Expert Systems With Applications 184 (2021) 115495

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A knowledge-based differential covariance matrix adaptation cooperative algorithm

Yang Zuo [a,1], Fuqing Zhao [b], Zekai Li [c]

[a] Electronic and Information Engineering College, Henan Institute of Technology, Xinxiang 453000, China
[b] School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou 730050, China
[c] Beijing Hollysys Industrial Software Co., Ltd, Xi'an 710077, China

ARTICLE INFO

Keywords:
Cooperation
Differential evolution
Covariance matrix adaptation
Niching population size
Learning mechanism
Design of experiment

ABSTRACT

In this paper, a knowledge-based differential covariance matrix adaptation cooperative algorithm (DCMAC) is proposed for continuous problems. On the basis of combining successful history-based adaptive DE variants with linear population size reduction (LSHADE) and covariance matrix adaptation evolutionary strategy (CMA-ES), DCMAC proposes a strategy based on knowledge reward and punishment to achieve the purpose of collaborative optimization. Afterward, an adaptive learning mechanism is introduced to optimize the parameters to balance the exploitation and exploration of DCMAC. This process enables the algorithm to have global search capability. Finally, the niching-based population size reduction mechanism is introduced to improve the local search ability of the DCMAC. A weighted mutation strategy with dynamic greedy p value and covariance matrix adaptation (CMA) sampled based on differential vector are presented. Meanwhile, the knowledge acquired in the previous iteration process is adopted in the algorithm to select a mutation strategy for generating the new candidate solutions in the next iteration. The niching population size reduction mechanism is introduced to maintain the diversity of the population and compared with the other classical population size reduction methods. The optimal combination of parameters in the DCMAC algorithm is testified by the design of the experiment. Furthermore, the DCMAC is testified on the CEC2017 benchmark functions. The effectiveness and efficiency of the DCMAC are demonstrated by the experimental results in solving complex continuous problems.

1. Introduction

Evolutionary algorithms (EAs) are the type of random search algorithms inspired by the evolution process of natural organisms (Viktorin, Senkerik, Pluhacek, Kadavy, & Zamuda, 2019). In the past decades, EAs have shown significant performance for solving various complicated problems, which are hard to address by certain traditional calculation methods (Zhao et al., 2018a). In addition, numerous EAs are applied to solve realistic engineering optimization and scientific applications (Ruiz, Pan, & Naderi, 2019; Wang & Wang, 2018). Classical *meta*-heuristic algorithms, including genetic algorithms (GA) (David & John, 1988), artificial bee colony algorithm (ABC) (Ji et al., 2019; Karaboga & Basturk, 2007), particle swarm optimization (PSO) (James et al., 1995; Mohammad et al., 2017), differential evolution (Rainer & Kenneth,1995; Xueqi & Che, 2019), and other typical hybrid evolution computation algorithm, are proposed.

Differential evolution (DE) algorithm is a population-based random search algorithm proposed by Rainer and Kenneth (1995). The DE is solved through cooperation and competition among individuals in the population. The basic operations of DE include mutation, crossover, and selection. After the population of DE is initialized, three operations are repeated until a certain termination condition is satisfied. The performance of DE is mainly determined by the mutation strategy and the crossover operation (Ali et al., 2019). Besides, the intrinsic control parameters are applied to balance the diversity of the population and accelerate the convergence rate of DE. The effective control mechanism of parameters and the adaptive search strategy are required to improve the performance of the algorithm. There are two different schemes to improve the parameters of DE, including deterministic rules and adaptive parameter control settings (Das et al., 2016). The deterministic rules are applied to control the parameter setting based on certain deterministic rules. The adaptive parameter control settings are implemented mainly through the interaction between feedback information of the search process of the algorithm and parameter control strategies. The

E-mail addresses: 19858324@qq.com (Y. Zuo), Fzhao2000@hotmail.com (F. Zhao), 410379065@qq.com (Z. Li).
[1] ORCID ID: 0000-0001-6594-1846.

https://doi.org/10.1016/j.eswa.2021.115495
Received 17 August 2020; Received in revised form 24 June 2021; Accepted 25 June 2021
Available online 30 June 2021

parameters of DE are adaptively controlled to ensure that the parameters of each generation are dynamically adjusted (Noor et al., 2017). A series of improvements in DE are proposed to overcome the disadvantages of DE in the past decades. According to the literature, a new mutation strategy, named *"DE/current −to −pbest"*, is proposed to improve the performance of adaptive differential evolution with an optional external archive (JADE) (Zhang and Sanderson, 2009). A success-history-based parameter adaptation for differential evolution (SHADE), with a success-historical memory for mutation factor F and crossover factor Cr, is proposed by Ryoji and Alex (2013) to improve the robustness of JADE. In SHADE, the average values of S_{CR} and S_F for each generation are stored in the historical memory M_{CR} and M_F. Therefore, even if a set of inferior values are included in the S_{CR} and S_F, the parameters stored in the memory of the previous generation is not directly impacted. The successful history-based adaptive DE variants with linear population size reduction (LSHADE) (Ryoji and Alex, 2014), which is the enhanced version of SHADE, is the winner of single-objective parameter competition in the CEC2014 benchmark function. The linear population size reduction (LPSR), a simple deterministic population resizing method, is introduced into LSHADE. A rank-based selective pressure strategy of the algorithm, named LSHADE-RSP, is proposed by Vladimir et al. (2018). The basic idea of LSHADE-RSP is to use selection pressure to adapt its mutation strategy.

In addition, the hybrid EAs often have better performances in search ability than that of single EAs in solving complex optimization problems. Therefore, various optimizers are mixed with the LSHADE in order to improve the performance of the LSHADE. The search mechanism of LSHADE is introduced into the biogeography-based optimization algorithm (BBO) to enhance the global search and anti-rotation ability of the algorithm (TDBBO) (Zhao et al., 2019a,b). The simulation results of TDBBO on the CEC2017 benchmark function confirm that TDBBO has better accuracy and convergence rate than the most advanced variant of BBO. The crossover and mutation operations of LSHADE are used for the diversity of the population in the gravitational search algorithm (GSA), named SGSADE (Zhao et al., 2018b). LSHADE is adopted as a local search mechanism in the grey wolf optimizer (GWO), and the exploitation of GWO is enhanced by updating the population (GOGWO2D) (Ibrahim et al., 2018). The experimental results prove that GOGWO2D finds the optimal solution of the global optimization problem. The LSHADE with semi-parameter adaptation hybrid with covariance matrix adaptation evolutionary strategy (LSHADE-SPACMA), a semi-parametric adaptive algorithm, is proposed by Ali et al. (2017). It is integrated with the covariance matrix adaptation evolutionary strategy (CMA-ES) to improve the performance of the LSHADE-SPACMA. A semi-parametric adaptive method based on randomization and self-adaptation is applied in LSHADE with semi-parameter adaptation (LSHADE-SPA). Moreover, CMA-ES combined with the crossover operation is introduced to improve the exploration capabilities of the LSHADE-SPACMA algorithm. Although the performance of LSHADE-SPACMA has been improved by the various strategies, there are certain shortcomings in the LSHADE-SPACMA. First, the parameter setting is difficult to adjust for different problems. Second, the application of LPSR by LSHADE-SPACMA results in the loss of the diversity of the population in the early iteration of the algorithm. In the early stage of an evolutionary process, certain diversity is required in the population of the candidates. If the population is linearly reduced at the initial stage of the iteration, the deleted individuals have insufficient time to evolve in the iteration of candidates.

In this study, a knowledge-based cooperative evolution algorithm DCMAC is proposed for solving continuous optimization problems. In general, the effectiveness of the search process is enhanced by the previously acquired prior knowledge. An effective method via decomposition and coordination is formed by cooperative optimization. A weighted mutation strategy, *"DE/current −to −pBest −w/1"*, is introduced and a dynamic greedy p value is used to improve exploration and exploitation behavior (Janez et al., 2017). In the proposed hybridization scheme, a differential vector is incorporated into the adaptive sampling structure of the CMA-ES to achieve the superior performance of the proposed algorithm. A knowledge-based reward mechanism is introduced to optimize the two cooperation strategies. The knowledge acquired in the previous iteration process is applied in the algorithm to select a mutation strategy for generating the new candidate solutions in the next iteration. A parameter adaptive strategy with a learning mechanism that used two sinusoidal formulas and the Cauchy distribution to adjust the scaling factor F is introduced. According to the average values of the previous generations, the scaling factor F is adjusted adaptively to balance the exploration and exploitation of the DCMAC. Finally, the niching-based population size reduction mechanism is introduced to improve the search ability of the DCMAC. In the early stage of the DCMAC, the population is not reduced to avoid losing the diversity of the population and increase the global search ability of the DCMAC. The population is linearly decreased to increase the local search ability of the DCMAC according to the function evaluation number in the later stage. The effectiveness of the DCMAC is demonstrated by the experimental results. The contributions of this paper are summarized as follows.

- A new weighted mutation strategy with the dynamic greedy value p, named *"DE/current −to −pBest −w/1"*, is applied to improve the search ability of the DCMAC. The difference vector is incorporated with the adaptive sampling to overcome the limitations of the CMA-ES algorithm. A knowledge-based reward mechanism is introduced to optimize the two cooperative strategies.
- A parametric learning mechanism with sinusoidal formula and the Cauchy distribution is introduced into the DCMAC to adjust the scaling factor F and balance the exploration and exploitation of the DCMAC.
- A mechanism based on niching population size reduction is embedded to reduce the population size and to avoid losing the diversity of the population. The convergence performance of DCMAC is analyzed by the Markov model, and the optimal combination of parameters is obtained by the design of experiment (DOE) method (Montgomery,2017).

The structure of this paper is as follows. The proposed DCMAC is described in Section 2. The convergence performance of the DCMAC algorithm is analyzed in Section 3. The related experimental analysis and discussion are in Section 4. Finally, conclusions are summarized and future work is suggested in Section 5.

2. Basic algorithm and operators

In this section, LSHADE and CMA-ES are briefly described. The notations in this paper are listed as follows.

N	the population size
P_0	a random initial population
D	dimension size
$f(X)$	the object function
G	generation counter
P_G	Ggeneration population.
M_{CR}	the historical memory ofCR
M_F	the historical memory of$F$
M_{freq}	the historical memory of$freq_i$
$freq$	the frequency of the sine function
$CR_{i,G}$	a crossover rate of the i-th individual in the Gth generation population
$F_{i,G}$	a scaling factor of the i-th individual in the Gth generation population
ED_i	Euclidean distance between individual X_{ig} and best individual$X_{best,g}$
S_{CR}	the mean values for CR that have been successful in previous generations.
S_F	the mean values for F that have been successful in previous generations.
$randn$	the normal distribution
$randc$	the Cauchy distribution
$X_{best,G}$	the best solution inP_g
$nfes$	the number of evolutions

(*continued on next page*)

(*continued*)

$nfes_{max}$	the maximum number of evolutions
σ	CMA-ES step size
C	the covariance matrix

2.1. LSHADE algorithm

LSHADE is a population-based optimization tool, where the system is initialized with a population of random individuals, and the algorithm searches for the optimal value by updating generations. The population of LSHADE is combined with a set of real parameter vectors $x_i = (x_1, \cdots, x_D)$, $i = 1, \cdots, N$, where N is the population size and D is the dimensionality of the target problem. The mutation vector $\vec{v}_{i,G}$ is generated by using the $''current-to-pbest/1''$ mutation strategy in LSHADE.

$$\vec{v}_{i,G} = \vec{x}_{i,G} + F_{i,G}(\vec{x}_{pbest,G} - x_{i,G}) + F\bullet(\vec{x}_{r_1,G} - \vec{x}_{r_2,G}) \tag{1}$$

where $\vec{x}_{pbest,G}$ is randomly selected from the top $N*p$ ($p \in [0, 1]$) members in the G generation. $r_1$is a random index selected from the population. The other random index r_2 is uniformly selected from a combination of population and external archives. An excellent parent vector that has been successfully generated is saved in the external archive. The scale factor F a positive control parameter for scaling the different vector.

$$\vec{u}_{i,j,G} = \begin{cases} \vec{v}_{i,j,G}, if(rand_{i,j} \le Crorj = j_{rand}) \\ \vec{x}_{i,j,G}, otherwise \end{cases} \tag{2}$$

where $i \in [1, N]$ and $j \in [1, D]$, $Cr \in [0, 1]$ is the crossover rate, a random integer uniformly distributed in $[1, D]$ is represented by j_{rand}.

The greedy selection strategy in LSHADE is adopted to select the optimal solution. The same or desirable fitness function value as $\vec{x}_{i,G}$ is generated by the trial vector $\vec{u}_{i,G}$, $\vec{u}_{i,G}$ is set to $\vec{x}_{i,G+1}$. Otherwise, the original vector $\vec{x}_{i,G}$ is retained. The options are as follows.

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G}, if\left(f\left(\vec{u}_{i,G}\right) \le f\left(\vec{x}_{i,G}\right)\right) \\ \vec{x}_{i,G}, otherwise \end{cases} \tag{3}$$

LSHADE is a parameter adaptive algorithm based on success history. LSHADE is different from other evolutionary algorithms with its unique trial vector generation strategy (mutation operator and crossover operator). The mutation operator perturbs the base vector by the sum of differences between several pairs of difference vectors to generate a mutation vector. The crossover operator generates the trial vector by combining the genes (variables) of the target vector and the mutation vector. The nature of its mutation operator and crossover operator allows LSHADE to control the scale and direction of exploration to find promising solutions (Cui et al., 2019).

2.2. CMA-ES algorithm

The CMA-ES algorithm is gradient-free optimization. In the past few years, CMA-ES has various derivative algorithms. These algorithms usually show positive results when solving high dimensional problems (Nikolaus,2006). In CMA-ES, multivariate normal distribution modelling is adopted to search space, and the new individual generation method used Gaussian distribution. The following are the main steps of the CMA-ES algorithm.

1. The initial population is the same as the population of DE and the function fitness values are evaluated. In the process of population initialization, several important parameters should be initialized first. The m is the mean value of the search distribution for each generation, the initial value of m is 0. σ is the step size, the initial value of σ is 1. C is the covariance matrix, the initial value of C is I. p_c is the evolutionary path of C. p_σ is the evolutionary path of σ . Both p_c and p_σ are initialized to 0.

2. Generating new individuals using the Gaussian distribution.

$$x_i = N(m, \sigma^2 C) \forall i = 1 : n \tag{4}$$

3. Updating the average vector m is used by the best μ individual according to $m = \sum_{i=1}^{\mu} w_i x_i$ where $\sum_{i=1}^{\mu} w_i = 1$ and $w_1 \ge w_2 \ge \cdots \ge w_\mu$.

