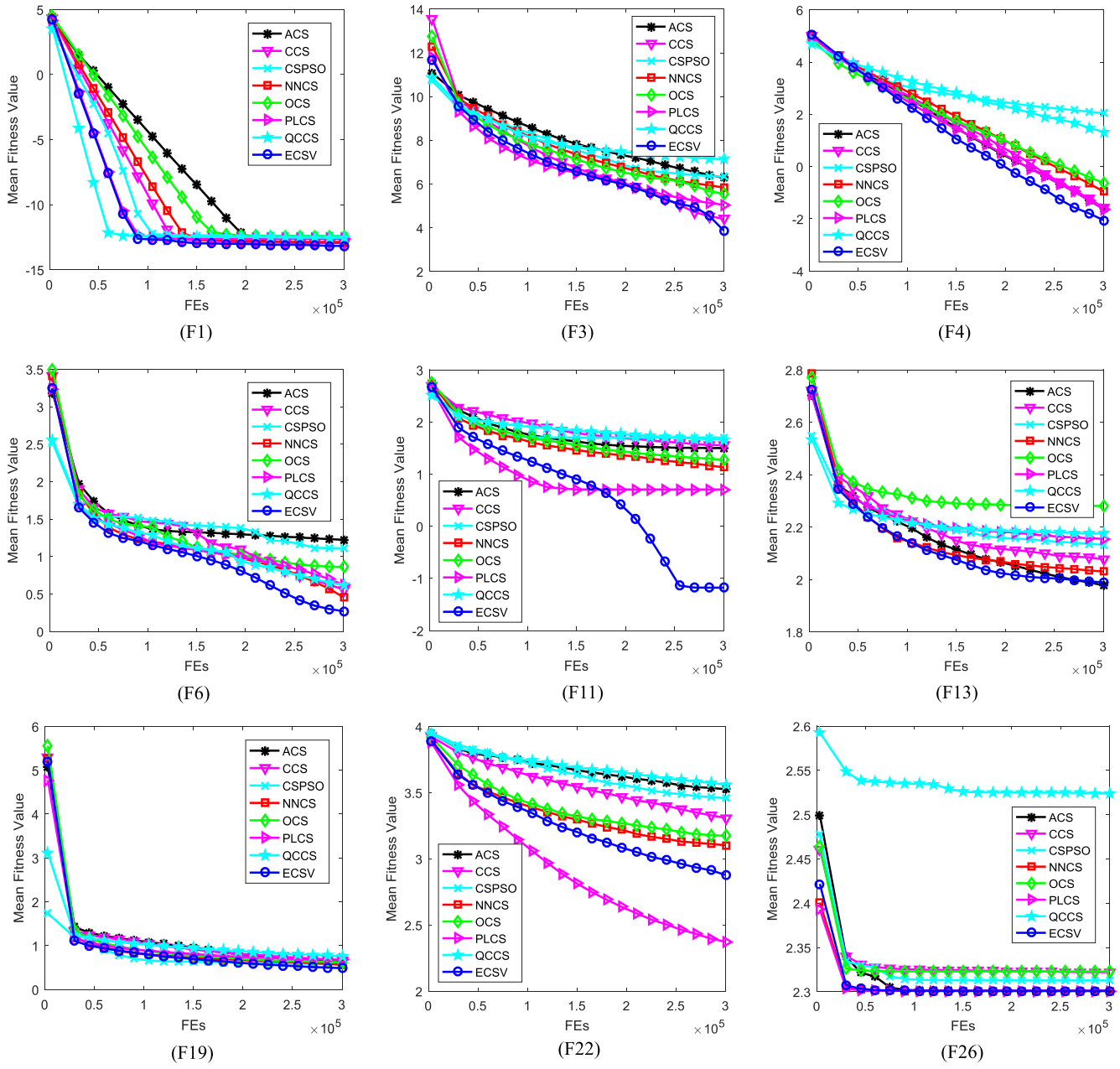

Fig. 2. Convergence graphs of ACS, CCS, CSPSO, NNCS, OCS, PLCS, QCCS and ECSV on the second test suite with 30 variables.

optimization performance of the ensemble algorithm with an external archive.

4.4. Comparison of ECSV with other CS variants

For comparison purposes, seven well-established CS variants are employed in this subsection. These CS algorithms include adaptive CS (ACS) (Sarangi, Panda, Das, & Abraham, 2018), chaos-enhanced CS (CCS) (Huang et al., 2016), the hybridization of CS and PSO (CSPSO) (Chi, Su, Zhang, Chi, & Zhang, 2019), nearest neighbour CS (NNCS) (Wang et al., 2016), oriented CS with Levy distribution and Cauchy distribution (OCS) (Cui, Sun, Wang, Xue, & Chen, 2017), peer-learning CS (PLCS) (Yang et al., 2017) and quantum chaotic CS (QCCS) (Boushaki, Kamel, & Bendjeghaba, 2018). Considering the stochastic characteristics, each algorithm is run 30 times independently for each problem. The population size is set to 50 for all the involved algorithms, and the allowed maximum number of function evaluations for these test problems with 30 and 50 decision variables is set to 300,000 and 500,000, respectively. The mean value and standard deviation (SD) are used as the performance metrics for comparison, and the best result is marked in boldface. Further, to get statistically sound conclusion, the Friedman test is conducted, the average ranking value (ARV) and the final rank (Rank) of each algorithm are presented. The parameter settings of these algorithms are given in Table 3.

4.4.1. Results and comparisons on the first test suite

In this subsection, ECSV is compared with other CS variants on the first test suite with 30 and 50 variables, and the comparison results are listed in Tables 4 and 5, respectively. For each algorithm, the average ranking value and the final rank are provided at the bottom of these tables. Additionally, to investigate the convergence rate of the proposed algorithm, the convergence graphs of some test problems with 30 variables are plotted in Fig. 1, where *FEs* represents the number of function evaluations. For the test problems with 50 variables, the

Table 8
Comparison results of CLPSO, DNLPSO, LFPSO, BBDE, jDE, SaDE, EPSDE and ECSV on the first test suite with 30 variables.

