Swarm and Evolutionary Computation 56 (2020) 100699

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Self-organizing neighborhood-based differential evolution for global optimization

Yiqiao Cai [a,b,*], Duanwei Wu [a], Ying Zhou [c], Shunkai Fu [a,b], Hui Tian [a,b], Yongqian Du [a,b]

[a] College of Computer Science and Technology, Huaqiao University, Xiamen, 361021, China
[b] Fujian Key Laboratory of Big Data Intelligence and Security, Huaqiao University, China
[c] School of Computer Sciences, Shenzhen Institute of Information Technology, Shenzhen, 518172, China

ARTICLE INFO

Keywords:
Differential evolution
Neighborhood utilization technique
Self-organizing map
Neighborhood learning
Neighborhood adaption
Mutation operator

ABSTRACT

Combining neighborhood utilization technique (NUT) has shown a tremendous benefit to differential evolution (DE) due to that the acquired neighborhood information of population is of great help in guiding the search. However, in most NUT-based DE algorithms, on the one hand, the neighborhood relationships between individuals cannot be effectively and properly learned, and on the other hand, the search roles of different individuals have not yet been fully considered in the design of the NUT. Therefore, this study develops a novel NUT, termed self-organizing neighborhood (SON), with three features: 1) the neighborhood relationships between individuals are incrementally learned and extracted by self-organizing map with the cosine similarity; 2) the neighborhood sizes for different individuals are adaptively adjusted according to their distinct roles in the search; 3) the evolution direction constructed with the neighborhood of each individual is incorporated into the mutation process to guide the search. By combining SON with DE, a SON-based DE (SON-DE) framework is proposed for global optimization. Experimental results on 58 real-parameter functions and 17 real-world problems have demonstrated the superiority of SON-DE in comparison with several state-of-the-art DE algorithms and evolutionary algorithms (EAs).

1. Introduction

As a simple yet efficient evolutionary algorithm (EA), differential evolution (DE) has shown its excellent performance in solving various optimization problems, such as multi-objective, multi-modal, multi-task, large-scale, dynamic, and uncertain optimization problems [1–3]. Besides, with the rapid development and popularization, DE has been widely and successfully applied to a mass of real-world optimization problems in the scientific and engineering fields [3,4].

Like other EAs, DE is a population-based stochastic optimization algorithm, which manipulates a group of individuals by three evolutionary operators, i.e., mutation, crossover, and selection [1]. To elaborate, DE first randomly initializes a population of individuals within the specified search space. Then, the mutation operator is used to generate a mutant vector for each individual (called the target vector). After that, the crossover operator is applied to each pair of the target vector and its mutant vector for obtaining a trial vector. Finally, the selection operator is employed to select the survival for the next generation between the target vector and its trial vector. Although DE has achieved the promising results for different types of optimization problems, its performance is greatly influenced by the following factors: the offspring generation strategy (mutation and crossover strategies) and the control parameters (the population size NP, the scale factor for mutation F, and the crossover rate C_r). Therefore, many attempts have been made to further improve the performance of DE in the aspect of convergence speed and robustness. In general, these improvements mainly focus on the following directions: parameters control strategy [5–8], offspring generation strategy [9–12], multiple-operator strategy [13–16], decentralized population structure [17–20], and hybrid strategy [21–23].

Among these works, the researches on the mutation operator have attracted much attention, leading to many advanced DE variants being presented. Recently, an increasing interest has been devoted to developing the neighborhood utilization technique (NUT) to extract useful information of the population for guiding the mutation process. Roughly, the remarkable works on this subject can be classified into the following two categories:

* Corresponding author. College of Computer Science and Technology, Huaqiao University, Xiamen, 361021, China.
E-mail address: caiyq@hqu.edu.cn (Y. Cai).

https://doi.org/10.1016/j.swevo.2020.100699
Received 15 May 2019; Received in revised form 3 March 2020; Accepted 23 April 2020
Available online 30 April 2020