4. Covariance matrix C and Step σ are updated.

$$C \leftarrow (1 - c_1 - c_\mu)C + c_1 p_c p_c{}^T + c_\mu \sum_{i=1}^{\mu} w_i y_{i:\lambda} y_{i:\lambda}{}^T$$

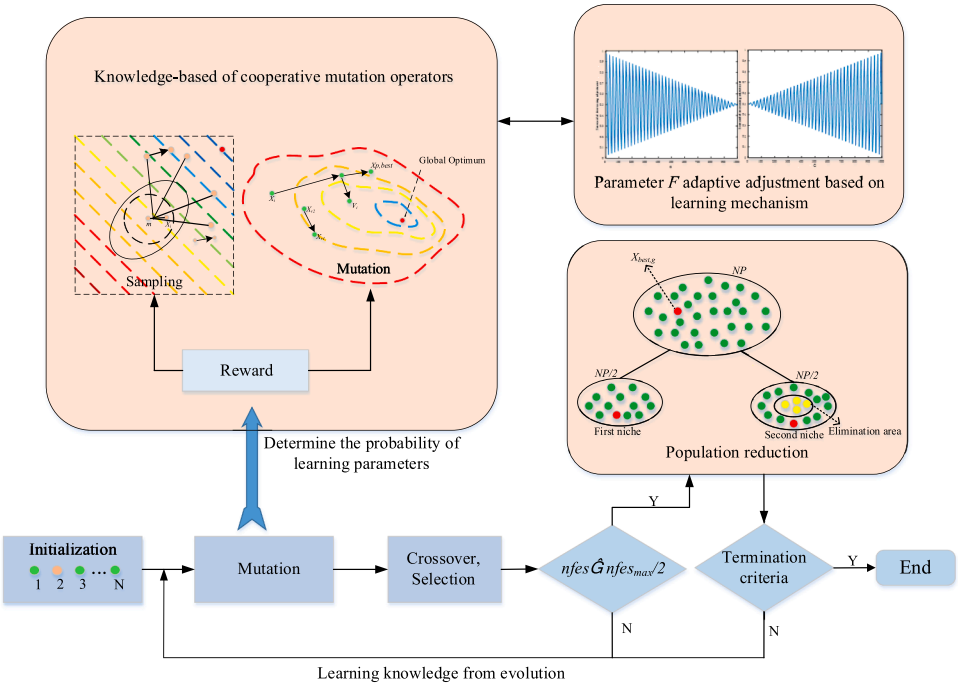


Fig. 1. The flowchart of the DCMAC algorithm.

$$\sigma \leftarrow \sigma \times exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||p_\sigma||}{E||N(0,I)||}-1\right)\right)$$

Based on the number of parent generations$\mu$and the maximum likelihood estimation y, the update process of the covariance matrix C and step σ are completed by using the evolutionary path. The algorithm uses the evolutionary path method to upgrade C and σ. The evolutionary path is the sum of a series of successive steps, also known as the accumulation. An exponential smoothing approach is used to construct the evolutionary path. The evolutionary path formulas are as follows.

$$p_c \leftarrow (1-c_c)p_c + \sqrt{1-(1-c_c)^2}\sqrt{\mu_w}y_w$$

$$p_\sigma \leftarrow (1-c_\sigma)p_\sigma + \sqrt{1-(1-c_\sigma)^2}\sqrt{\mu_w}C^{-\frac{1}{2}}y_w$$

In summary, the main characteristics of CMA-ES are as follows: (1) CMA-ES is invariant to linear transformations in the search space. (2) Multivariate normal distribution is adopted to generate search solutions that conforms to the maximum entropy principle, in which the least possible assumptions on target function are used in the distribution shape. (3) CMA-ES reduces any convex-quadratic function to a sphere model by increasing the possibility of previous successful steps without using derivatives.

3. The proposed DCMAC algorithm

The LSHADE algorithm and its variants are a potential framework to obtain a desired performance for DE-based optimization techniques. It is worth noting that LSHADE has a powerful global-search ability. The CMA-ES is a stochastic method for continuous domain (real-parameter) optimization of non-convex and non-linear functions. The CMA-ES obeys the adaptive covariance matrix to the multivariate normal distribution, whose search distribution is hugely utilized to detect the landscape of a convex-quadratic objective function. Therefore, CMA-ES has strong competitiveness in solving high-dimensional problems and also has strong local search capabilities (Wang et al., 2019; Patrick et al., 2018).

In this paper, the DCMAC algorithm proposes a cooperative mutation strategy to implement two algorithms, LSHADE and CMA-ES. During the execution of the algorithm, the results of the mutated population are recorded from the first iteration as knowledge. The knowledge reward mechanism is proposed to evaluate the fitness of parents and children based on the recorded knowledge. More individuals can be assigned to the algorithm with better fitness in the next iteration. In this process, since the cooperative variation strategy is highly sensitive to parameters, the parameter adaptive learning mechanism is introduced. The sensitive parameters are adjusted according to the mean value of several generations by the method of adaptive sinusoidal parameter learning. This parameter adaptive learning mechanism can realize the purpose of adaptive control of population evolution speed. In addition, the niching population size reduction (NPSR) mechanism is designed to increase the diversity of the population and enhance the local search.

3.1. Description of the DCMAC algorithm

The flowchart of the DCMAC algorithm is shown in Fig. 1. First, the cooperative mutation operator is used to implement the knowledge-based cooperative mutation strategy in the algorithm. The initialized population and the state of each individual in the decision space is determined. Second, A knowledge-based reward mechanism is introduced to optimize the two cooperation strategies. Third, the improved parameter adaptation learning mechanism is used to balance algorithm development and exploration. The scaling factor F based on the experience of previous generations is adjusted to achieve an adaptive parameter learning mechanism. In addition, the niching population size reduction strategy is used to maintain the population diversity and improve the local search ability of the algorithm when $nfes \geq \frac{nfes_{max}}{2}$.

The knowledge-based approach refers to valuable data gathered during evolution to guide further evolution. The collaborative strategy is an allocation that guides the next generation based on the collection of valuable individuals from the previous generation. The collection of valuable individuals from the previous generation is knowledge. In addition, when the algorithm faces different problems in the evolutionary process, such a knowledge-based cooperative strategy can gradually increase its adaptive ability to solve problems based on the data collected in the evolutionary process.

The performance of CMA-ES differs when solving hybrid functions and several complex real-life applications. The DE mutation scheme based on the difference vector is combined with CMA-ES to overcome the limitations of CMA-ES and can adapt to the natural scale of the problem. At each stage of the algorithm, a proper trade-off between exploitation and exploration capabilities is needed. The stagnation of the population is eliminated when using an appropriate percentage of the average vector and the current target vector. Moreover, the $''DE/current-to-pBest-w/1''$ mutation strategy with dynamic p is used to generate a new solution. Although the improved $''DE/current-to-pBest-w/1''$ and CMA-ES assist in improving the search performance of the algorithm, according to the no free lunch theorem, different algorithms have their strengths in solving different problems. A knowledge-based reward mechanism is introduced to optimize the two cooperation strategies. The knowledge acquired in the previous iteration process is applied in the algorithm to select a mutation strategy for generating the new candidate solutions in the next iteration. Therefore, the knowledge-based cooperative strategy is unique in this paper compared with other algorithms.

Differential evolution is one of the well-known sensitive algorithms to parameters. In DCMAC, the parameters have a considerable influence on the exploitation and exploration of algorithms. F is an important parameter to control the population evolution rate, it is closely related to the convergence rate. With the increase of the F, the exploratory of the algorithm is improved, and the mutation vectors are widely distributed in the search space. Therefore, the adaptive sinusoidal parameter learning mechanism is used to adjust the current most suitable value F according to the mean values of previous generations.

In the early stage of the DCMAC, the population is not reduced to avoid losing the diversity of the population and increase the global search ability of the DCMAC. The population is linearly decreased to increase the local search ability of the DCMAC according to the function evaluation number in the later stage. The advantage of niching population size reduction strategy is to give potential individuals enough evolutionary time and maintain the diversity of the population. The pseudo-code of DCMAC is summarized in Algorithm 1.

Algorithm 1 The main procedure of DCMAC.

Initialize: $N = 18*D$, a random initial population $P_0 = (X_{1,0}, \cdots, X_{N,0})$
while failure to meet termination criteria **do**
Generate a F_i using the parameter adaptation learning mechanism.
$[P_{LSHADE}, P_{CMA}] = \text{Split}(P_G, FCP_G)$
Generate donor vectors $v_{G,LSHADE}$ using Eq.(5).
Generate donor vectors $v_{G,CMA}$ using Eq.(6).
v_G = Concatenate ($v_{G,LSHADE}, v_{G,CMA}$) and u_G = Generate trial vectors (v_G, CR_G)
Update P_G according to the evaluation of u_G, Store successful FCP_G, F_G, andCR_G
Niching population size reduction strategies are used.
Update CMA-ES parameters
G++
end
Until*nfes* > *Max_nfes*
Output: the best solution $X_{best,G-1}$ in P_{G-1}.

3.2. Cooperation of mutation operators

In this paper, a weighted version of the mutation strategy (Janez et al., 2017) is introduced, named $''DE/current-to-pBest-w/1''$,

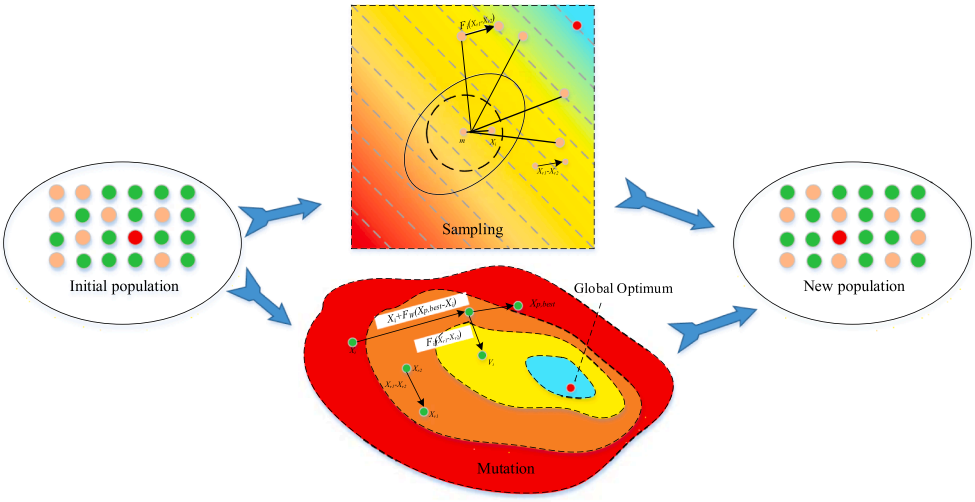


Fig. 2. The examples of cooperation mutation operators.

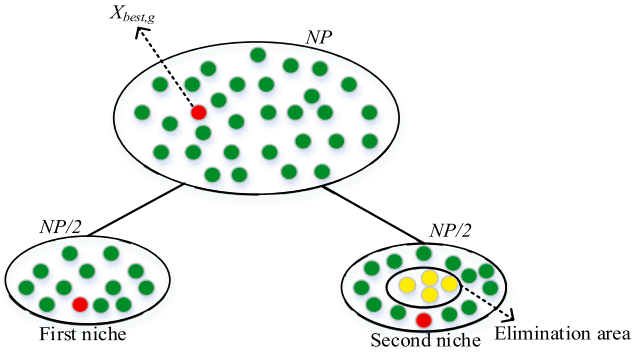


Fig. 3. The examples of niching population size reduction.

according to Eq.(5).

$$\overrightarrow{v}_{i,G} = \overrightarrow{x}_{i,G} + F_\omega\left(\overrightarrow{x}_{pbest,G} - \overrightarrow{x}_{i,G}\right) + F(\overrightarrow{x}_{r_1,G} - \overrightarrow{x}_{r_2,G}) \tag{5}$$

where F is the scaling factor. The scaling factor of different stages F_ω is shown as follows.

$$F_\omega = \begin{cases} 0.7{*}F, nfes < 0.2{*}max\_nfes \\ 0.8{*}F, nfes < 0.4{*}max\_nfes \\ 1.2{*}F, otherwise \end{cases}$$

The purpose of the weighted mutation strategy is to use F_ω multiplied by vector $x_{pbest,G}$ in the DCMAC. Therefore, the values of the control parameters for the DCMAC are different at different stages of evolution, which embodies a search process from global to local. In addition, the mutation strategy is improved by adjusting the greedy dynamic parameters p of mutation to balance the exploration and exploitation of DCMAC. The greedy degree of the algorithm is controlled with p value. When p is set to a comparatively large value, the greedy degree of the algorithm decreases and the search domain extends. Conversely, the greediness of the algorithm is increased by the small p value. To improve the exploration of DCMAC, p is set to a value decreased linearly in the iteration period of the DCMAC. The value p is calculated as follows.

$$p_{G+1} = round[\left(\frac{p_{min} - p_{init}}{max\_nfes}\right){*}nfes + p_{init}]$$

where p_{init} is the initial p value, and p_{min} is the minimum p value. The new population vector is shown as follows.

Table 1
The results for $D = 30$ and $D = 50$.

Func.	30D		50D	
	DCMAC	DCMAC_1	DCMAC	DCMAC_1
F1	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
F3	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
F4	**5.63E + 01**	5.89E + 01	4.72E + 01	**3.83E + 01**
F5	**7.96E + 00**	1.34E + 01	**1.17E + 01**	3.23E + 01
F6	5.37E-09	**2.69E-09**	**0.00E + 00**	2.35E-10
F7	**3.86E + 01**	4.64E + 01	**6.42E + 01**	8.74E + 01
F8	**8.09E + 00**	1.28E + 01	**1.16E + 01**	3.13E + 01
F9	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
F10	**1.41E + 03**	2.40E + 03	**3.14E + 03**	3.58E + 03
F11	1.37E + 01	**1.35E + 01**	**2.58E + 01**	2.93E + 01
F12	**2.26E + 02**	5.45E + 02	**1.21E + 03**	1.54E + 03
F13	**1.50E + 01**	4.18E + 01	**3.13E + 01**	1.32E + 02
F14	**2.13E + 01**	2.99E + 01	**2.48E + 01**	5.07E + 01
F15	**2.42E + 00**	6.46E + 00	**2.13E + 01**	2.83E + 01
F16	**2.64E + 01**	5.23E + 01	**2.76E + 02**	4.15E + 02
F17	**2.79E + 01**	3.54E + 01	**2.33E + 02**	3.72E + 02
F18	**2.07E + 01**	2.24E + 01	**2.35E + 01**	2.83E + 01
F19	**5.92E + 00**	7.20E + 00	**1.53E + 01**	2.44E + 01
F20	**2.96E + 01**	8.63E + 01	**1.22E + 02**	2.35E + 02
F21	**2.07E + 02**	2.16E + 02	**2.13E + 02**	2.39E + 02
F22	**1.00E + 02**	**1.00E + 02**	2.28E + 03	**1.00E + 02**
F23	**3.51E + 02**	3.59E + 02	**4.31E + 02**	4.50E + 02
F24	**4.25E + 02**	4.32E + 02	**5.08E + 02**	5.27E + 02
F25	**3.87E + 02**	3.87E + 02	**4.80E + 02**	5.38E + 02
F26	9.14E + 02	**7.36E + 02**	1.11E + 03	**3.00E + 02**
F27	**5.05E + 02**	5.06E + 02	5.32E + 02	**5.31E + 02**
F28	**3.15E + 02**	3.18E + 02	**4.59E + 02**	4.63E + 02
F29	**4.35E + 02**	4.42E + 02	**3.47E + 02**	3.70E + 02
F30	**1.97E + 03**	1.99E + 03	**6.21E + 05**	6.44E + 05

$$\overrightarrow{v}_{i,G} = m{*}(1 - BP) + BP{*}\overrightarrow{x}_{i,G} + F\left(\overrightarrow{x}_{r_1,G} - \overrightarrow{x}_{r_2,G}\right) + (1 - BP){*}\sigma{*}N(0, C) \tag{6}$$

The balance factor BP is dynamically increased from 0.5 to 1, and the generation process is as follows.

$$BP = 0.5{*}\left(1 + \frac{nfes}{nfes_{max}}\right)$$

The balance factor BP is mainly used to balance the mean and the target vector in the mutation strategy. Different stages of the algorithm have an appropriate trade-off between exploration and exploitation capabilities. All vectors are disturbed by approximately the identical

Table 2
Rankings obtained through Wilcoxon's test.