Fun	Term	CLPSO	DNLPSO	LFPSO	BBDE	jDE	SaDE	EPSDE	ECSV
f1	Mean	1.40E−023	2.90E−029	5.35E−028	**0.00E+000**	**0.00E+000**	**0.00E+000**	6.73E−030	**0.00E+000**
	SD	9.85E−024	6.96E−029	4.93E−028	**0.00E+000**	**0.00E+000**	**0.00E+000**	3.69E−029	**0.00E+000**
f2	Mean	3.12E+003	1.80E−016	1.06E−002	5.82E−013	9.81E−007	6.88E−006	**4.97E−026**	2.86E−011
	SD	7.34E+002	5.10E−016	7.30E−003	7.34E−013	3.05E−006	1.93E−005	**6.27E−026**	4.42E−011
f3	Mean	1.74E+007	1.37E+006	2.40E+006	5.56E+005	1.60E+005	4.78E+005	1.63E+006	**5.60E+004**
	SD	4.58E+006	1.15E+006	1.04E+006	2.83E+005	1.03E+005	2.11E+005	6.12E+006	**9.47E+004**
f4	Mean	1.38E+004	8.67E+000	3.78E+002	**3.80E−003**	6.19E−002	1.68E+002	7.63E+001	2.14E−001
	SD	3.23E+003	1.33E+001	4.04E+002	**8.10E−003**	7.02E−002	2.91E+002	4.14E+002	2.97E−001
f5	Mean	4.96E+003	2.75E+003	2.07E+003	2.54E+003	**2.28E+002**	3.30E+003	1.39E+003	1.14E+003
	SD	4.86E+002	8.34E+002	7.34E+002	7.46E+002	**2.64E+002**	4.50E+002	6.85E+002	5.29E+002
f6	Mean	1.40E+000	3.84E+001	7.78E+001	1.98E+001	1.50E+001	4.49E+001	**3.66E−001**	3.50E+000
	SD	1.71E+000	6.42E+001	9.86E+001	3.12E+001	1.94E+001	3.11E+001	**1.01E+000**	1.26E+001
f7	Mean	8.54E−001	1.93E−002	1.53E−002	1.97E−002	1.32E−002	1.92E−002	1.67E−002	**1.70E−003**
	SD	1.51E−001	1.32E−002	1.37E−002	1.56E−002	6.40E−003	1.55E−002	1.27E−002	**5.60E−003**
f8	Mean	**2.09E+001**	**2.09E+001**	**2.09E+001**	**2.09E+001**	2.11E+001	**2.09E+001**	**2.09E+001**	**2.09E+001**
	SD	**3.75E−002**	5.82E−002	6.17E−002	6.00E−002	7.49E−002	5.15E−002	5.64E−002	5.93E−002
f9	Mean	**0.00E+000**	6.32E+001	5.98E+001	5.20E+001	**0.00E+000**	1.33E−001	**0.00E+000**	2.66E−001
	SD	**0.00E+000**	1.52E+001	1.59E+001	3.37E+001	**0.00E+000**	3.44E−001	**0.00E+000**	4.48E−001
f10	Mean	1.32E+002	7.06E+001	7.32E+001	1.74E+002	1.80E+002	4.84E+001	**4.24E+001**	6.88E+001
	SD	1.81E+001	1.86E+001	1.64E+001	1.18E+001	4.70E+001	1.27E+001	**9.27E+000**	1.42E+001
f11	Mean	2.62E+001	1.58E+001	**1.39E+001**	1.90E+001	4.08E+001	1.68E+001	3.47E+001	2.42E+001
	SD	**1.49E+000**	3.08E+000	3.09E+000	1.28E+001	3.51E+000	2.43E+000	3.97E+000	2.89E+000
f12	Mean	1.61E+004	2.05E+004	5.74E+003	1.84E+004	5.74E+004	4.27E+003	3.60E+004	**3.31E+003**
	SD	5.04E+003	2.00E+004	4.26E+003	2.18E+004	5.02E+004	3.98E+003	6.00E+003	**3.25E+003**
f13	Mean	**1.91E+000**	2.79E+000	3.01E+000	1.25E+001	4.47E+000	3.95E+000	1.94E+000	2.76E+000
	SD	2.59E−001	7.12E−001	8.21E−001	1.75E+000	1.30E+000	3.54E−001	**1.65E−001**	6.38E−001
f14	Mean	1.27E+001	**1.21E+001**	1.22E+001	1.30E+001	1.39E+001	1.26E+001	1.35E+001	1.27E+001
	SD	1.91E−001	6.03E−001	5.00E−001	2.73E−001	**1.66E−001**	2.38E−001	2.92E−001	2.06E−001
	ARV	5.2857	4.5000	4.8214	4.9286	4.9643	4.4643	4.0000	**3.0357**
	Rank	8	4	5	6	7	3	2	**1**

The best results, i.e., the lowest mean and standard deviation (SD), are shown in bold.

convergence graphs are similar to those with 30 variables and are therefore not reported.

As indicated in Table 4, in terms of solution quality, ECSV produces the lowest mean value for five test problems. Followed by CCS and OCS, they all get the lowest values for three functions. Specifically, ACS performs better than others on f10, CCS provides promising solutions on f1, f5 and f11, CSPSO is competitive in tackling f14, NNCS finds reasonable results on f1 and f7, PLCS performs well on f1 and f13, OCS does well in handling f2, f6 and f8, while it does not work for improving the solution accuracy on f4 and f10. Similarly, ECSV performs well on f1, f3, f4, f9 and f12, and it is the second best on f7, f10 and f13. Regarding the unimodal problem f1, CCS, NNCS, PLCS, QCCS and ECSV converge to the global optimal solution zero. Also, in terms of the statistical results using Friedman test, ECSV provides the lowest average ranking value of 2.7500, which illustrates that ECSV can well balance the exploration and exploitation capabilities for these test problems with different characteristics. Therefore, the proposed ECSV is very competitive in handling the first test suite with 30 decision variables.