1) *Population topology-based NUT (PT-NUT)*: In this category, the neighborhood of each individual is created based on the order of its index in the population. During the evolutionary process, each individual only interacts with its fixed neighbors defined by the used topology. In this way, the flow of population information can be controlled by the employed structured population with the different degrees of connectivity, the amount of clustering, and the average shortest distance between individuals [24]. In the literature of NUT-based DE, PT-NUT can be divided into two groups based on the used topology, i.e., coarse-grained (distributed) topology-based NUT and fine-grained (cellular, ring, and small-world) topology-based NUT.
   - For the coarse-grained topology-based NUT, distributed topology is used to partition the population into several large and loosely connected sub-populations. Falco et al. proposed an enhanced distributed DE (dDE) with an asynchronous adaptive multi-population model to select the sub-population for migration [25]. Ge et al. proposed a dDE with an adaptive mergence-and-split operator to dynamically allocate the computational resources among different sub-populations for large-scale optimization problems [20]. Besides, many works related to dDE have been proposed for solving different optimization problems, such as the heterogeneous dDE (HdDE) [19], dDE with explorative-exploitative population families (DDE-EEPF) [26], dDE with scale factor inheritance mechanism (FACPDE) [27], and so on.
   - For the fine-grained topology-based NUT, cellular (ring or small-world) topology is employed to divide the population into small and tightly connected sub-populations. Dorronsoro and Bouvry introduced several decentralized population topologies (i.e., cellular, ring, and small-world) into DE to define the neighborhood for each individual [19]. Cai et al. proposed a neighborhood-guided DE (NGDE) by combining the neighborhood information from the ring topology and the fitness information of the population [17]. Das et al. proposed a DE variant (DEGL) with the neighborhood-based mutation operators by using the ring topology to define the global and local neighborhoods [9]. Additionally, fine-coarse topology-based NUT has been incorporated into many DE variants, such as the bare-bones DE (BBDE) [28], cellular-based DE (CellularDE) [29], DE with cellular direction information (DE-CDI) [30], adaptive multiple-elites-guided composite DE (AMECoDEs) [31], and so on.
   - Apart from DE with single topology, multiple fine-grained and coarse-grained topologies are used simultaneously in DE to take the advantages of both. Sun et al. proposed a DE variant with five different topologies and selected a topology for different generations based on the improvements obtained by the best individual [32]. Cai et al. proposed a neighborhood-adaptive DE (NaDE) by defining multiple neighborhoods for each individual with the cellular and ring topologies and adaptively selecting the neighborhood based on the historical successful and failed experiences [33]. Like that, Sun et al. proposed a multi-topology-based DE (MTDE) with an individual-dependent topology adaptation where the differences between individuals in the search roles are used to select the topology for different individuals [18].
2) *Individual characteristics-based NUT (IC-NUT)*: In this category, the neighborhood of each individual is defined based on the characteristics of individuals in the objective and decision space. At different stages of evolution, the individuals with similar characteristics in the objective (i.e., the fitness value) or decision space (i.e., the position information) will have similar behaviors and roles in guiding the search of the population. Therefore, many works focus on introducing IC-NUT into DE to construct the neighborhood for each individual. Here, IC-NUT will be broadly classified into the following groups: position proximity-based NUT and fitness value-based NUT.
   - For the position proximity-based NUT, the neighborhood of each individual is built based on the Euclidean distance between each pair of individuals, which takes advantage of the possible clustering structures of the problem landscape. In this group, NUT guides the mutation process in two ways. The one is to select parents from the neighborhood of each individual in a probabilistic way. For example, Epitropakis et al. proposed a proximity-based DE framework (ProDE) where the selection probabilities of individuals are inversely proportional to their distances from the target individual [34]. Qu et al. proposed a neighborhood mutation for multimodal optimization where a set of individuals with the smallest Euclidean distance to the target individual are selected to form a subpopulation [35]. He et al. proposed a fuzzy neighborhood-based DE with orientation (FNODE) for nonlinear equation systems where the fuzzy rule is used to construct the subpopulations based on the distribution of individuals [36]. The other is to divide the whole population into several subpopulations by the clustering method. For example, Cai et al. proposed a learning-enhanced DE (LeDE) by adopting a clustering-based learning strategy to strengthen the exchange of information between clusters [37]. Liu and Guo proposed a clustering-based DE with random-based sampling and Gaussian sampling (GRCDE) where the whole population is partitioned into several subpopulations by one-step k-means clustering [38]. For the multimodal optimization, the clustering method is frequently incorporated into DE to maintain the population diversity and locate multiple optima, such as cluster-based DE with self-adaptive strategy (self-CCDE) [39], automatic niching DE with affinity propagation clustering (ANDE-APC) [40], and so on.
   - For the fitness value-based NUT, the neighborhood of each individual is defined based on the fitness values of the population, which makes use of the valuable information for the search from the better species [41]. Differing from the proximity-based neighborhood, the fitness-based neighborhood is defined implicitly in the objective space. For example, Gong and Cai proposed a DE variant with ranking-based mutation operators (RankDE) where all the individuals are sorted according to their fitness values and are proportionally selected on their rankings [41]. Cai et al. proposed a social learning DE (SL-DE) where the social influence of each individual is evaluated based on its ranking in the population and the neighborhood of each individual is constructed with its social influence [42].
   - In addition to the above, several researches have combined the fitness and position information of the population to propose the hybrid-based NUT. For example, Cai et al. proposed a DE with neighborhood and direction information (NDi-DE) where a probability selection is used based on the distance information and the best and worst near-neighbors are adopted to provide the direction information [43]. Gao et al. proposed a species-based DE with the self-adaptive strategy (self-CSDE) where the fitness information is used to determine the species seeds and the position information is used to select the individuals within the same species [39]. Like that, Deng et al. proposed a DE with dynamic speciation-based mutation (DSM-DE) where both fitness and position information are used to identify the species [44]. Wang et al. proposed an enhanced DE with multi-objective sorting-based mutation operators (MSDE) where all the individuals are ranked based on their fitness values and diversity contribution by using the nondominated sorting operator and the parents are proportionally selected based on their rankings [45]. Cai et al. proposed a DE variant with the fitness-and-position based selection (FPS-DE) where both fitness and position information are used to calculate the influence of each individual and the selection of parents is executed according to their influence [46].