D	DCMAC VS	R^+	R^-	$p-value$	+	≈	-	$\alpha = 0.05$	$\alpha = 0.1$
30	DCMAC_1	275.00	25.00	0.000355	21	5	3	**Yes**	**Yes**
50	DCMAC_1	293.00	258.00	0.002840	22	3	4	**Yes**	**Yes**

Table 3
The results for $D = 30$ and $D = 50$.

Func.	30D		50D	
	DCMAC	DCMAC-SPA	DCMAC	DCMAC-SPA
F1	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
F3	**0.00E + 00**	2.11E + 00	**0.00E + 00**	**0.00E + 00**
F4	**5.63E + 01**	5.93E + 01	**4.72E + 01**	4.81E + 01
F5	**7.96E + 00**	1.21E + 01	**1.17E + 01**	2.88E + 01
F6	**5.37E-09**	4.63E-08	**0.00E + 00**	2.96E-06
F7	**3.86E + 01**	4.42E + 01	**6.42E + 01**	8.04E + 01
F8	**8.09E + 00**	1.24E + 01	**1.16E + 01**	2.79E + 01
F9	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
F10	**1.41E + 03**	2.11E + 03	**3.14E + 03**	3.35E + 03
F11	1.37E + 01	**1.31E + 01**	**2.58E + 01**	2.65E + 01
F12	**2.26E + 02**	3.57E + 02	**1.21E + 03**	1.40E + 03
F13	**1.50E + 01**	1.98E + 01	**3.13E + 01**	4.44E + 01
F14	2.13E + 01	**2.09E + 01**	**2.48E + 01**	2.61E + 01
F15	**2.42E + 00**	3.55E + 00	**2.13E + 01**	2.41E + 01
F16	**2.64E + 01**	4.68E + 01	**2.76E + 02**	3.34E + 02
F17	**2.79E + 01**	2.81E + 01	**2.33E + 02**	2.96E + 02
F18	**2.07E + 01**	2.17E + 01	**2.35E + 01**	2.58E + 01
F19	**5.92E + 00**	6.68E + 00	**1.53E + 01**	1.86E + 01
F20	**2.96E + 01**	7.42E + 01	**1.22E + 02**	1.82E + 02
F21	**2.07E + 02**	2.14E + 02	**2.13E + 02**	2.31E + 02
F22	**1.00E + 02**	**1.00E + 02**	2.28E + 03	**1.00E + 02**
F23	**3.51E + 02**	3.58E + 02	**4.31E + 02**	4.43E + 02
F24	**4.25E + 02**	4.29E + 02	**5.08E + 02**	5.14E + 02
F25	**3.87E + 02**	3.87E + 02	**4.80E + 02**	5.38E + 02
F26	9.14E + 02	**6.59E + 02**	1.11E + 03	**3.00E + 02**
F27	5.05E + 02	**5.05E + 02**	5.32E + 02	**5.26E + 02**
F28	3.15E + 02	**3.00E + 02**	**4.59E + 02**	4.65E + 02
F29	4.35E + 02	**4.33E + 02**	**3.47E + 02**	3.50E + 02
F30	**1.97E + 03**	1.98E + 03	6.21E + 05	**6.17E + 05**

mean vector when the algorithm iterates to a later stage. Therefore, the incorporation of the target vector is assisted to set the path of evolution reasonably and eliminate the stagnation of the population.

In mutation strategy, the historical experience of the population is an important learning source. In addition, different individuals in different states need different mutation strategies to improve themselves. For these reasons, a knowledge-based reward mechanism is introduced to enhance the two cooperative strategies. The class probability variable (*FCP*) is randomly selected from the M_{FCP}. M_{FCP} is updated at the end of each generation according to the performance of the two mutation strategies. Therefore, potential individuals are assigned to the mutation strategies with superior performance. M_{FCP} is updated as follows.

$$M_{FCP,G+1} = (1-c)M_{FCP,G} + c\Delta_{Alg1} \tag{7}$$

where c is the learning probability and Δ_{Alg1} is expressed as the improvement rate of the algorithm. The M_{FCP} is a storage pool. The M_{FCP} is a probability variable. The individuals that each generation generated by Eq.(5) and Eq. (6) are put inM_{FCP}. These individuals are compared to the parent individuals. The way to compare is by the following formula.

$$\Delta_{Alg1} = min\left(0.8, max\left(0.2, \frac{\omega_{Alg1}}{\omega_{Alg1}+\omega_{Alg2}}\right)\right)\omega_{Alg1} = \sum_{i=1}^{n} f(x) - f(u)$$

where the minimum probability of the algorithm is 0.2 and the maximum probability is 0.8. Therefore, the *FCP* is always maintained between (0.2, 0.8) to execute Eq.(5) and Eq.(6) simultaneously. $\omega_{Alg1}$is the summation of differences between old and new fitness values for each individual belongs algorithm $Alg1$.ω_{Alg2} is the summation of differences between old and new fitness values for each individual belongs algorithm $Alg2$. f is the fitness function. × is the old individual. u is the new individual.

In DCMAC, two different mutation strategies are adaptively selected as shown in Eq. (7). The probability variable M_{FCP} is adjusted according to the fitness value of the parent and a majority of individuals are assigned to the mutation strategies with desirable performance. If the fitness value of the individual obtained by Eq.(5) is better than the fitness value of the individual obtained by Eq. (6), Eq. (7) is calculated according to fitness value. According to the result of Eq. (7), the bulk individuals are allocated in the next iteration to reward Eq. (5) and a small amount of individuals are allocated in the next iteration to punish Eq. (6). Conversely, Eq. (6) is rewarded and Eq. (5) is punished. This cooperative method effectively improves the search accuracy of the algorithm. The adaptive selection of the weighted mutation strategy and the CMA-ES sampling process are shown in Fig. 2.

3.3. Improved parameter adaptation learning mechanism

In DCMAC, the scaling factor F is adjusted by the sinusoidal formula

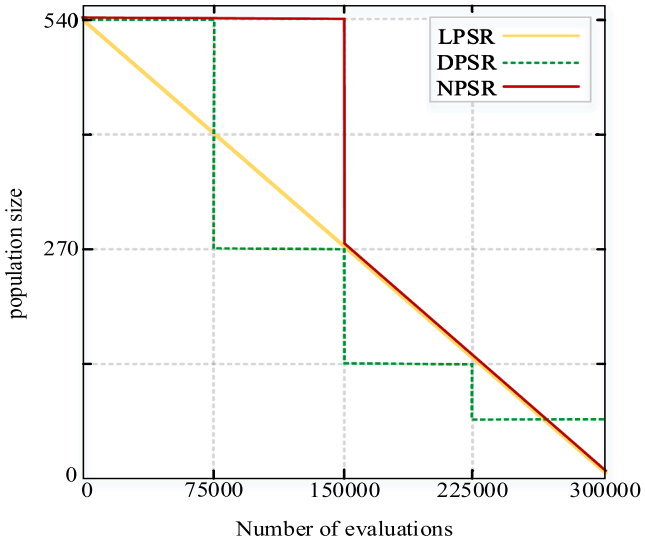


Fig. 4. Population size reduction process.

Table 4
Rankings obtained through Wilcoxon's test.

D	DCMAC VS	R^+	R^-	$p-value$	+	≈	-	$\alpha = 0.05$	$\alpha = 0.1$
30	DCMAC-SPA	243.00	57.00	0.007877	19	5	5	Yes	Yes
50	DCMAC-SPA	266.00	85.00	0.021506	22	3	4	Yes	Yes

Table 5
The results for $D = 30$ and $D = 50$.

	30D			50D		
Func.	DPSR	LPSR	NPSR	DPSR	LPSR	NPSR
F1	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
F3	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
F4	**4.71E + 01**	5.17E + 01	5.63E + 01	**4.49E + 01**	5.05E + 01	4.72E + 01
F5	1.40E + 01	1.10E + 01	**7.96E + 00**	1.33E + 01	1.21E + 01	**1.17E + 01**
F6	6.71E-09	**2.68E-09**	5.37E-09	6.35E-07	3.29E-08	**0.00E + 00**
F7	3.70E + 01	**3.51E + 01**	3.86E + 01	5.86E + 01	**5.65E + 01**	6.42E + 01
F8	1.49E + 01	1.02E + 01	**8.09E + 00**	1.83E + 01	1.77E + 01	**1.16E + 01**
F9	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
F10	1.66E + 03	1.54E + 03	**1.41E + 03**	3.32E + 03	3.18E + 03	**3.14E + 03**
F11	**1.33E + 01**	1.63E + 01	1.37E + 01	2.68E + 01	2.63E + 01	**2.58E + 01**
F12	3.27E + 02	2.79E + 02	**2.26E + 02**	1.41E + 03	1.29E + 03	**1.21E + 03**
F13	1.53E + 01	**1.24E + 01**	1.50E + 01	3.64E + 01	**2.93E + 01**	3.13E + 01
F14	2.13E + 01	**2.12E + 01**	2.13E + 01	2.43E + 01	**2.42E + 01**	2.48E + 01
F15	**2.11E + 00**	2.11E + 00	2.42E + 00	2.46E + 01	2.29E + 01	**2.13E + 01**
F16	6.55E + 01	3.41E + 01	**2.64E + 01**	3.39E + 02	3.20E + 02	**2.76E + 02**
F17	3.32E + 01	**2.72E + 01**	2.79E + 01	2.37E + 02	**2.31E + 02**	2.33E + 02
F18	2.19E + 01	2.14E + 01	**2.07E + 01**	2.65E + 01	2.49E + 01	**2.35E + 01**
F19	8.04E + 00	7.40E + 00	**5.92E + 00**	1.84E + 01	1.70E + 01	**1.53E + 01**
F20	3.81E + 01	3.83E + 01	**2.96E + 01**	1.27E + 02	**1.14E + 02**	1.22E + 02
F21	2.14E + 02	2.12E + 02	**2.07E + 02**	2.45E + 02	2.26E + 02	**2.13E + 02**
F22	**1.00E + 02**	**1.00E + 02**	**1.00E + 02**	9.17E + 02	**8.81E + 02**	2.28E + 03
F23	3.58E + 02	3.56E + 02	**3.51E + 02**	4.49E + 02	4.41E + 02	**4.31E + 02**
F24	4.31E + 02	4.28E + 02	**4.25E + 02**	5.28E + 02	5.11E + 02	**5.08E + 02**
F25	3.87E + 02	3.87E + 02	**3.87E + 02**	4.82E + 02	4.80E + 02	**4.80E + 02**
F26	9.89E + 02	9.61E + 02	**9.14E + 02**	1.37E + 03	1.18E + 03	**1.11E + 03**
F27	5.05E + 02	5.06E + 02	**5.05E + 02**	5.29E + 02	**5.28E + 02**	5.32E + 02
F28	3.22E + 02	3.26E + 02	**3.15E + 02**	**4.59E + 02**	**4.59E + 02**	**4.59E + 02**
F29	4.44E + 02	**4.35E + 02**	4.35E + 02	3.80E + 02	3.50E + 02	**3.47E + 02**
F30	1.98E + 03	1.99E + 03	**1.97E + 03**	6.37E + 05	6.30E + 05	**6.21E + 05**

of the adaption learning mechanism and the Cauchy distribution (Noor et al., 2017). The first configuration $F_{i,g}$ is a non-adaptive sinusoidal reduction adjustment according to Eq. (8).

$$F_{i,G} = \frac{1}{2} * \left(sin(2\pi * freq * G + \pi) * \frac{G_{max} - G}{G_{max}} + 1 \right) \tag{8}$$

where $freq = 0.5$ is set to a fixed value. The sinusoidal increment is adaptively adjusted according to Eq. (9).

$$F_{i,G} = \frac{1}{2} * \left(sin(2\pi * freq_i * G) * \frac{G_{max} - G}{G_{max}} + 1 \right) freq_i = randc(\mu freq_i, 0.1) \tag{9}$$

where $freq_i$ is an adaptive frequency generated by using the Cauchy distribution, G_{max} is the maximum number of iterations, $\mu freq_i$ is randomly selected from the external memory M_{freq}, the average frequency of success of the previous generations in S_{freq} is stored in M_{freq}.

When $G \leq \frac{G_{max}}{2}$, a hybrid sinusoidal configuration is selected. In the sinusoidal pool, different methods are selected according to the previous experience. The success vectors are generated by each sinusoidal configuration entering the next generation and record by $ns_{G,k}$. If $G \leq 20$, one of the sinusoidal formulas is randomly selected. Otherwise, the probability P_k of is calculated by the following equation.

$$P_k = \frac{S_{k,G}}{\sum_{k=1}^{K} S_{k,G}} \tag{10}$$

$$S_{k,G} = \frac{\sum_{i=G-LP}^{G-1} ns_{k,i}}{\sum_{i=G-LP}^{G-1} ns_{k,i} + \sum_{i=G-LP}^{G-1} nf_{k,i}} + \theta$$

where P_k is the probability of selecting the two sine formulas, $S_{k,G}$ represents the success rate of the test vector. To avoid empty success rate, θ is added and set to 0.01.

When $G > \frac{G_{max}}{2}$, according to Eq. (11), the Cauchy distribution is used to adjust the proportion factor. The crossover rate Cr_i is the same as LSHADE-SPACMA and is adjusted by the normal distribution, according to Eq. (12).

$$F_i = randc(MF_i, \sigma) \tag{11}$$

where $\sigma = 0.1$ is the standard deviation of the Cauchy distribution

Table 7
ANOVA results for parameter settings of DCMAC.

Source	Sum of squares	Degrees of freedom	Mean Square	*F*-ratio	*p*-value
M_{Cr}	5836063.6	3	1945354.5	7.89	**0.0001**
M_F	960580.5	3	320193.5	1.3	0.2794
σ	125702.5	2	62851.2	0.25	0.7755
$\mu freq$	5745677.8	2	2872838.9	11.65	0
$M_{Cr} * M_F$	716147.6	9	79,572	0.32	0.9657
$M_{Cr} * \sigma$	1645905.8	6	274317.6	1.11	0.3608
$M_{Cr} * \mu freq$	4102393.4	6	683732.2	2.77	**0.0157**
$M_F * \sigma$	1079773.4	6	179962.2	0.73	0.6266
$M_F * \mu freq$	1593639.7	6	265606.6	1.08	0.3816
$\sigma * \mu freq$	485798.5	4	121449.6	0.49	0.7412
Residual	23,668,884	96	246550.9		
Total	45960566.7	143			

Table 6
Rankings obtained through Wilcoxon's test.