From Table 5, CCS can produce reasonable results on f1, f3, f5 and f11, but it does not work for f6, f9 and f14. Also, PLCS does well in solving the functions f2, f12 and f13, ECSV finds promising solutions on f1, f4, f6, f7 and f9, while ACS, CSPSO, NNCS, OCS and QCCS can only get the lowest mean value of one of these test functions. Note that, NNCS exhibits good overall performance in dealing with these problems. With respect to the unimodal function f1 with 50 variables, CCS, NNCS and ECSV can converge to the global optimum, while the search behavior of PLCS and QCCS may lead to individual oscillation, and they cannot produce the optimal solution. Further, using the Friedman test, these involved CS algorithms are sorted in the following order: ECSV, PLCS, NNCS, CCS, OCS, CSPSO, ACS and QCCS, which means that ECSV provides the best performance and ranks first among these CS algorithms. Besides, for these test problems with different variables, ECSV algorithm produces the smallest standard deviation for more functions, which indicates that ECSV has better robustness. In terms of the above analysis, we can conclude that ECSV exhibits outstanding and robust performance over other CS variants in tackling these test problems with 30 and 50 variables.

Since ECSV is an ensemble algorithm, its convergence speed depends largely on that of the three constituent methods. As shown in Fig. 1, among these involved CS variants, ECSV may converge slower than some other algorithms, but it exhibits better overall performance. More specifically, for f1, ECSV converges slower than QCCS, while it can also find the optimal solution at the cost of fewer function evaluations. For f3, the curve of ECSV descends faster than those of others. With respect to f10, only ACS and QCCS converge faster than ECSV. Also, for f4, f9 and f12, ECSV converges relatively slowly in the initial stage of evolution, but it has stronger search ability in the later stage. In summary, ECSV can be regarded as a promising optimization technique.

4.4.2. Results and comparisons on the second test suite

For a comprehensive comparison, the convergence performance of ECSV is further investigated on the second test suite with 30 and 50 variables. The parameter settings of all algorithms are the same as those mentioned above. The comparison results in terms of mean value and standard deviation are provided in Tables 6 and 7, and the Friedman test is also conducted. For these test problems with 30 variables, the convergence graphs of some functions are presented in Fig. 2.

From Table 6, it is found that ECSV demonstrates promising

Table 9
Comparison results of CLPSO, DNLPSO, LFPSO, BBDE, jDE, SaDE, EPSDE and ECSV on the second test suite with 30 variables.

Fun	Term	CLPSO	DNLPSO	LFPSO	BBDE	jDE	SaDE	EPSDE	ECSV
F1	Mean	2.27E−013	2.12E−013	2.35E−013	2.73E−013	**0.00E+000**	**0.00E+000**	5.31E−014	6.82E−014
	SD	**0.00E+000**	5.77E−014	4.15E−014	9.25E−014	**0.00E+000**	**0.00E+000**	9.78E−014	1.06E−013
F2	Mean	2.01E+007	1.95E+006	2.22E+006	5.08E+005	1.13E+005	2.94E+005	2.26E+006	**3.30E+002**
	SD	4.97E+006	1.08E+006	1.02E+006	6.33E+005	4.88E+004	1.18E+005	8.21E+003	**1.07E+003**
F3	Mean	1.27E+009	9.00E+007	8.99E+007	2.12E+007	5.05E+006	1.01E+007	8.43E+007	**7.51E+003**
	SD	7.90E+008	1.19E+008	1.19E+008	3.79E+007	7.25E+006	2.34E+007	2.25E+008	**3.63E+004**
F4	Mean	4.01E+004	1.42E+002	4.96E+002	1.99E+003	5.73E+003	6.67E+001	9.76E+003	**8.80E−003**
	SD	9.15E+003	8.68E+001	2.00E+002	5.07E+002	2.12E+004	1.28E+002	2.85E+004	**4.20E−002**
F5	Mean	2.77E−013	2.20E−013	2.43E−013	2.73E−013	1.14E−013	**0.00E+000**	1.40E−013	1.17E−013
	SD	5.73E−014	6.63E−014	5.77E−014	7.07E−014	**0.00E+000**	**0.00E+000**	5.73E−014	2.08E−014
F6	Mean	3.34E+001	3.52E+001	3.18E+001	2.16E+001	1.37E+001	3.16E+001	9.39E+000	**1.86E+000**
	SD	7.53E+000	2.36E+001	2.32E+001	2.13E+001	5.13E+000	2.93E+001	**1.72E+000**	6.68E+000
F7	Mean	9.58E+001	5.11E+001	3.92E+001	3.96E+001	**4.34E+000**	1.79E+001	6.29E+001	4.95E+001
	SD	1.13E+001	2.60E+001	2.00E+001	2.51E+001	**2.98E+000**	1.08E+001	3.08E+001	1.50E+001
F8	Mean	2.10E+001	**2.09E+001**	**2.09E+001**	**2.09E+001**	2.11E+001	2.10E+001	**2.09E+001**	**2.09E+001**
	SD	5.53E−002	6.62E−002	4.94E−002	5.19E−002	4.86E−002	4.51E−002	5.29E−002	**4.50E−002**
F9	Mean	2.82E+001	1.77E+001	1.71E+001	**1.16E+001**	4.00E+001	1.72E+001	3.38E+001	2.56E+001
	SD	1.91E+000	3.13E+000	2.90E+000	2.25E+000	4.38E+000	3.10E+000	3.67E+000	**1.82E+000**
F10	Mean	6.96E+000	1.56E−001	1.61E−001	2.18E−001	3.30E−002	2.40E−001	7.47E−002	**6.90E−003**
	SD	2.03E+000	1.17E−001	8.31E−002	9.22E−002	1.71E−002	1.34E−001	3.66E−002	**6.30E−003**
F11	Mean	7.01E−014	3.78E+001	3.42E+001	9.13E+001	**1.89E−015**	4.25E+000	**1.89E−015**	6.63E−002
	SD	2.45E−014	1.07E+001	9.19E+000	4.12E+001	**1.04E−014**	1.95E+000	**1.04E−014**	2.52E−001
F12	Mean	1.47E+002	5.99E+001	6.14E+001	1.71E+002	1.70E+002	**4.22E+001**	5.21E+001	5.55E+001
	SD	1.79E+001	1.64E+001	1.61E+001	1.49E+001	4.68E+001	**1.11E+001**	1.48E+001	1.35E+001
F13	Mean	1.90E+002	1.31E+002	1.38E+002	1.73E+002	2.05E+002	9.64E+001	**7.91E+001**	9.76E+001
	SD	1.61E+001	2.62E+001	2.75E+001	**1.20E+001**	2.59E+001	2.70E+001	2.12E+001	2.37E+001
F14	Mean	1.36E+001	1.35E+003	1.33E+003	5.62E+003	8.26E+000	4.05E+002	**2.98E−001**	6.08E+002
	SD	4.73E+000	2.96E+002	2.63E+002	6.85E+002	1.59E+001	1.25E+002	**2.29E−001**	2.27E+002
F15	Mean	4.74E+003	**3.43E+003**	3.46E+003	7.19E+003	8.23E+003	6.10E+003	6.67E+003	4.04E+003
	SD	3.78E+002	6.41E+002	7.89E+002	**3.10E+002**	5.30E+002	7.37E+002	8.40E+002	5.92E+002
F16	Mean	**1.72E+000**	1.90E+000	1.93E+000	2.38E+000	3.91E+000	2.40E+000	2.45E+000	1.78E+000
	SD	2.92E−001	6.62E−001	3.52E−001	**2.48E−001**	6.16E−001	2.93E−001	2.53E−001	2.86E−001
F17	Mean	3.11E+001	5.49E+001	5.91E+001	1.84E+002	3.07E+001	5.32E+001	**3.04E+001**	5.12E+001
	SD	1.75E−001	8.46E+001	9.64E+000	1.17E+001	2.88E−001	3.67E+000	**2.88E−004**	5.49E+000
F18	Mean	2.04E+002	1.41E+002	2.18E+002	2.01E+002	2.35E+002	1.85E+002	1.33E+002	**1.20E+002**
	SD	1.77E+001	4.60E+001	1.60E+001	1.32E+001	1.92E+001	**8.71E+000**	1.31E+001	2.23E+001
F19	Mean	**6.32E−001**	2.53E+000	2.62E+000	1.26E+001	4.49E+000	6.75E+000	1.92E+000	3.06E+000
	SD	**1.62E−001**	5.14E−001	8.09E−001	1.60E+000	1.70E+000	1.56E+000	1.78E−001	7.57E−001
F20	Mean	1.32E+001	1.20E+001	1.19E+001	1.45E+001	1.33E+001	**1.13E+001**	1.31E+001	1.24E+001
	SD	4.09E−001	1.61E+000	1.26E+000	1.06E+000	**3.10E−001**	3.49E−001	6.51E−001	5.65E−001
F21	Mean	**2.61E+002**	2.96E+002	3.18E+002	2.94E+002	2.86E+002	3.41E+002	2.94E+002	3.81E+002
	SD	**4.19E+001**	7.70E+001	8.61E+001	8.14E+001	6.03E+001	7.77E+001	8.13E+001	7.23E+001
F2	Mean	**1.49E+002**	1.48E+003	1.74E+003	5.01E+003	9.79E+002	4.79E+002	2.78E+002	7.55E+002
	SD	**2.30E+001**	5.90E+002	4.79E+002	1.25E+003	6.84E+002	3.89E+002	1.37E+002	2.44E+002
F23	Mean	5.17E+003	4.25E+003	**3.98E+003**	7.11E+003	8.48E+003	6.35E+003	6.87E+003	4.91E+003
	SD	3.26E+002	7.72E+002	8.34E+002	**2.98E+002**	6.97E+002	4.29E+002	6.63E+002	6.52E+002
F24	Mean	2.82E+002	2.66E+002	2.58E+002	2.47E+002	2.57E+002	**2.18E+002**	2.91E+002	2.69E+002
	SD	**3.65E+000**	1.06E+001	1.24E+001	1.03E+001	2.43E+001	5.49E+000	4.99E+000	9.46E+000
F25	Mean	2.94E+002	2.90E+002	2.80E+002	2.63E+002	2.67E+002	**2.54E+002**	2.99E+002	2.81E+002
	SD	5.58E+000	1.19E+001	8.50E+000	6.60E+000	2.32E+001	1.64E+001	**3.67E+000**	7.22E+000
F26	Mean	2.02E+002	3.11E+002	3.27E+002	2.94E+002	**2.00E+002**	2.04E+002	3.75E+002	**2.00E+002**
	SD	5.54E−001	6.23E+001	5.14E+001	6.29E+001	1.49E−001	2.18E+001	4.47E+001	**2.60E−004**
F27	Mean	1.02E+003	8.37E+002	7.85E+002	6.86E+002	1.20E+003	**5.33E+002**	1.21E+003	9.04E+002
	SD	1.26E+002	1.05E+002	1.01E+002	7.43E+001	1.11E+002	**5.49E+001**	6.22E+001	1.57E+002
F28	Mean	3.00E+002	3.00E+002	4.76E+002	3.37E+002	3.00E+002	**2.93E+002**	3.00E+002	3.00E+002
	SD	1.80E−003	2.89E−013	4.41E+002	2.02E+002	8.44E−014	3.65E+001	**0.00E+000**	2.31E−013
	ARV	5.0179	4.5714	4.7143	5.2679	4.7321	3.6964	4.5179	**3.4821**
	Rank	7	4	5	8	6	2	3	**1**

The best results, i.e., the lowest mean and standard deviation (SD), are shown in bold.