D	NPSR VS	R^+	R^-	$p-value$	+	≈	-	$\alpha = 0.05$	$\alpha = 0.1$
30	LPSR	229.00	47.00	0.005636	16	6	7	**Yes**	**Yes**
	DPSR	223.00	30.00	0.001726	18	7	4	**Yes**	**Yes**
50	LPSR	212.00	88.00	0.076459	17	5	7	No	**Yes**
	DPSR	271.50	53.50	0.003357	20	4	5	**Yes**	**Yes**

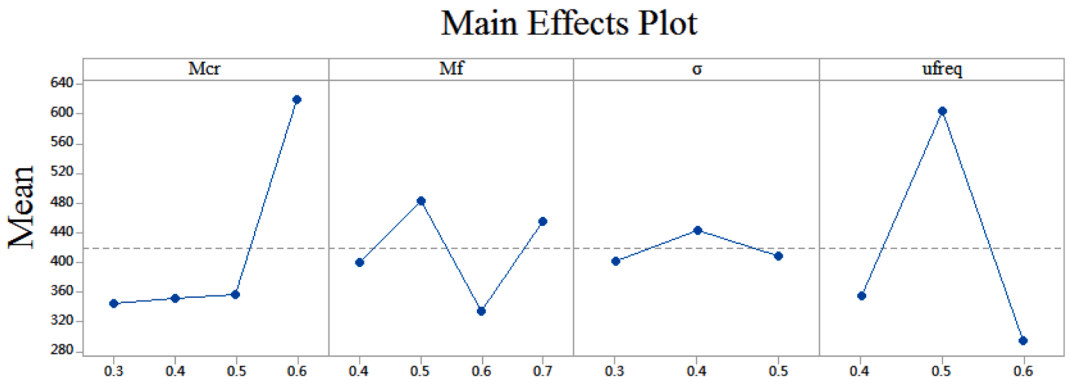


Fig. 5. Main effects plot of parameters.

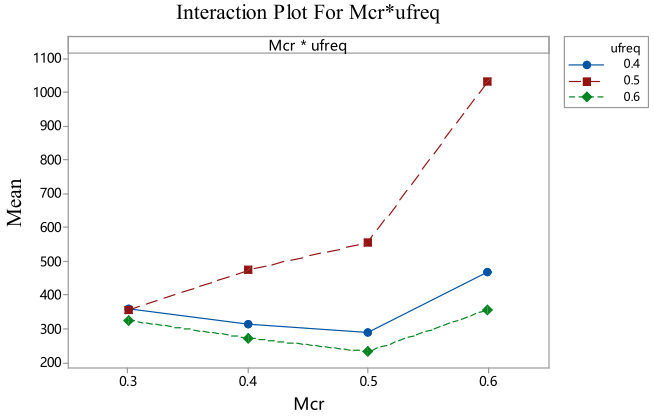


Fig. 6. Interaction plot of parameters.

and MF_i is the memory that stores the previous generations of successful methods.

$$Cr_i = randn(M_{cr_i}, 0.1) \tag{12}$$

where M_{cr_i} is used to store the memory of the previous generation of successful methods, and the initial value set to 0.5.

$$MF_{k,G+1} = mean_{WL}(S_F)$$

$$Mcr_{k,G+1} = mean_{WL}(S_{cr})$$

$$mean_{WL}(S) = \frac{\sum_{k=1}^{|S|} w_k \bullet S_k^2}{\sum_{k=1}^{|S|} w_k \bullet S_k}$$

$$w_k = \frac{\Delta f_k}{\sum_{j=1}^{|S|} \Delta f_j}$$

where $\Delta f_k = \left| f(U_{k,G}) - f(X_{k,G}) \right|$. The pseudo-code of improved parameter adaptation learning mechanism is given in Algorithm 2.

Algorithm 2. Parameter adaptation learning mechanism

```
if G ≤ G_max/2
  if G ≤ LP
    The two sinusoidal configurations have the same probability, P1 = P2 = 0.5
  else
    Adjust F_{i,G} using Eq.(8) or Eq.(9) with the greatest probability
  end
  if G > G_max/2
    Generate F_i using the Cauchy distribution, as inEq.(11)
  end
```

3.4. Niching population size reduction (NPSR)

Linear population reduction mechanism is adopted in various variant algorithms, such as LASDHE and LSHADE-SPACMA. However, the linear population reduction mechanism has two drawbacks. First, if the population is linearly reduced from the initial stage of the iteration, the deleted individuals do not have enough time to evolve. Second, the current population reduction method not only reduces the population size affecting the diversity of the population but also increases the possibility of premature convergence. Detailed contents about population reduction are found in Awad et al. (2018).

The population is not reduced to provide all individuals with the time required for evolution when $nfes < \frac{nfes_{max}}{2}$. In addition, two separate niches are formed before the population begins to shrink. This process aims to select the areas effectively that need to be eliminated during the search. Moreover, the Euclidean distance between the best individual and the other individuals is calculated according to Eq. (13).

$$ED_i = ED_i\left(\vec{X}_{i,G}, \vec{X}_{best,G}\right) = \sqrt{\left(\vec{x}^1_{i,G} - \vec{x}^1_{best,G}\right)^2 + \left(\vec{x}^2_{i,G} - \vec{x}^2_{best,G}\right)^2 + \cdots + \left(\vec{x}^D_{i,G} - \vec{x}^D_{best,G}\right)^2} \tag{13}$$

where ED_i is the Euclidean distance between the individual $\vec{X}_{i,G}$ and the best individual $\vec{X}_{best,G}$. Afterward, the process, as shown in Fig. 3, is used to illustrate that the individuals are ranked by the Euclidean distance. The best individual and his neighbour are stored in the first niche. The remaining individuals are stored in the second niche. The two niches are set to $N/2$ respectively and decrease as the population decreases.

The population of DCMAC is linearly decreased, when $nfes \geq \frac{nfes_{max}}{2}$ according to Eq.(14). The purpose of selecting the individuals with the smallest Euclidean distance from the optimal individual for eliminating is to delete the neighbourhood solution and redundant solution of the optimal individuals preferentially.

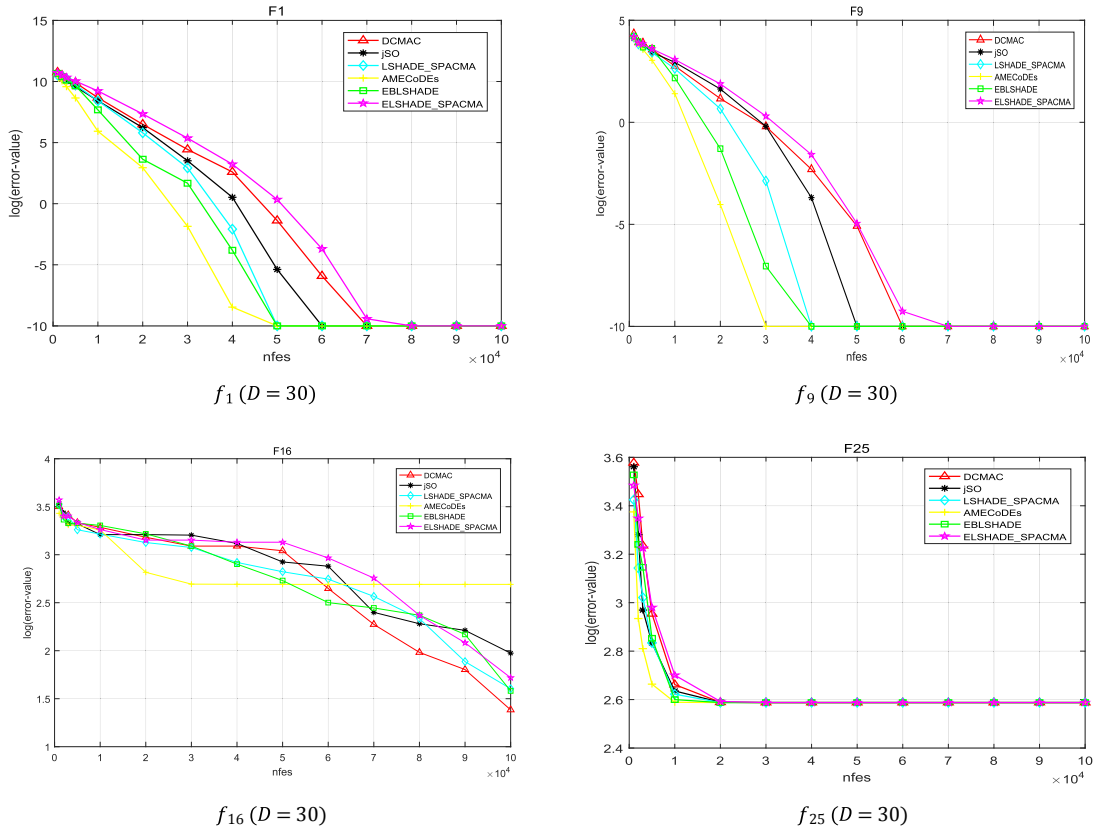
$$N_{G+1} = round\left[\left(\frac{N_{min} - N_{init}}{max\_nfes}\right) * nfes + N_{init}\right] \tag{14}$$

where N_{init} is the initial population size and $N_{min} = 4$ is reduced to the minimum number of individuals. The pseudo-code of population size reduction is shown in Algorithm 3.

Algorithm 3. Population size reduction

```
if nfes > nfes_max/2
  Arranged in ascending order according to individual ED_i
  Fill the first niche with the X_best,G and its best neighbors based on ED_i with size N/2
  The second niche stores the remaining individuals
```

(continued on next page)



$f_1\ (D=30)$ $f_9\ (D=30)$

$f_{16}\ (D=30)$ $f_{25}\ (D=30)$

Fig. 7. Convergence plots of the benchmark functions (30D).

(*continued*)

	Algorithm 3. Population size reduction
5	Calculate the scale of the next generation population $N_{(G+1)}$ using Eq.(14)
6	Pick the best individual in the second niche and mark the neighborhood as the elimination area.
7	Eliminate N_{diff} individuals from the marker elimination region of the second niche
8	**end**

4. Convergence analysis of DCMAC

In this paper, three strategies (mechanism of the niching population size reduction, cooperative mutation strategy and adaptive parameter learning mechanism) are adopted in the DCMAC algorithm. Three strategies are applied to enhance the performance of DCMAC and avoid being fallen into local optimal and premature solution sets. The proposed knowledge-based reward mechanism assigns potential individuals to mutation strategies with prominent performance. Therefore, the convergence of DCMAC is not affected by CMA-ES, and the convergence of LSHADE is equivalent to the convergence of DCMAC (Hu et al., 2014). In this section, the convergence of DCMAC is demonstrated.

As based on the population random optimization algorithm, the evolution process of the DCMAC algorithm is mapped to a stochastic process, and the three basic operations of the algorithm are regarded as random mappings. The individuals search space is assumed to be $S = R^D$, S^N denotes the population space, and$f : S - R^+$ denotes the fitness function.

Step 1. The three different individuals are selected randomly in the population to generate new individual using Eq.(5). In a sense, it is randomly mapped $T_m : S^N \to S$. The probability distribution is as follows.

$$\begin{aligned} &P(T_m(X) = v_i) \\ &\quad = \sum_{x_{r_1}, x_{r_2}, x_{r_3} \in S^3} P(T_m^1(X) = \{x_{r_1}, x_{r_2}, x_{r_3}\}) \cdot P(T_m^2(x_{r_1}, x_{r_2}, x_{r_3}) = x_i \\ &\quad = \sum_{x_{r_1}, x_{r_2}, x_{r_3} \in S^3} P(T_m^1(X) = \{x_{r_1}, x_{r_2}, x_{r_3}\}) \end{aligned} \tag{15}$$

Step 2. The crossover operation of the DCMAC algorithm randomly generates new individuals by the variant individual v_i and the original target individual x_i. Therefore, it is also a random mapping, which is expressed as $T_c : S^2 \to S$. The probability distribution is:

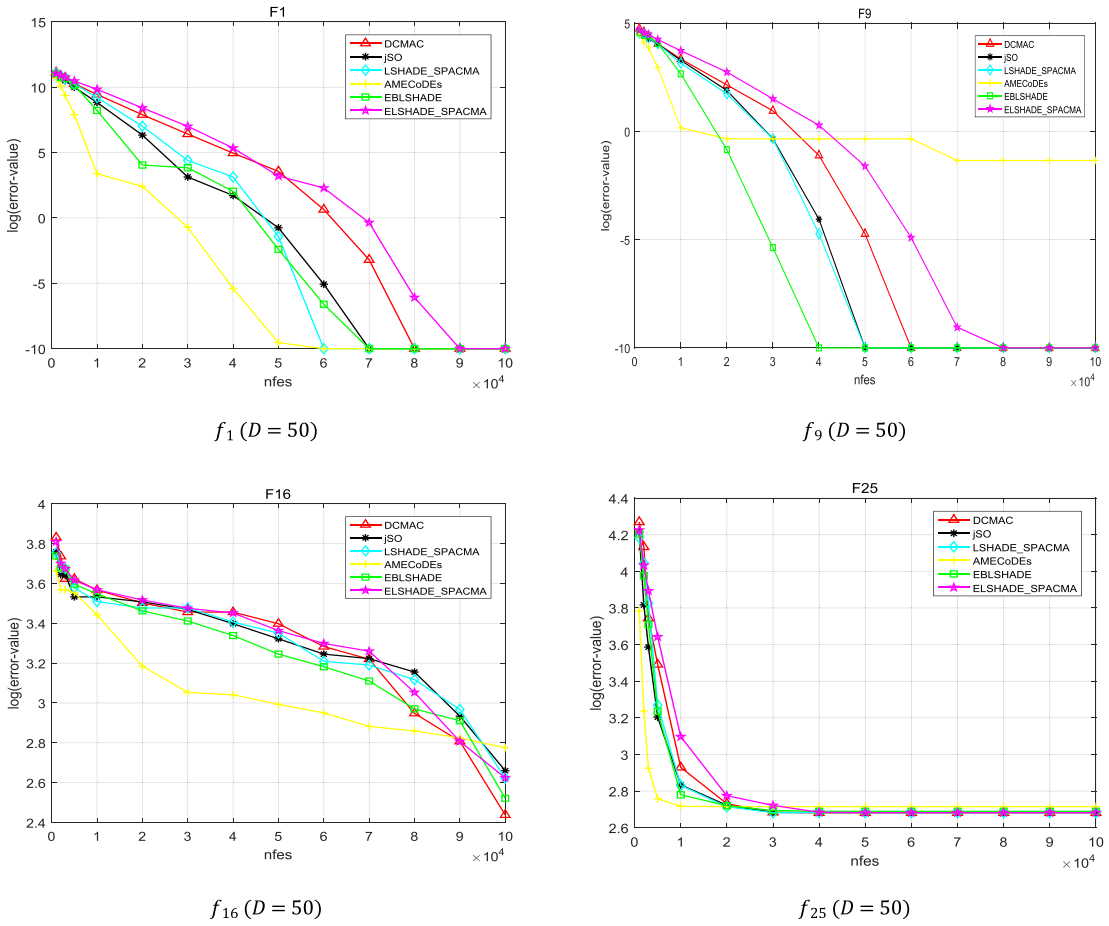
$$P(T_c(x_i, v_i) = u_i) = \begin{cases} 0, u_i = x_i \\ m \bullet C_D^k CR^k (1 - CR)^{D-k}, otherwise \\ CR^N + \dfrac{1}{N}, u_i = v_i \end{cases} \tag{16}$$

$k$is the number of elements that produce intersection, where $1 < k < D$. m is the total number of new individuals u_i generated by intersection.