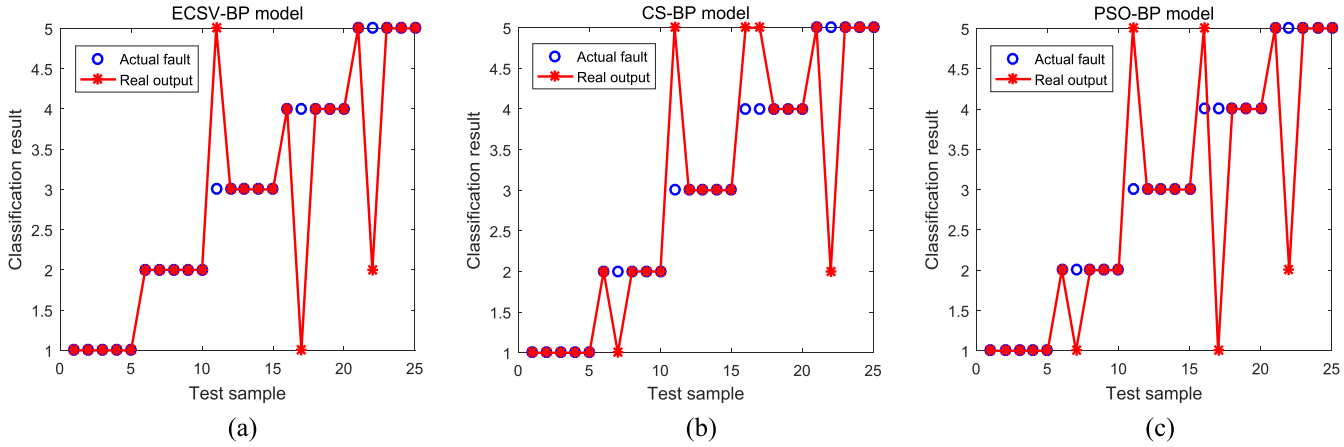

Fig. 3. Classification results of ECSV-BP, CS-BP and PSO-BP models.

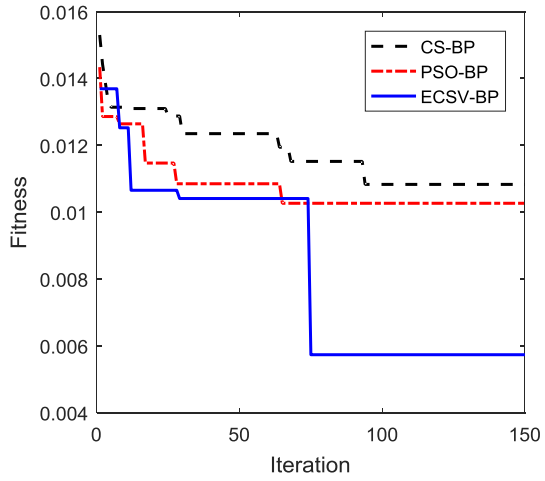

Fig. 4. Fitness curves of ECSV-BP, CS-BP and PSO-BP models.

performance. In more detail, ECSV obtains the best results on ten problems namely F1, F2, F3, F4, F6, F8, F10, F11, F19 and F26, and it is the second best on F5, F12, F13, F14, F15, F17, F18, F22 and F28. Further, NNCS gets the lowest mean value for six problems, followed by ACS, CCS and PLCS for five functions. CSPSO provides the best performance on F8, F20 and F28, OCS produces reasonable results on F8, F15, F16 and F23, while QCCS is not good at solving any problem except F8. As for the statistical results of Friedman test, ECSV produces the lowest average ranking value of 2.9821, which further illustrates that ECSV can effectively exploit the merits of these constituent algorithms. With respect to NNCS, it exhibits the second best convergence performance, which may be due to the use of the nearest neighbor scheme and probability mutation mechanism. Also, QCCS has the worst search ability in solving these test problems among the involved algorithms. In conclusion, ECSV exhibits the competitive overall performance when dealing with the second test suite with 30 variables.

With regard to the comparison results reported in Table 7, it can be seen that the search ability of these algorithms will deteriorate to some extent with the increase of problem variables. Actually, ECSV is superior to or at least comparable to others on F1, F3, F4, F5, F8, F10, F11, F17, F19, F26 and F28, ACS performs best on F12, F13 and F25, and CCS provides the solutions with higher quality on F9, F18, F24 and F27. Also, CSPSO finds reasonable results on F6, F7, F8 and F20, NNCS provides promising solutions on F8, F21, F26 and F28, OCS is the best algorithm for the problems F2, F15, F16 and F23, PLCS performs well on F8, F14, F22, F26 and F28, while QCCS is the loser of the CEC 2013 competition. Further, according to the statistical results of Friedman test, it is impressive noting that ECSV ranks first among these algorithms. CCS ranks second, while QCCS gets the highest average ranking value of 6.1071. In terms of solution stability, ECSV provides the smallest standard deviation for more functions in the second test suite. In summary, ECSV exhibits significantly better performance in solving these benchmark functions with different variables.

As shown in Fig. 2, ECSV produces the best comprehensive performance considering the convergence rate and solution accuracy. Specifically, for these test problems except F22, ECSV converges rapidly with the increasing number of function evaluations and provides reasonable results. Also, ECSV converges only slower than PLCS on F22, and it performs the second best. Apparently, ECSV can find more promising solutions compared with its competitors. To sum up, ECSV is very competitive in dealing with CEC 2013 benchmark problems.

4.5. Comparison of ECSV with other evolutionary algorithms

PSO and DE are two popular optimization techniques. Experimental studies show that they have good convergence performance in solving complex optimization problems. To provide a more comprehensive comparison, the proposed ECSV is compared with several well-known variants of the two algorithms, including CLPSO (Liang, Qin, Suganthan, & Baskar, 2006), DNLPSO (Nasir et al., 2012), LFPSO (Haklı & Uğuz, 2014), BBDE (Omran, Engelbrecht, & Salman, 2009), jDE (Brest, Greiner, Boskovic, Mernik, & Zumer, 2006), SaDE (Qin, Huang, & Suganthan, 2009) and EPSDE (Mallipeddi et al., 2011). In this subsection, the two test suites with 30 variables mentioned above are considered to further investigate the performance of ECSV. Regarding these compared PSO and DE variants, the population size is set to 40, 50, 50, 50, 100, 50 and 50, respectively. Additionally, the maximal function evaluation for all algorithms is set to 300,000, and each method for each test problem is run 30 times. For the two test suites, the mean value and standard deviation obtained by these algorithms are recorded in Tables 8 and 9, respectively. Further, the Friedman test is conducted to compare the performance of all algorithms.

Table 8 provides the comparison results obtained by these involved algorithms on the first test suite with 30 variables. As seen, there is no algorithm that can produce promising solutions for all problems. Specifically, in terms of solution quality, CLPSO wins on f8, f9 and f13, DNLPSO performs best on f14, LFPSO achieves better results on f8 and f11, BBDE finds reasonable solutions on f1, f4 and f8, jDE performs well on f1, f5 and f9, SaDE converges to the global optimum on f1, EPSDE yields better results on f2, f6, f8, f9 and f10, and ECSV produces promising solutions on f1, f3, f7, f8 and f12. According to the results of Friedman test, ECSV is superior to other methods, which illustrates that ECSV is universal in handling various test functions. To sum up, the proposed ECSV algorithm is efficient and effective optimizer in solving

CEC 2005 test problems.