Step 3. The selection operator is selected from the newly generated individual u_i and the original target vector x_i to retain the population of the next generation with better fitness, which is recorded as $T_s : S^2 \to S$. The probability of population accepts a new individual for called a greedy operations choice.

$$P(T_s(x_i, u_i) = u_i) = \begin{cases} 1, f(u_i) \le f(x_i) \\ 0, otherwise \end{cases} \tag{17}$$

Potential individuals in the population are retained by the selection operator that the evolutionary direction does not retreat. In summary, the DCMAC algorithm is recorded as.



$f_1\ (D=50)$ $f_9\ (D=50)$

$f_{16}\ (D=50)$ $f_{25}\ (D=50)$

Fig. 8. Convergence plots of the benchmark functions (50D).

$$X(n+1)=\{x_i(n+1)=T_s\circ T_c\circ T_m(X(n)),i=1,\cdots,NP\}$$

Theorem 1. The evolutionary direction of populations in DCMAC algorithms is monotonous, $F(X(n+1))\le F(X(n))$.

Proof. According to Definition 3, the selection operator is a greedy selection operation, and the evolution direction of the population is monotonous and non-incremental.

Theorem 2. In DCMAC, the population sequence $\{X(n), n\in N^+\}$ is a homogeneous Markov chain.

Proof. The population sequence $X(N+1)=T(X(n))=Ts^{\circ}Tc^{\circ}Tm(X(n))$, Ts, Tc, Tm are independent of n. Therefore, $\{X(n);n\in N^+\}$ is the Markov chain, where

$$P(T(X(N))_i=x_i(n+1))$$

$$=\sum_{u_i\in S}\sum_{v_i\in S_{x_{r_1}}}\sum_{,x_{r_2},x_{r_3}}P\left(T_m^1(X(n))=\{x_{r_1},x_{r_2},x_{r_3}\}\right)\bullet P(T_C(x_i(n),v_i))$$

$$=u_i)\bullet P(T_s(x_i(n),u_i)=x_i(n+1))$$

Therefore, $P\left(T(X(n))_i=x_i(n+1)\right)\rangle 0$. The probability of transition is recorded as

$$P(T(X(n))=X(n+1))=\prod_{i=1}^{NP}P(T(X(n))_i=x_i(n+1))>0$$

Therefore, the population sequence $\{X(n);n\in N^+\}$ of the DCMAC algorithm is a homogeneous irreducible aperiodic Markov chain on S^{NP}. The probability of transition is recorded as

$$P(X(n+1)=Y|X(n)=X)$$

$$=\left\{\prod_{i=1}^{NP}P(T(X(n))_i=x_i(n+1),\exists i_x,i_y\in[1,NP],s.t.f(x_{ix})\right.$$

$$=F(x),f\left(y_{iy}\right)=F(Y)0,otherwise$$

Theorem 3. *The Markov chain population sequence* $\{X(n);n\in N^+\}$ *of the DCMAC algorithm converges with probability* 1 *to the subset* M_0 *of the satisfying species cluster* M^* *in the solution space*: $M_0^*=\{Y=(y_1,\cdots,y_{NP})|y_i\in M^*\}$, *namely*

$$\lim_{n\to\infty}P(X(n)\in M_0^*|X(0)=X_0)=1$$

Proof. *Let* X^* *is the only optimal satisfactory solution for* $f(x)$. *According to* Eq.(15)*and* Eq.(16), $P(X,Y)$ *has the following properties*.

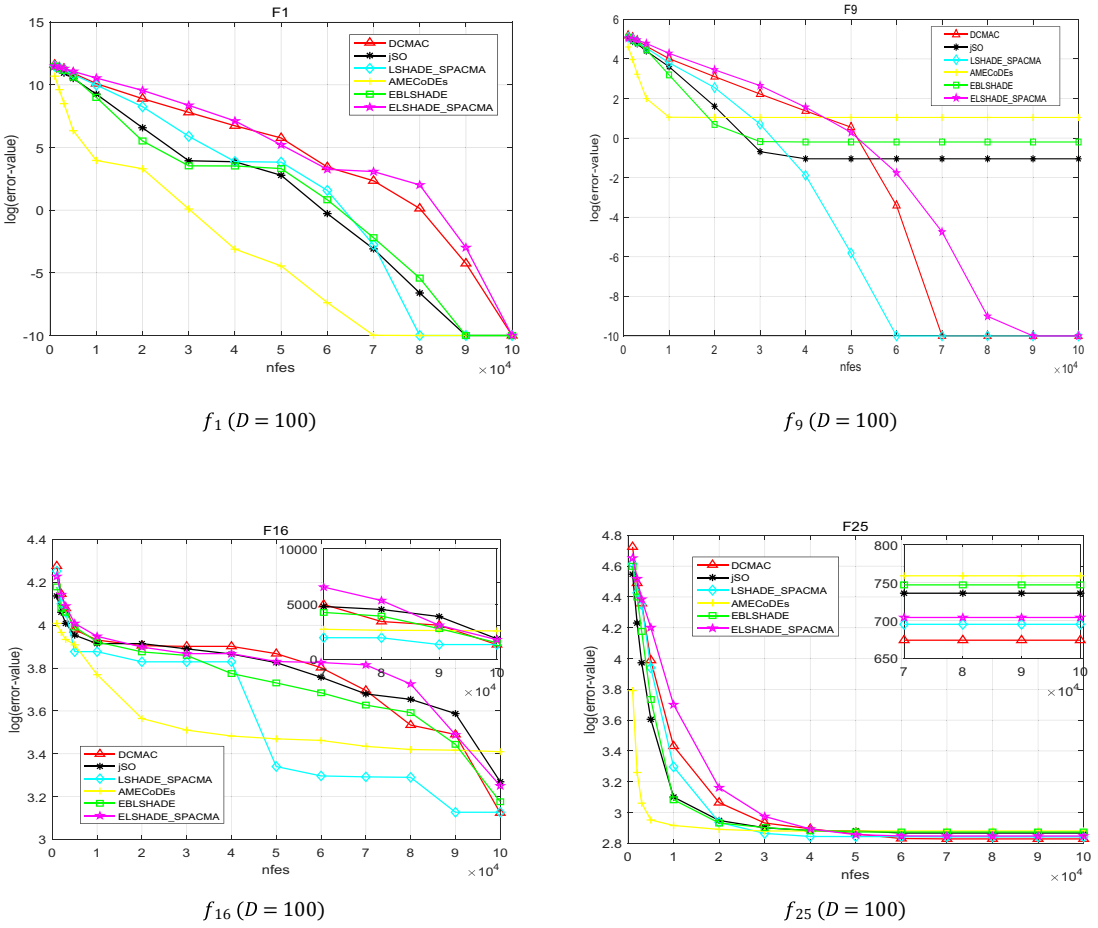
(1) When $X,Y\in M_0^*,P(X,Y)>0,P(Y,X)>0$, two-state interworking $X\leftrightarrow Y$.

(2) When $X\in M_0^*,Y\notin M_0^*,P(X,Y)=0$, scilicet $X\nrightarrow Y$.

Therefore, the population sequence of DCMAC is a non-periodic homogeneous Markov chain.

$$\lim_{n\to\infty}P(X(n)=Y|X(0)=X_0)=\begin{cases}\pi(Y),Y\in M_0^*\\0,Y\notin M_0^*\end{cases}$$

Then, $X(n)$ must be entering M_0^* to satisfy a certain limit probability



$f_1\ (D = 100)$ $f_9\ (D = 100)$

$f_{16}\ (D = 100)$ $f_{25}\ (D = 100)$

Fig. 9. Convergence plots of the benchmark functions (100D).

Table 8
Rankings obtained through the Wilcoxon signed-rank test.

D	DCMAC VS	R^+	R^-	p-value	+	≈	-	$\alpha = 0.05$	$\alpha = 0.1$
30	EBLSHADE	128.50	124.50	0.948218	10	7	12	No	No
	AMECoDEs	293.00	7.00	0.000044	22	5	2	**Yes**	**Yes**
	LSHADE-SPACMA	217.00	59.00	0.016259	17	6	6	**Yes**	**Yes**
	jSO	142.00	111.00	0.614695	13	7	9	No	No
	ELSHADE-SPACMA	252.00	48.00	0.003563	18	5	6	**Yes**	**Yes**
50	EBLSHADE	271.00	54.00	0.003507	20	4	5	**Yes**	**Yes**
	AMECoDEs	356.00	22.00	0.000060	25	2	2	**Yes**	**Yes**
	LSHADE-SPACMA	255.00	70.00	0.012806	20	4	5	**Yes**	**Yes**
	jSO	245.00	80.00	0.026417	20	4	5	**Yes**	**Yes**
	ELSHADE-SPACMA	203.50	93.50	0.106448	17	5	7	No	No
100	EBLSHADE	354.00	52.00	0.000584	22	1	6	**Yes**	**Yes**
	AMECoDEs	375.00	31.00	0.000090	26	1	2	**Yes**	**Yes**
	LSHADE-SPACMA	236.00	142.00	0.258721	16	2	11	No	No
	jSO	311.00	95.00	0.013905	19	1	9	**Yes**	**Yes**
	ELSHADE-SPACMA	227.00	151.00	0.361165	15	2	12	No	No

distribution $\pi(Y)$, thus

$$\lim_{n\to\infty} P(X(n) \in M_0^* | X(0) = X_0) = 1$$

According to Definition 1, DCMAC converges to the global optimum in probability 1.

5. Experiments and comparisons

In this paper, the CEC 2017 (Awad et al., 2017) benchmark functions are used for evaluated DCMAC and compared with other variant algorithms of LSHADE. Details information of the benchmark function is described in the paper by Awad et al. (2016). It is necessary to point out that f_2 does not participate in comparison in the paper because of its unstable behavior. All algorithms are programmed with MATLAB (2016b). The experiments run on a PC with 3.4 GHz Intel (R) Core i7-6700 CPU, 8 GB RAM and 64-bit OS to ensure the fairness of the algorithm. In addition, due to limited space, interested readers obtain the experimental results for all algorithms from online Supplemental

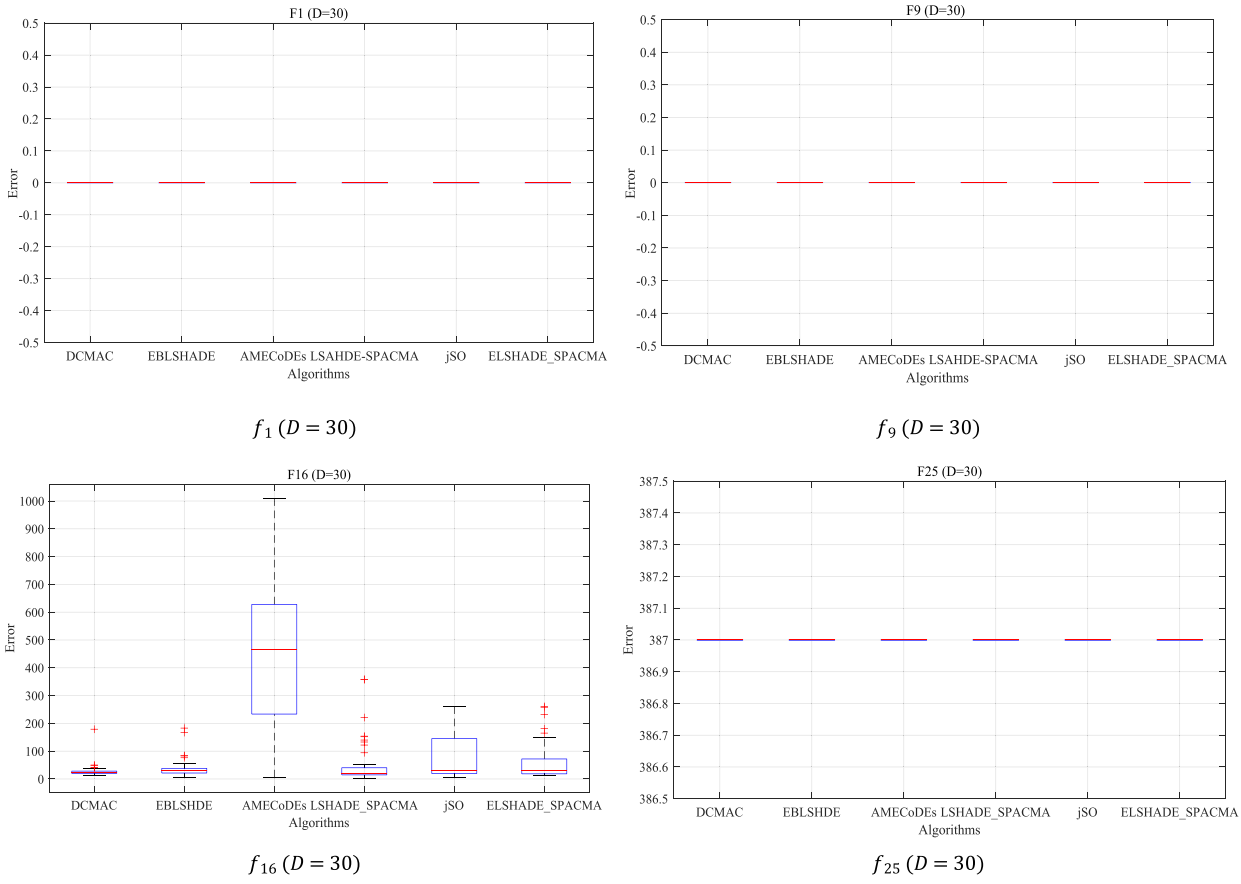


Fig. 10. Box plots of benchmark functions (30D).

material.

5.1. Effectiveness of knowledge-based cooperative mutation strategy

This section verifies the validity of the knowledge-based cooperative mutation strategies. The DCMAC, which has "DE/*current* −*to* −*pbest*/1" and random sampling of CMA-ES (DCMAC_1) is compared with the original DCMAC algorithm to verify that the component is beneficial to improve the search efficiency of the algorithm. The average error results of DCMAC and DCMAC_1 in different function types on D = 30, 50 are shown in Table 1, and the best result for each function is shown in boldface. From Table 1, the performance of DCMAC is better than DCMAC_1 on most functions. Under different dimensions, Wilcoxon's test results of DCMAC using different mutation strategies are shown in Table 2. R^+ represents the sum of ranks for the functions of the DCMAC better than the second mechanism in the row and R^- represents the sum of ranks the functions of the second mechanism superior to the DCMAC. It is worth noting that in pair-wise comparison, DCMAC is the first mechanism in the row. From Table 2, DCMAC is significantly better than DCMAC_1 in 30 and 50 dimensions when the confidence is $\alpha = 0.1$ and $\alpha = 0.05$.

The greedy degree of the algorithm is controlled with p value. When p is set to a comparatively large value, the greedy degree of the algorithm decreases and the search domain extends. Conversely, the greediness of the algorithm is increased by the small p value. Therefore, the "*DE*/*current* −*to* −*pBest* −*w*/1" with dynamic greedy p value is employed as mutation strategies in the DCMAC rather than the "*DE*/*current* −*to* −*pbest*/1". In addition, The DE mutation scheme based on the difference vector is combined with CMA-ES to overcome the limitations of CMA-ES, which can adapt to the natural scaling of the problem. The stagnation of the population is eliminated when using an appropriate percentage of the average vector and the current target vector. In DCMAC, the knowledge-based cooperative mutation strategies proposed are effective through the above experiments and analysis.

5.2. Effectiveness of parameter adaptation learning mechanism

This section verifies the validity of the parameter adaptation learning mechanism. The DCMAC, which has semi-parametric adaptive (DCMAC-SPA), is compared with the original DCMAC algorithm to verify the component that is beneficial to improve the search efficiency of the algorithm. The experimental results of DCMAC-SPA and DCMAC are shown in Table 3 to 4. From Table 4, it is found that DCMAC is significantly better than DCMAC-SPA on 29 benchmark functions with $\alpha = 0.1$ and $\alpha = 0.05$ for D = 30, 50. In DCMAC, the parameter adaptation learning mechanism proposed is effective through the results obtained from the above experiments and analysis.

5.3. Effectiveness of population size reduction

The deterministic population size reduction (DPSR), which used a fixed population size, is proposed by Janez and Mirjam (2008). After certain iterations, the population size is reduced to half of the original population size. DPSR mechanism contains two parameters. N_{start} is expressed as the initial population size, and P_{max} is the reduced number of the population size. When the population size is reduced to half ($P_{max} - 1$), x_i and $x_{i+N/2} (i = 1, 2, \cdots, N/2)$ are compared and the excellent values are preserved. Another classic linear population size reduction is proposed by Ryoji and Alex (2014). An algorithm using the LPSR mechanism reduces the population size linearly during the search period. LPSR mechanism has been used in several variants of the DE algorithm, such as jSO, LASHADE-SPACMA.

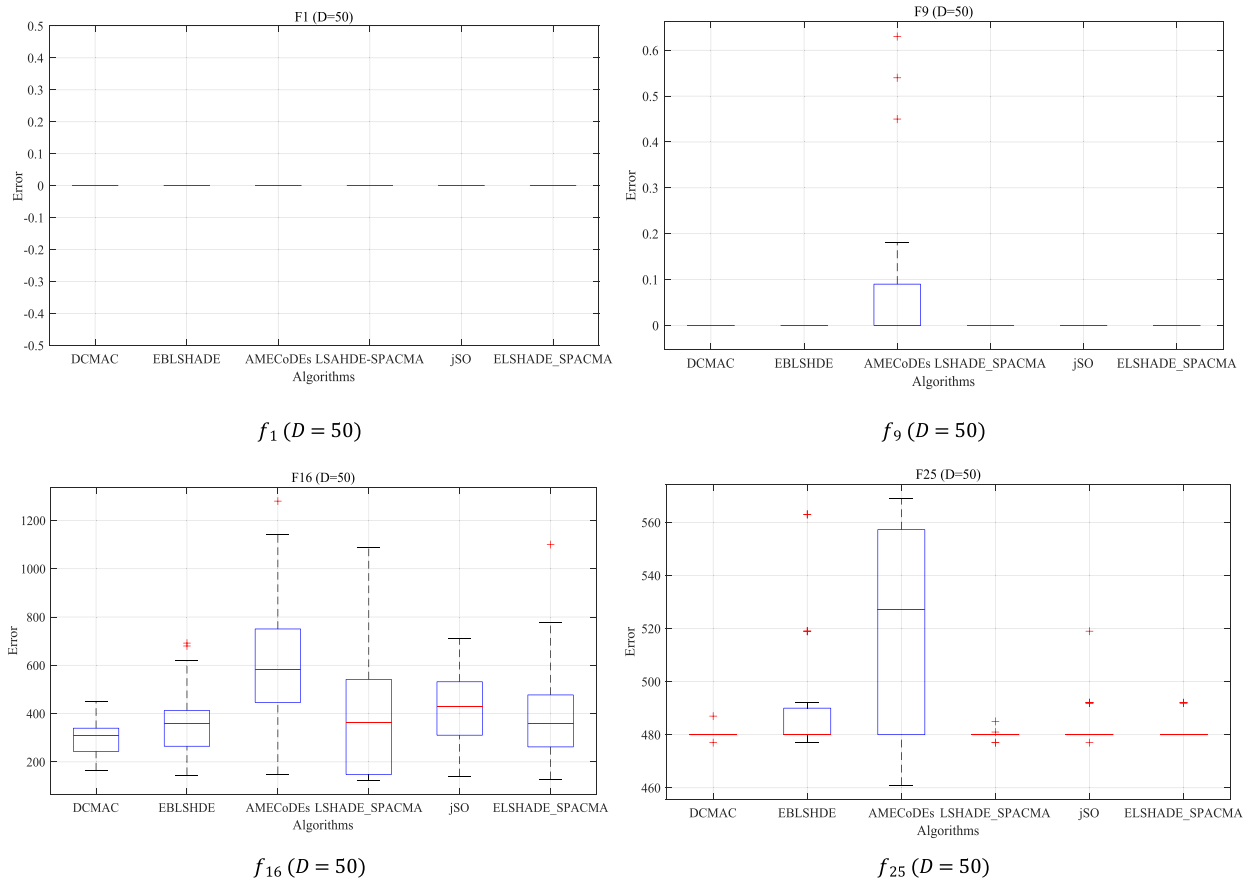


Fig. 11. Box plots of benchmark functions (50D).

At the beginning of the iteration, individuals are deleted by the two population reduction mechanisms mentioned above, which lead individuals to have insufficient time to evolve. According to Adam (2017), the desirable diversity of the population is not required for DE throughout the search process. The diversity of the population of the DE algorithm is required to reduce at the later stage to discover the optimal solution. In this paper, the mechanism of niching population size reduction is adopted. In the early stage of DCMAC, the population is not reduced to avoid losing the diversity of the population. According to function evaluations in the later stage, the local search ability of the DCMAC is improved by the linear descent of population.

In this section, the mechanism of niching population size reduction is described and compared with the other two mechanisms. Under the same operating conditions, DPSR, LPSR and NPSR are used in the algorithm, respectively. The process of three different population size reduction mechanisms at D = 30 is shown in Fig. 4. In order to evaluate the performance of NPSR in the proposed DCMAC algorithm, the average error of DCMAC on D = 30, 50 using the three different population reduction mechanisms mentioned are shown in Table 5.

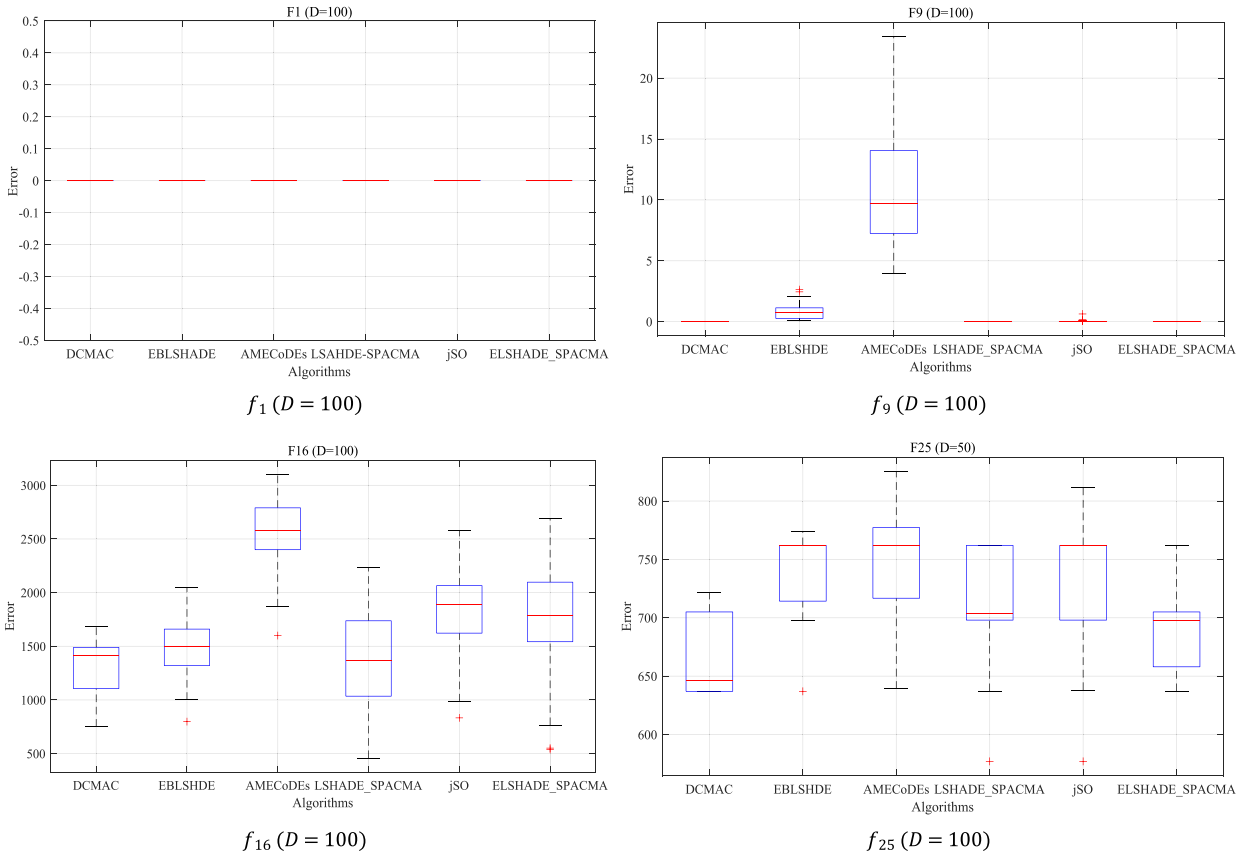
From Table 6, NPSR is significantly better than LPSR and DPSR in 30 and 50 dimensions when the confidence is $\alpha = 0.1$. Therefore, the NPSR is employed as a local search operator in the DCMAC rather than the LPSR and DPSR.

5.4. Parameters analysis

Parameter calibration experiment is critical, parameter sets have an important effect on the performance for each algorithm. In this paper, a set of suitable parameters for the DCMAC are determined by DOE experimental method. The DCMAC contains four critical parameters: M_{cr}(crossover rate selected from S_{cr}), M_F(mutation factor selected from S_F), $\mu freq$ (frequency), σ(Step size). The selected parameters are as follows:$M_{cr} \in \{0.3, 0.4, 0.5, 0.6\}$, $M_F \in \{0.4, 0.5, 0.6, 0.7\}$, $\sigma \in \{0.3, 0.4, 0.5\}$, $\mu freq \in \{0.4, 0.5, 0.6\}$. The results of all possible combinations of the above-selected parameters are$4 \times 4 \times 3 \times 3 = 144$. Each parameter combination is run 30 times independently to ensure that the experiment is statistically significant. Following the literature (Shao et al., 2019), the experimental results are analyzed by the multivariate analysis of variance (ANOVA). Whether the interaction between parameters is significant or not is expressed by variance analysis. The results of the ANOVA are shown in Table 7.

According to the results from Table 7, the *P-values* of parameters M_{Cr} and $\mu freq$ is less than the confidence level ($\alpha = 0.05$), indicating that these parameters are more sensitive than other parameters in DCMAC. Meanwhile, the parameter M_{Cr} corresponds to the greatest $F-ratio$. It suggests that the parameters M_{Cr} have the greatest effect on the average performance of DCMAC among all related factors. From Fig. 5, the parameters are selected as follows: $M_{Cr} = 0.3$, $M_F = 0.6$, $\sigma = 0.3$, and $\mu freq = 0.6$.

Furthermore, if the *P-value* between the two parameters is smaller than 0.05, the main effect plot is meaningless (Shao et al., 2019b). As shown in Table 7, the *p-value* of the parameter $M_{Cr}*\mu freq$ is 0.0157, smaller than 0.05. The interactions between the parameters M_F and the frequency $\mu freq$ are significantly demonstrated. From the interaction plot between M_F and $\mu freq$ in Fig. 6, the selection of $M_F = 0.6$ and $\mu freq = 0.6$ contribute to the best performing DCMAC. The interaction diagram is shown in Fig. 6, the selection of $M_F = 0.6$ and $\mu freq = 0.6$is contributed to obtaining the best performance of DCMAC. In summary, the following parameters are used in DCMAC: $M_{Cr} = 0.3$, $M_F = 0.6$, $\sigma = 0.3$, and $\mu freq = 0.6$.



$f_1\ (D = 100)$ $f_9\ (D = 100)$

$f_{16}\ (D = 100)$ $f_{25}\ (D = 100)$

Fig. 12. Box plots of benchmark functions (100D).

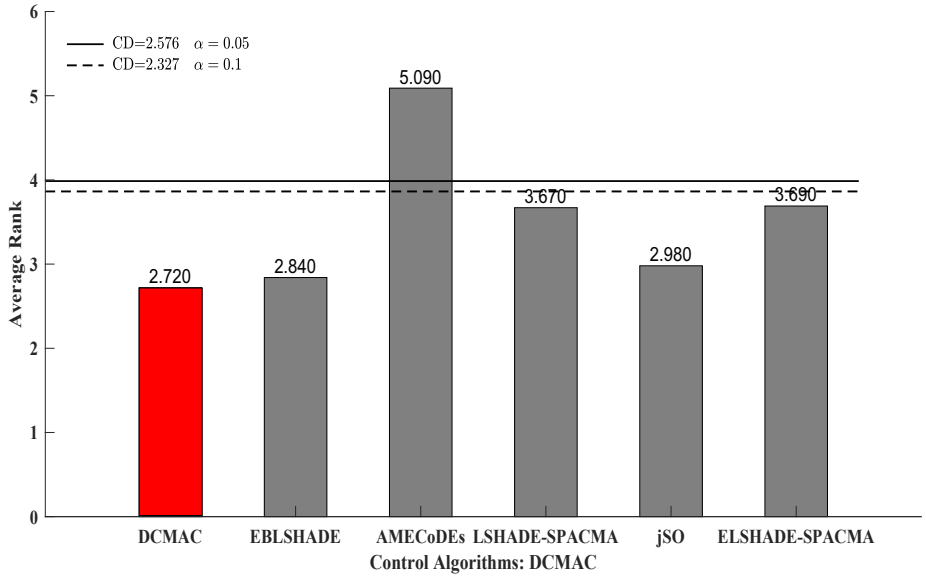


Fig. 13. Rankings for.D = 30.

5.5. Comparison of DCMAC with certain state-of-the-art algorithms

In this section, the proposed DCMAC is compared with five advanced DE variant algorithms. Five algorithms are briefly introduced as follows.