Table 9 presents the comparison results on the second test suite with 30 variables. In more detail, CLPSO outperforms others on F16, F19, F21 and F22, DNLPSO performs well on F8 and F15, LFPSO has the best performance on F23, BBDE produces better solutions on F8 and F9, jDE provides promising results on F1, F7, F11 and F26, SaDE wins on eight test functions, especially in solving some composition problems. EPSDE yields reasonable solutions on F8, F11, F13, F14 and F17, while ECSV does well in tackling the functions F2, F3, F4, F6, F8, F10, F18 and F26. Also, the performance of these methods ranks as follows: ECSV, SaDE, EPSDE, DNLPSO, LFPSO, jDE, CLPSO and BBDE. Apparently, ECSV exhibits the best search capability, while BBDE is the loser among these competitors. Further, for these 42 test functions, ECSV also has good stability by comparing the standard deviation. From the experimental results on the two test suites, it can be seen that the comprehensive performance of ECSV is the best among these involved methods.

4.6. Power transformer fault diagnosis

In this subsection, the fault diagnosis of power transformer is used as a practical application problem to investigate the application ability of the proposed ECSV algorithm.

Power transformer is one of the most important equipments in power system, which is mainly responsible for voltage change and energy conversion. Faults in the transformer will seriously endanger the safety of power grid, and even cause huge economic losses. Therefore, timely and accurate diagnosis of potential transformer faults is critical to the safe operation of power system. Dissolved gas analysis (DGA) is one of the effective techniques to detect and identify power transformer faults. The existing DGA methods include Roger's ratio method, Duval's triangle method and IEC 60,599 method (Illias, Chai, & Bakar, 2016). Although these methods are easy to implement, they may produce incorrect diagnostic results because their ratio boundaries are too absolute. In recent years, the combination of artificial neural network (ANN) and DGA has achieved good diagnostic results. In artificial neural networks, BP neural network (BPNN) is the most commonly used method, but it has some defects such as over-fitting, weak generalization ability and so on. To enhance the classification performance, a fault diagnosis model of power transformer based on BP neural network and ECSV (ECSV-BP) is constructed. In this scheme, the developed ECSV algorithm is used to optimize the initial weights and thresholds of BPNN.

In this optimization model, the concentration percentages of dissolved gases hydrogen (H_2), methane (CH_4), ethane (C_2H_6), ethylene (C_2H_4) and acetylene (C_2H_2) in transformer oil are used as input eigenvectors. Moreover, five main faults are considered, which are the medium and low thermal fault (T_1), high thermal fault (T_2), low energy discharge (D_1), high energy discharge (D_2) and partial discharge (PD) (Yang et al., 2019). In this work, 91 sets of dissolved gas data from different literatures are employed, 66 of which are used as training samples of the developed ECSV-BP model, and the remaining data are used for testing samples. Obviously, data samples from different sources can verify the generalization and applicability of ECSV-BP model. For ease of calculation, three-layer topological neural network is used to identify the power transformer faults. According to the principle of minimum error, the number of neurons in the hidden layer is set to 12. Furthermore, the population size is set as 30, and the maximum number of iterations is set to 150. To further verify the superiority of ECSV-BP model, BPNN combined with CS (CS-BP) and BPNN combined with PSO (PSO-BP) models are constructed and compared. The classification results of these three models are reported in Fig. 3. It should be added that "1″ in ordinates corresponds to T_1, "2" represents T_2, "3" corresponds to D_1, "4" denotes D_2, and "5" corresponds to PD fault.

From Fig. 3, it is obvious that ECSV-BP produces the highest diagnostic accuracy. In more detail, ECSV-BP provides incorrect classification results for three of these test samples, and the classification accuracy is 88%. The diagnostic accuracy of both CS-BP and PSO-BP is 80%. Moreover, for samples Nos. 11, 17, and 22, the correct classification results cannot be obtained by using these three models, which may be due to the fact that these samples are ill-conditioned data or the number of related types of training data is very small. In summary, it can be concluded that the developed ECSV-BP model has good generalization ability and robustness, and can still achieve better fault diagnosis effect under the condition of insufficient training samples.

Fig. 4 provides the fitness curves of different models. As seen, PSO-BP has the fastest convergence speed and its fitness value remains unchanged after about 60 generations. At the beginning of the iteration, the fitness curve of ECSV-BP is steeper than other models. After 72 iterations, the fitness curve descends sharply, which indicates that ECSV-BP jumps out of the local optimal region and continues to search for the optimal solution. Furthermore, CS-BP exhibits the slowest convergence rate and the lowest search accuracy. Overall, the proposed ECSV algorithm can better solve the parameter optimization problem of BP neural network.

5. Summary of results

For different CS algorithms, they may have different properties and are suitable for solving different types of optimization problems. Thus, it is promising to combine the merits of multiple CS algorithms to develop a new CS variant. In this work, ECSV is integrated by CCS, NNCS and PLCS. Also, the introduction of an external archive can help to maintain the population diversity and further enhance the overall performance. Extensive experiments are conducted on 42 test problems with different variables to investigate the effectiveness of ECSV algorithm. Since these problems are complex shifted and rotated functions, finding their optimal solutions is a very challenging task. In terms of the comparison results, it can be observed that the proposed ECSV algorithm is a very competitive method. More specifically, compared with other CS variants, ECSV performs best and ranks first. Meanwhile, ECSV is efficient and effective in comparison with other popular evolutionary algorithms. Besides, the developed ECSV-BP model provides the highest classification accuracy in diagnosing power transformer faults, which also demonstrates that ECSV algorithm is equally applicable to solving real-world problems.

In addition to the advantages mentioned above, the developed ECSV algorithm has some limitations. ECSV algorithm is composed of three different CS variants, so the different selection of the constituent algorithm has a great impact on the comprehensive performance of ECSV. Moreover, since different optimization problems may have different characteristics, it is not guaranteed that ECSV can find promising solutions for all these problems. Therefore, some work should be done to further strengthen the convergence performance.