- LSHADE-SPACMA is a variant of the LSHADE algorithm, mainly using a semi-parametric adaptive method based on randomization and self-adaptation. In addition, a hybridization framework is introduced between the modified versions of LSHADE-SPA and CMA-ES.
- jSO is a variant of the iL-SHADE algorithm (Janez et al., 2016)that proposes a new weighted version of mutation strategy and ranks second in the CEC2017 test suite.

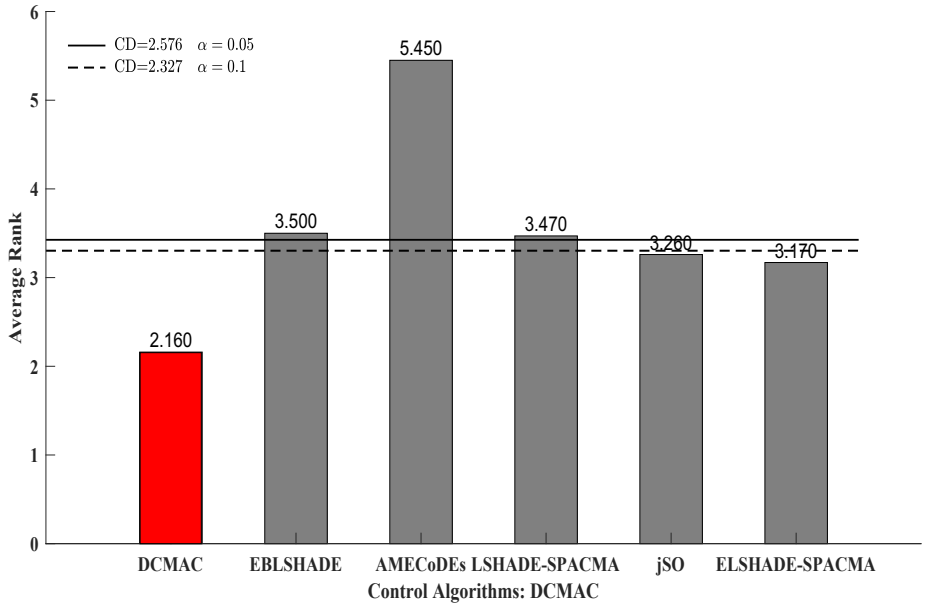


Fig. 14. Rankings for.D = 50.

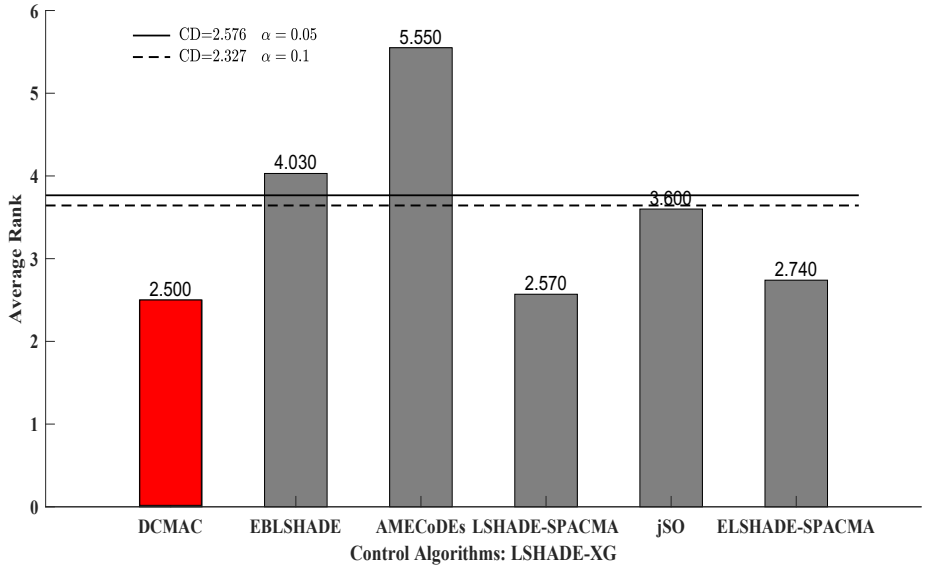


Fig. 15. Rankings for.D = 100.

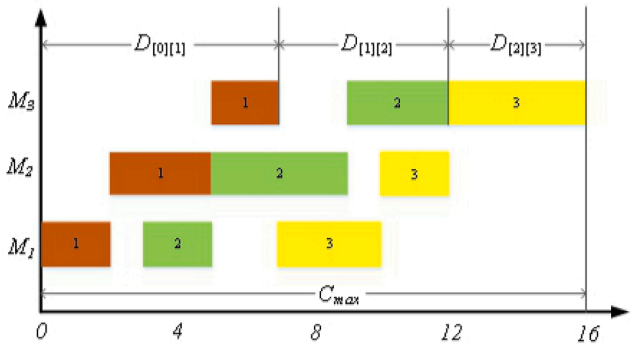


Fig. 16. No-wait flow shop scheduling Gantt chart.

Table 9
The principle of codec conversion.

	1	2	3	4	5	6
$x_{i,j}$	0.3	0.06	0.9	−0.24	0.85	−0.53
$\sigma_{i,j}$	4	3	6	2	5	1
$\pi_{i,j}$	6	4	2	1	5	3

- The main contribution of the EBLSHADE algorithm (Ali et al., 2018) is to propose two different mutation strategies and a hybrid framework. To perturb the target vector, the proposed two mutation strategies rank the three selected variables from the current generation. The proposed hybrid framework combines two mutation strategies with the DE family of algorithms.
- In the AMECoDEs algorithm (Cui et al., 2018), an adaptive multi-elite guided composite differential evolution algorithm with a

Table 10
ARPD values of the three algorithms.

$n*m$	IIGA	$DPSO_{VND}$	DCMAC	$n*m$	IIGA	$DPSO_{VND}$	DCMAC
20*5	**0.00**	0.01	**0.00**	100*5	2.07	0.20	**0.04**
20*10	0.32	0.32	**0.00**	100*10	0.98	**0.08**	0.23
20*20	0.03	0.05	**0.02**	100*20	0.73	**0.10**	0.35
50*5	0.87	0.05	**0.09**	200*10	1.45	0.09	**0.06**
50*10	0.24	**0.06**	0.10	200*20	0.97	0.53	**0.03**
50*20	0.19	**0.15**	0.18	500*20	1.58	0.37	**0.00**

Table 11
Rankings obtained through Wilcoxon's test.

DCMAC VS	R^+	R^-	Z	$p-value$	$\alpha = 0.05$	$\alpha = 0.01$
IIGA	4383.50	176.50	−7.808004	5.8101E-15	**Yes**	**Yes**
$DPSO_{VND}$	3264.00	1686.00	−2.753887	0.005889	**Yes**	**Yes**

shifting mechanism is proposed. This algorithm simultaneously generates two candidate solutions for each individual using two test vector generation strategies and adopting the best individuals to participate in the selection.

- ELSHADE-SPACMA is an improved variant of the LSHADE-SPACMA algorithm, primarily by integrating another directed mutation strategy within the original hybridization framework. On the CEC2018 test suite, ELSHADE-SPACMA ranked third overall.

The classical convergence curves of the comparison algorithm and DCMAC are shown in Figs. 7–9, and the certain local convergence curve is also listed. The selected functions are the convergence result of f_1,f_9, f_{16} and f_{25} on 30D, 50D and 100D. As shown in Figs. 7–9, since the niching population size reduction mechanism is introduced by DCMAC, the diversity of the population is retained in early iterations and local search capability is improved in later iterations. Therefore, although the convergence rate of the DCMAC is not the fastest compared with that of other algorithms, the accuracy of the solution obtained by DCMAC is the highest among all the algorithms.

Statistical tests are shown that DCMAC algorithm is significantly improved compared with the comparative algorithm. In this paper, Wilcoxon's sign rank test (Zhao et al., 2018c) and Friedman test are selected for nonparametric statistical tests. The statistical analysis results of DCMAC and comparison algorithm using 29 benchmark functions in different dimensions are shown in Table 8. R^+ represents the sum of ranks for the functions of the DCMAC better than the second algorithm in the row and R^- represents the sum of ranks the functions of the second algorithm superior to the DCMAC. It is worth noting that in pair-wise comparison, DCMAC is the first algorithm in the row. According to the test results, $(+, \approx, \text{and} -)$ respectively represent DCMAC better than the second algorithm, there is no significant difference between the two algorithms, and DCMAC is worse than the second algorithm. The statistical results using DCMAC as the control algorithm are summarized in Table 8. When D = 30, 50, 100, DCMAC provides higher R^+ values than R^- as shown in Table 8. Especially when D = 50, 100, DCMAC is significantly better than other compared algorithms, although a significant difference is not observed between DCMAC and the comparison algorithm in the certain algorithm, the value R^+ is better than the value R^-. In summary, DCMAC outperforms other compared algorithms in $\alpha = 0.05$ and $\alpha = 0.1$. It is worth noting that the DCMAC algorithm is outstanding in solving high dimensional problems.

To further demonstrate the stability of the DCMAC algorithm, the box plots for f_1,f_9, f_{16} and f_{25} are shown in Figs. 10–12. As seen in Figs. 10–12, DCMAC is a slightly stable algorithm in the compared algorithms.

Friedman's test and Bonferroni-Dunn tests are used to further illustrate the significant differences between DCMAC and the five competitors are shown in Figs. 13 to 15. From Figs. 13 to 15, the proposed DCMAC algorithm ranks first in all dimensions. The additional Bonferroni–Dunn's method is applied as a post procedure to calculate the critical difference (CD) to evaluate the significance level of all algorithms for α = 0.05 and α = 0.1, as shown in Eq. (18).

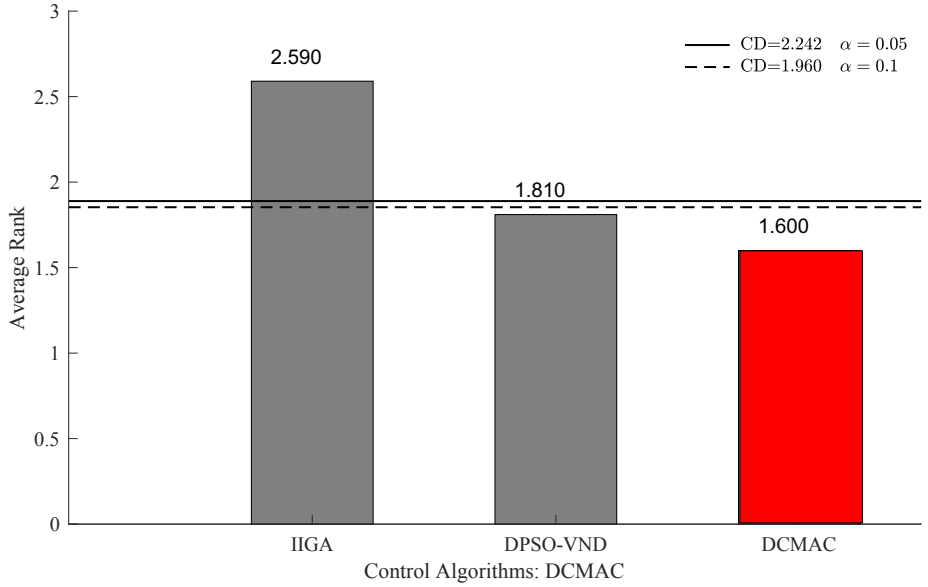


Fig. 17. Rankings for DCMAC.

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{18}$$

where parameters k and N are the number of algorithms and the number of instances, respectively. In the experimental evaluations, $k = 6$ and $N = 29$. When $\alpha = 0.1$, q_α is 2.327, and when $\alpha = 0.05$, q_α is 2.576, from Table B.16 [two-tailed α(2)] of Jerrold (Jerrold, 1999). Although there is no significant difference between the DCMAC and the compared algorithms in individual cases, the rank of DCMAC is the smallest among all the test cases.

It is noteworthy that the ensemble strategy focuses on improving the global search capability in the algorithm exploration stage, including the knowledge-based cooperative mutation strategy, parametric learning mechanism and niching population size reduction strategies. Individuals in the population become similar when the DCMAC algorithm falls into the local optimal solution. The mutation strategies and crossover operation do not generate new individuals and lead to population stagnation. For example, when the DCMAC algorithm falls into the local optimum, even if a weighted mutation strategy is applied in the DCMAC algorithm, none of the mutation strategies assist the DCMAC algorithm in escaping a local optimal. However, a knowledge-based cooperative mutation strategy is applied to avoid the DCMAC falling into the local optimum as far as possible. In different variants of DE, one of the key points for improving the performance of DE lies in the setting of parameters. The parameter adaptive strategies with a learning mechanism are introduced to the mutation strategy to ensure F self-adaptive and self-learning according to the experience of the previous generations in the DCMAC algorithm. In Table 8, DCMAC has desirable performance in high dimensional problems when utilizing the niching of population size reduction mechanism. At the beginning iteration of the algorithm, a sufficient computing resource is offered to avoid losing the diversity of the population. Then, the population is sorted by the Euclidean distance of the optimal solution and preferentially deleted the redundant solution and the suboptimal solution. However, the improvement effect is not responsible for the number of evaluations is limited in low dimensional problems. The conclusions are described below.

- The experimental comparison shows that the performance of DCMAC is better than other compared algorithms. According to Table 8, DCMAC proves to be significantly different from the majority of compared algorithms by using Wilcoxon's test, and it gets a better solution when solving high dimensional problems. Figs. 13 to 15 show the DCMAC is superior to other compared algorithms because its rank is the smallest among all comparison algorithms.
- The convergence performance of DCMAC is more stable than the state of art algorithm. Figs. 7 to 9 show, compared with the comparative algorithm, although the DCMAC algorithm does not converge fast, the local optimum is removed and a better result is achieved.
- The convergence accuracy and stability of DCMAC are superior to other compared algorithms are shown in Figs. 10–12.

6. Application in scheduling problems

The No-wait flow shop problem (NWFSP) is already widespread in the real world, such as the chemicals, plastics, metals, electronics, pharmaceuticals, and food-processing industries (Aldowaisan and Allahverdi, 2004; Sapkal and Laha, 2013). This problem is that once the processing is entered, it is not allowed to interrupt until all operations are completed. The problem is strongly NP-hard for three or more machines (Linet al., 2016). In this paper, the DCMAC algorithm is proposed to solve the problem and the minimized makespan is used as an evaluation criterion for optimized NWFSP (Lin and Ying, 2016).

6.1. Problem description

A set $N = \{J_1, J_2, ...J_n\}$ of n jobs is to be processed on m machines. Different jobs are processed in the same order on the machine and the constraints are as follows.

- A job can only be processed on one of the machines at the same time.
- Only one job of work can be processed by the same machine at any one time.
- There is no waiting time between two adjacent processes for the same job.
- The processing time of each job on different machines is known.
- The scheduling goal is to solve the optimal processing sequence of the jobs to minimize the scheduling index.

According to the constraint condition, when the first machine can not process the second job immediately after the first job is processed, the start time of the second job will be delayed. Fig. 16 describes the Gantt chart of no-wait flow shop scheduling with three jobs and three machines.