6. Conclusions

In this paper, we present a new CS algorithm called ECSV to achieve the ensemble of three CS variants, namely CCS, NNCS and PLCS. In ECSV, these constituent algorithms are assigned certain probabilities of being selected on the basis of their previous experiences in generating improved solutions. It should be added that the probability matching operation is triggered periodically. To further enhance the exploration performance, an external archive is introduced into ECSV to store these discarded solutions. Extensive experiments are conducted on two test suites with 30 and 50 decision variables, which are taken from CEC 2005 and CEC 2013. From the comparison results, it can be seen that ECSV is superior to the seven CS variants proposed in recent literature. Furthermore, ECSV is compared with other popular evolutionary algorithms, such as CLPSO, DNLPSO, LFPSO, BBDE, jDE, SaDE and EPSDE. In terms of the Friedman test results, ECSV produces the lowest ranking values for the two test suites. To sum up, ECSV is very efficient and effective in handling these different types of optimization problems. In the future work, we will investigate the influence of other

adaptive selection schemes on the performance of the algorithm and explore a more effective balance strategy between exploration and exploitation. In addition, considering that the current work focuses on the ensemble of three CS algorithms, it may be more interesting to integrate more search strategies to further enhance the universality and robustness of the algorithm.

Acknowledgements

The work is supported by the National Natural Science Foundation of China under Grant No. 51669006, No. 61773314.