6.2. Mathematical model

$\pi = [\pi(1), \pi(2), \cdots, \pi(n)]$ is a scheduling sequence, C_{max} is the makespan of the scheduling sequence π. $p(\pi(i), k)$ is the processing time of the job i on the machine k. The minimum processing delay time $D(i,j)$ from job i to job j is calculated as follows.

$$D(i,j) = \max_{k=1,\cdots,m} \left\{ \sum_{h=k}^{m} (p(j,h) - p(i,h)) + p(i,k) \right\} \tag{19}$$

The makespan C_{max} calculation can be done very rapidly as follows.

$$C_{max}(\pi) = \sum_{i=2}^{n} D(\pi(j-1), \pi(j)) + \sum_{k=1}^{m} p(\pi(1), k) \tag{20}$$

In order to simplify the calculation process, $\pi(0)$ is a virtual job with a processing time of 0 can be introduced into the sequence. Therefore, the scheduling sequence is $\pi^{'} = [\pi(0), \pi(1), \pi(2), \cdots, \pi(n)]$. The calculation formula of the makespan C_{max} is modified as follows:

$$\begin{aligned} C_{max}(\pi) &= \sum_{i=2}^{n} D(\pi(j-1), \pi(j)) + \sum_{k=1}^{m} p(\pi(1), k) \\ &= \sum_{i=2}^{n} D(\pi(j-1), \pi(j)) + \sum_{i=2}^{n} D(\pi(0), \pi(1)) \\ &= \sum_{i=1}^{n} D(\pi(j-1), \pi(j)) \end{aligned} \tag{21}$$

6.3. Encoding and decoding of DCMAC

The DCMAC proposed in this paper uses a procedure-based coding method to solve NWFSP. First, the individuals in the sequence $X_i = [x_{i,1}, x_{i,2}, \cdots, x_{i,n}]$ are arranged in ascending order. A temporary sequence $\sigma_i = [\sigma_{i,1}, \sigma_{i,2}, \cdots, \sigma_{i,n}]$ is obtained. Then, Eq. (22) is used to obtain the corresponding sequence $\pi_i = [\pi_{i,1}, \pi_{i,2}, \cdots, \pi_{i,n}]$of discrete artifacts.

$$\pi_{i,\sigma_{i,j}} = jj \in [1:n] \tag{22}$$

In Table 9, the current individual vector $X_i = [0.3,\ 0.06,\ 0.9,\ -0.24,\ 0.85, -0.53]$. The smallest number after sorting is 0.35. The corresponding sort value of $x_{i,6}$ in σ_i is 1. By the same process, temporary sequences $\sigma_i = [4,3,6,2,5,1]$ are obtained. According to Eq. (22)., if $j = 1$,the$\sigma_{i,1} = 4, \pi_{i,\sigma_i} = \pi_{i,4} = 1$. The same method can be used to derive a procedure $\pi_i = [6,4,2,1,5,3]$. This simple and efficient mapping rule transforms individuals in a continuous domain into corresponding discrete individuals.

6.4. Simulation experiment and result analysis

6.4.1. Experimental setup

For the NWFSP problem, the DCMAC algorithm is compared with other algorithms. The minimum makespan is used to test the performance of different algorithms. The 120 instances provided by Taillard (1993) are used in the computational simulation. In the experiment, the average relative percentage deviation (ARPD) was used to measure the experimental results of the algorithm. ARPD calculation formula is as follows.

$$ARPD = \frac{1}{R} \bullet \sum_{i=1}^{R} \frac{C_i - C_{opt}}{C_{opt}} \bullet 100 \tag{23}$$

C_i is the solution obtained on the run time i of given an example in an algorithm. R is the total number of runs. C_{opt} is the optimal solution.

The DCMAC algorithm is compared with the other two current classical algorithms in the experiment. The classical algorithms are the improved iterated greedy algorithm (IIGA) (Pan et al., 2008a) and a discrete particle swarm optimization algorithm (DPSOVND) (Pan et al., 2008b). The same experimental conditions were used in the experiment, including the same termination criteria, the same computer, and the same programming language. In order to ensure the fairness of the experiment, all algorithms are programmed using MATLAB (2016b). The experiment was run on Intel(R) Core™I7-6700@ 3.40GhHz CPU, 8 GB of RAM, and Windows 64-bit operating system.

6.4.2. Simulation experiment and result analysis

Table 10 lists the ARPD values of the three algorithms, with the best results shown in bold. Table 10 shows that the ARPD value of the DCMAC algorithm is the smallest. Especially when solving large-scale problems, the performance of the DCMAC algorithm is much better than the comparison algorithm.

Table 11 shows the Wilcoxon symbolic rank test results of the comparison algorithm. According to the statistical analysis results in Table 11, the R^+ values of DCMAC are higher. The experimental results show that DCMAC is superior to the other two comparison algorithms for DPFSP when $\alpha = 0.05, \alpha = 0.01$.

The Friedman and Bonferroni-Dunn test is used to analyze the experimental data and show the effectiveness of the algorithm. As shown in Fig. 17, the rank of DCMAC is the smallest, although there is no significant difference between DCMAC and $DPSO_{VND}$. The experimental results show that DCMAC is superior to the two comparison algorithms.

7. Conclusions and future research

The paper presented a knowledge-based adaptation cooperative algorithm (DCMAC) to solve the numerical optimization problem. Extensive experimental analyses show that the proposed algorithm is superior to the existing algorithm regarding diversity and solution quality. In addition, the proposed algorithm can obtain feasible solutions with desirable quality to the problems with different scales. The superior performances of this algorithm are mainly attributed to the following aspects.

- In the proposed DCMAC, the cooperative evolution mechanism is realized in the hybrid framework of modified LSHADE and CMA-ES to further enhance the exploitation ability.
- The historical information population evolution is extracted as a kind of guiding knowledge to switch search engines. This process implies that the reward mechanism in reinforcement learning is introduced into the learning mechanism.
- In the proposed algorithm, the parameter values, which use an adaptive closed-loop control system, are dynamically adjusted. In terms of parameters, the robustness of the algorithm is enhanced.

According to the analysis experiment in this paper, the algorithm proposed not only has a high sensitivity to the parameters but also has a further requirement on the convergence rate. The next work will start from the parameters and convergence rate. The further work is as follows.

- The knowledge in this paper is the use of the data of the previous generation. The next work can focus on the use of the prediction method to analyze the existing knowledge as a whole and guide the next evolution to improve the effectiveness of the existing knowledge.
- In DCMAC, the evaluation function is expensive and a surrogate model is considered in the future to reduce evaluation function consumption and accelerate convergence.
- The algorithm in this paper focuses on the scheduling problem. Although the DCMAC has been applied to the classic problem of scheduling, many practical problems are now more complex. The next step is to study the application of this algorithm in more complex scheduling, such as distributed scheduling problem.

CRediT authorship contribution statement

Yang Zuo: Investigation, Software, Writing - original draft, Investigation, Software, Writing - original draft. **Fuqing Zhao:** Funding acquisition, Investigation, Supervision. **Zekai Li:** Methodology, Resources.

Declaration of Competing Interest

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

Acknowledgments

This work was financially supported by the National Natural Science Foundation of China under grant numbers 61663023. It was also supported by the Key Research Programs of Science and Technology Commission Foundation of Gansu Province (2017GS10817), Lanzhou Science Bureau project (2018-rc-98), Public Welfare Project of Zhejiang Natural Science Foundation (LGJ19E050001), Wenzhou Public Welfare Science and Technology project (G20170016), Science and Technology Project of Henan Province (212102210320), respectively.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.eswa.2021.115495.

References

Ali, W. M., Anas, A. H., & Kamal, J. (2019). Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global numerical optimization. *Swarm and Evolutionary Computation, 50*, Article 100455.

Awad, N., Ali, M., Liang, J., Qu, B., & Suganthan, P. (2016). *Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective bound constrained real-parameter numerical optimization*. Nanyang Technological University Singapore.

Ali, W. M., Anas, A. H., Anas, F., & Kamal, J. (2017). LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems. *Evolutionary Computation*, 145–152.

Ali, W. M., Anas, A. H., Kamal, J. (2018). Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global numerical optimization. Swarm and Evolutionary Computation, 50, UNSP 100455.

Cui, L., Li, G., Zhu, Z., Lin, Q., Wong, K.-C., Chen, J., et al. (2018). Adaptive multiple-elites-guided composite differential evolution algorithm with a shift mechanism. *Information Sciences, 422*, 122–143.

Cui, L., Li, G., Zhu, Z., Ming, Z., Wen, Z., & Lu, N. (2019). Differential evolution algorithm with dichotomy-based parameter space compression. *Soft Computing, 23* (11), 3643–3660.

David, E. G., & John, H. H. (1988). Genetic algorithms and machine learning. *Machine Learning, 3*, 95–99.
Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization, 39*(3), 459–471.
Hu, ZhongBo, Xiong, ShengWu, Su, QingHua, & Fang, ZhiXiang (2014). Finite Markov chain analysis of classical differential evolution algorithm. *Journal of Computational and Applied Mathematics, 268*, 121–134.
Ji, J., Song, S., Tang, C., Gao, S., Tang, Z., & Todo, Y. (2019). An artificial bee colony algorithm search guided by scale-free networks. *Information Sciences, 473*, 142–165.
James, K., & Russell, C. E. (1995). Particle swarm optimization. *In Proceedings of the IEEE international conference on neural networks 1942–1948*.
Janez, B., Mirjam, S. M., & Borko, B. (2017). Single objective real-parameter optimization: algorithm jSO. *Evolutionary Computation*, 1311–1318.
Janez, B., Mirjam, S. M., & Borko, B. (2016). iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization. In *In IEEE Congress on Evolutionary Computation (CEC) held as part of IEEE World Congress on Computational Intelligence (IEEE WCCI)* (pp. 1188–1195).
Montgomery, D. C. (2017). *Design and analysis of experiments*. John Wiley and Sons.
Noor, H. A., Mostafa, Z. A., & Suganthan, P. N. (2017). Ensemble Sinusoidal Differential Covariance Matrix Adaptation with Euclidean Neighborhood for Solving CEC2017 Benchmark Problems. In *In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC)* (pp. 372–379).
Nikolaus, H. (2006). The CMA evolution strategy: a comparing review.In Towards a new Evolutionary Computation, 75-102.
Awad, Noor H., Ali, Mostafa Z., & Suganthan, Ponnuthurai N. (2018). Ensemble of parameters in a sinusoidal differential evolution with niching-based population reduction. *Swarm and Evolutionary Computation, 39*, 141–156.
N.H. Awad, M. Z. Ali, P. N. Suganthan. (2017). Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. Technical Report.
Patrick, S., Hansgeorg, B., & Michael, H. (2018). A Covariance matrix self-adaptation evolution strategy for optimization under linear constraints. *IEEE Transactions on Evolutionary Computation, 23*, 514–524.
Ruiz, Rubén, Pan, Quan-Ke, & Naderi, Bahman (2019). Iterated Greedy methods for the distributed permutation flowshop scheduling problem. *Omega, 83*, 213–222.
Rainer, S., & Kenneth, Price (1995). *Differential evolution-A simple and efficient adaptive scheme for global optimization over continuous spaces*. Berkeley: ICSI.
Ryoji, T., Alex F. (2013). Evaluating the performance of SHADE on CEC 2013 benchmark problems. In 2013 IEEE Congress on Evolutionary Computation (CEC),1952-1959.
Ryoji, T., & Alex, F. (2014). Improving the search performance of SHADE using linear population size reduction. *Evolutionary Computation*, 1658–1665.
Ibrahim, Rehab Ali, Elaziz, Mohamed Abd, & Lu, Songfeng (2018). Chaotic opposition-based grey-wolf optimization algorithm based on differential evolution and disruption operator for global optimization. *Expert Systems with Applications, 108*, 1–27.
Das, Swagatam, Mullick, Sankha Subhra, & Suganthan, P. N. (2016). Recent advances in differential evolution – An updated survey. *Swarm and Evolutionary Computation, 27*, 1–30.
Shao, Zhongshi, Pi, Dechang, Shao, Weishi, & Yuan, Peisen (2019). An efficient discrete invasive weed optimization for blocking flow-shop scheduling problem. *Engineering Applications of Artificial Intelligence, 78*, 124–141.
Vladimir, S., Shakhnaz, A., & Eugene, S. (2018). LSHADE Algorithm with Rank-Based Selective Pressure Strategy for Solving CEC 2017 Benchmark Problems. In *In 2018 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1–8).
Viktorin, A., Senkerik, R., Pluhacek, M., Kadavy, T., & Zamuda, A. (2019). Distance based parameteradaptation for success-history based differential evolution. *Swarm and Evolutionary Computation, 50*, 100462. https://doi.org/10.1016/j.swevo.2018.10.013
Wang, J.-J., & Wang, L. (2018). A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1–15.
Xueqi, Wu., & Che, Ad.a. (2019). A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. *Omega, 82*, 155–165.
Wang, Hao, Emmerich, Michael, & Bäck, Thomas (2019). Mirrored orthogonal sampling for covariance matrix adaptation evolution strategies. *Evolutionary Computation, 27*(4), 699–725.
Zhao, Fuqing, Liu, Huan, Zhang, Yi, Ma, Weimin, & Zhang, Chuck (2018a). A discrete water wave optimization algorithm for no-wait flow shop scheduling problem. *Expert Systems with Applications, 91*, 347–363.
Mohammad, R, B., Zbigniew, M. (2017). Particle swarm optimization for single objective continuous space problems: a review. In Evolutionary Computation,1-54.
Zhang, J. Q., & Sanderson, A. C. (2009). JADE: adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation, 13*(5), 945–958.
Zhao, Fuqing, Qin, Shuo, Zhang, Yi, Ma, Weimin, Zhang, Chuck, & Song, Houbin (2019a). A two-stage differential biogeography-based optimization algorithm and its performance analysis. *Expert Systems with Applications, 115*, 329–345.
Zhao, Fuqing, Xue, Feilong, Zhang, Yi, Ma, Weimin, Zhang, Chuck, & Song, Houbin (2018b). A hybrid algorithm based on self-adaptive gravitational search algorithm and differential evolution. *Expert Systems with Applications, 113*, 515–530.
Zhao, Fuqing, Zhang, Lixin, Liu, Huan, Zhang, Yi, Ma, Weimin, Zhang, Chuck, et al. (2019b). An improved water wave optimization algorithm with the single wave mechanism for the no-wait flow-shop scheduling problem. *Engineering Optimization, 51*(10), 1727–1742.
Aldowaisan, Tariq, & Allahverdi, Ali (2004). New heuristics for m-machine nowait flowshop to minimize total completion time. *Omega, 32*(5), 345–352.
Sapkal, Sagar U., & Laha, Dipak (2013). A heuristic for no-wait flow shop scheduling. *International Journal of Advanced Manufacturing Technology, 68*(5-8), 1327–1338.
Lin, Shih-Wei, & Ying, Kuo-Ching (2016). Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. *Omega, 64*, 115–125.
Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research, 64*(2), 278–285.
Pan, Quan-Ke, Wang, Ling, & Zhao, Bao-Hua (2008). An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion. *The International Journal of Advanced Manufacturing Technology, 38*(7-8), 778–786.