References

Abdel-Basset, M., Hessin, A. N., & Abdel-Fatah, L. (2018). A comprehensive study of cuckoo-inspired algorithms. *Neural Computing and Applications, 29*(2), 345–361.
Awad, N. H., Ali, M. Z., & Suganthan, P. N. (2018). Ensemble of parameters in a sinusoidal differential evolution with niching-based population reduction. *Swarm and Evolutionary Computation, 39*, 141–156.
Boushaki, S. I., Kamel, N., & Bendjeghaba, O. (2018). A new quantum chaotic cuckoo search algorithm for data clustering. *Expert Systems With Applications, 96*, 358–372.
Brest, J., Greiner, S., Boskovic, B., Mernik, M., & Zumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation, 10*(6), 646–657.
Chen, X., & Yu, K. J. (2019). Hybridizing cuckoo search algorithm with biogeography-based optimization for estimating photovoltaic model parameters. *Solar Energy, 180*, 192–206.
Cheng, J. T., Wang, L., Jiang, Q. Y., Cao, Z. J., & Xiong, Y. (2018). Cuckoo search algorithm with dynamic feedback information. *Future Generation Computer Systems, 89*, 317–334.
Cheng, J. T., Wang, L., & Xiong, Y. (2019). Cuckoo search algorithm with memory and the vibrant fault diagnosis for hydroelectric generating unit. *Engineering with Computers, 35*(2), 687–702.
Chi, R., Su, Y. X., Zhang, D. H., Chi, X. X., & Zhang, H. J. (2019). A hybridization of cuckoo search and particle swarm optimization for solving optimization problems. *Neural Computing and Applications, 31*(Supplement 1), 653–670.
Cui, Z. H., Sun, B., Wang, G. G., Xue, Y., & Chen, J. J. (2017). A novel oriented cuckoo search algorithm to improve DV-Hop performance for cyber-physical systems. *Journal of Parallel and Distributed Computing, 103*, 42–52.
Fouladgar, N., Hasanipanah, M., & Amnieh, H. B. (2017). Application of cuckoo search algorithm to estimate peak particle velocity in mine blasting. *Engineering with Computers, 33*(2), 181–189.
Goli, A., Tirkolaee, E. B., Malmir, B., Bian, G. B., & Sangaiah, A. K. (2019). A multi-objective invasive weed optimization algorithm for robust aggregate production planning under uncertain seasonal demand. *Computing*. https://doi.org/10.1007/s00607-018- 00692-2.
Haklı, H., & Uğuz, H. (2014). A novel particle swarm optimization algorithm with Levy flight. *Applied Soft Computing, 23*, 333–345.
Huang, L., Ding, S., Yu, S. H., Wang, J., & Lu, K. (2016). Chaos-enhanced Cuckoo search optimization algorithms for global optimization. *Applied Mathematical Modelling, 40*, 3860–3875.
Illias, H. A., Chai, X. R., & Bakar, A. H. A. (2016). Hybrid modified evolutionary particle swarm optimisation-time varying acceleration coefficient-artificial neural network for power transformer fault diagnosis. *Measurement, 90*, 94–102.
Jia, B., Yu, B., Wu, Q., Wei, C., & Law, R. (2016). Adaptive affinity propagation method based on improved cuckoo search. *Knowledge-Based Systems, 111*, 27–35.
Kanagaraj, G., Ponnambalam, S. G., Jawahar, N., & Mukund, N. J. (2014). An effective hybrid cuckoo search and genetic algorithm for constrained engineering design optimization. *Engineering Optimization, 46*(10), 1331–1351.
Kordestani, J. K., Firouzjaee, H. A., & Meybodi, M. R. (2018). An adaptive bi-flight cuckoo search with variable nests for continuous dynamic optimization problems. *Applied Intelligence, 48*(1), 97–117.
Laha, D., & Gupta, J. N. (2018). An improved cuckoo search algorithm for scheduling jobs on identical parallel machines. *Computers & Industrial Engineering, 126*, 348–360.
Li, S. F., & Cheng, C. Y. (2017). Particle swarm optimization with fitness adjustment parameters. *Computers & Industrial Engineering, 113*, 831–841.
Li, G. H., Lin, Q. Z., Cui, L. Z., Du, Z. H., Liang, Z. P., Chen, J. Y., ... Ming, Z. (2016). A novel hybrid differential evolution algorithm with modified CoDE and JADE. *Applied Soft Computing, 47*, 577–599.
Li, X. T., & Yin, M. H. (2015). Modified cuckoo search algorithm with self adaptive parameter method. *Information Sciences, 298*, 80–97.
Liang, J. J., Qu, B. Y., Suganthan, P. N., & Hernández-Díaz, A. G. (2013). Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization. Technical Report.
Liang, J. J., Qin, A. K., Suganthan, P. N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation, 10*(3), 281–295.
Liu, X. Y., & Fu, M. L. (2015). Cuckoo search algorithm based on frog leaping local search and chaos theory. *Applied Mathematics and Computation, 266*, 1083–1092.
Lynn, N., & Suganthan, P. N. (2017). Ensemble particle swarm optimizer. *Applied Soft Computing, 55*, 533–548.
Ma, H. S., Li, S. X., Li, S. F., Lv, Z. N., & Wang, J. S. (2018). An Improved dynamic self-adaption cuckoo search algorithm based on collaboration between subpopulations. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-018-3512-3.
Majumder, A., Laha, D., & Suganthan, P. N. (2018). A hybrid cuckoo search algorithm in parallel batch processing machines with unequal job ready times. *Computers & Industrial Engineering, 124*, 65–76.
Mallipeddi, R., & Suganthan, P. N. (2010). Ensemble of constraint handling techniques. *IEEE Transactions on Evolutionary Computation, 14*(4), 561–579.
Mallipeddi, R., Suganthan, P. N., Pan, Q. K., & Tasgetiren, M. F. (2011). Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing, 11*, 1679–1696.
Meng, X. J., Chang, J. X., Wang, X. B., & Wang, Y. M. (2019). Multi-objective hydropower station operation using an improved cuckoo search algorithm. *Energy, 168*, 425–439.
MiarNaeimi, F., Azizyan, G., & Rashki, M. (2018). Multi-level cross entropy optimizer (MCEO): An evolutionary optimization algorithm for engineering problems. *Engineering with Computers, 34*(4), 719–739.
Nasir, M., Das, S., Maity, D., Sengupta, S., Halder, U., & Suganthan, P. N. (2012). A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization. *Information Sciences, 209*, 16–36.
Omran, M. G. H., Engelbrecht, A. P., & Salman, A. (2009). Bare bones differential evolution. *European Journal of Operational Research, 196*, 128–139.
Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation, 13*(2), 398–417.
Rao, R. V., Savsani, V. J., & Vakharia, D. P. (2012). Teaching-learning-based optimization: An optimization method for continuous non-linear large scale problems. *Information Sciences, 183*, 1–15.
Salgotra, R., Singh, U., & Saha, S. (2018). New cuckoo search algorithms with enhanced exploration and exploitation properties. *Expert Systems With Applications, 95*, 384–420.
Salza, P., & Ferrucci, F. (2019). Speed up genetic algorithms in the cloud using software containers. *Future Generation Computer Systems, 92*, 276–289.
Samir, K., Brahim, B., Capozucca, R., & Wahab, M. A. (2018). Damage detection in CFRP composite beams based on vibration analysis using proper orthogonal decomposition method with radial basis functions and cuckoo search algorithm. *Composite Structures, 187*, 344–353.
Sarangi, S. K., Panda, R., Das, P. K., & Abraham, A. (2018). Design of optimal high pass and band stop FIR filters using adaptive Cuckoo search algorithm. *Engineering Applications of Artificial Intelligence, 70*, 67–80.
Storn, R., & Price, K. (1997). Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341–359.
Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. -P., Auger, A., & Tiwari, S. (2005). Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical Report.
Tirkolaee, E. B., Goli, A., Hematian, M., Sangaiah, A. K., & Han, T. (2019). Multi-objective multi-mode resource constrained project scheduling problem using Pareto-based algorithms. *Computing*. https://doi.org/10.1007/s00607-018- 00693-1.
Tong, L. Y., Dong, M. G., & Jing, C. (2018). An improved multi-population ensemble differential evolution. *Neurocomputing, 290*, 130–147.
Vafashoar, R., & Meybodi, M. R. (2018). Multi swarm optimization algorithm with adaptive connectivity degree. *Applied Intelligence, 48*(4), 909–941.
Valian, E., & Valian, E. (2013). A cuckoo search algorithm by Lévy flights for solving reliability redundancy allocation problems. *Engineering Optimization, 45*(11), 1273–1286.
Vrugt, J. A., Robinson, B. A., & Hyman, J. M. (2009). Self-adaptive multimethod search for global optimization in real-parameter spaces. *IEEE Transactions on Evolutionary Computation, 13*(2), 243–259.
Walton, S., Hassan, O., Morgan, K., & Brown, M. R. (2011). Modified cuckoo search: A new gradient free optimisation algorithm. *Chaos, Solitons & Fractals, 44*, 710–718.
Wang, G. G., Gandomi, A. H., Zhao, X. J., & Chu, H. C. E. (2016). Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. *Soft Computing, 20*(1), 273–285.
Wang, H., Wu, Z. J., Rahnamayan, S., Sun, H., Liu, Y., & Pan, J. S. (2014). Multi-strategy ensemble artificial bee colony algorithm. *Information Sciences, 279*, 587–603.
Wang, L. J., Zhong, Y. W., & Yin, Y. L. (2016). Nearest neighbour cuckoo search algorithm with probabilistic mutation. *Applied Soft Computing, 49*, 498–509.
Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*(1), 67–82.
Wu, G. H., Shen, X., Li, H. F., Chen, H. K., Lin, A. P., & Suganthan, P. N. (2018). Ensemble of differential evolution variants. *Information Sciences, 423*, 172–186.
Xiong, G. J., Shi, D. Y., & Duan, X. Z. (2013). Multi-strategy ensemble biogeography-based optimization for economic dispatch problems. *Applied Energy, 111*, 801–811.
Yang, X. H., Chen, W. K., Li, A. Y., Yang, C. S., Xie, Z. H., & Dong, H. Y. (2019). BA-PNN-based methods for power transformer fault diagnosis. *Advanced Engineering Informatics, 39*, 178–185.
Yang, X. S., & Deb, S. (2014). Cuckoo search: Recent advances and applications. *Neural Computing and Applications, 24*(1), 169–174.
Yang, Q. D., Gao, H. B., & Zhang, W. J. (2017). Biomass concentration prediction via an input-weighed model based on artificial neural network and peer-learning cuckoo search. *Chemometrics and Intelligent Laboratory Systems, 171*, 170–181.
Yildiz, A. R. (2013). Cuckoo search algorithm for the selection of optimal machining parameters in milling operations. *International Journal of Advanced Manufacturing Technology, 64*(1–4), 55–61.