Although the previous studies on NUT have demonstrated their effectiveness in improving the performance of DE, it can be found that two important issues are still not addressed well in most NUT-based DE

variants. First, although the scatting of population information can be controlled by PT-NUT, thus leading to the promotion of exploration, the neighborhood relationships between individuals cannot be adaptively and properly learned by the fixed topology without considering the distribution of individuals in the search space. As shown in Refs. [47], the dynamical neighborhood information has contributed greatly to exploiting the local region of the fitness landscape and reducing the genetic drift. Therefore, the neighborhood information at different stages of evolution needs to be effectively learned, extracted and used in the design of NUT for DE. Second, much effort has been invested in presenting the NUT-based DE for different optimization scenarios, but the distinct search roles of different individuals have not yet been taken into account in most NUTs. As shown in Ref. [18,45,48], individuals with different fitness values reveal distinct search roles in guiding the search of the population. Generally, superior individuals tend to drive the population towards the promising regions for exploitation, while the inferior individuals are likely to keep the diversity of populations for exploration. Thus, it is expected that integrating the search roles of different individuals into the design of NUT will further benefit the performance of NUT-based DE.

Motivated by the above discussions, this study develops a novel NUT, self-organizing neighborhood (SON), to address the two issues. SON consists of three key components, i.e., *neighborhood learning strategy*, *neighborhood adaption mechanism*, and *neighborhood utilization strategy*. By applying SON to DE, the resultant DE framework, termed SON-based DE (SON-DE), is proposed for global optimization.

Overall, the features of SON-DE are summarized as follows:

- The neighborhood relationships between individuals are incrementally learned by self-organizing map (SOM) [49]. With the improved individuals, the neighborhood information of the current population can be learned by reusing the distribution information of the historical population.
- The cosine distance is adopted in SOM to measure the similarity between individuals. In this way, the neighborhood relationships between individuals are built by considering their search directions, and thus each individual will be guided by the neighbors with a similar search direction.
- A novel neighborhood adaption mechanism is presented to adaptively adjust the neighborhood size of each individual to match its search role at different stages of evolution.
- Evolution direction, constructed with the neighbors of each individual, is incorporated into the mutation operator to further guide its search by providing a consistent and promising search direction.

To evaluate the performance of SON-DE, the comprehensive experiments are carried out on a set of benchmark functions which consists of 28 test functions from the 2013 IEEE congress on evolutionary computation (CEC2013) special session on real-parameter optimization [50], 30 test functions from the CEC2017 special session on real-parameter optimization [51], and 17 real-world numerical optimization problems from the CEC2011 competition [52]. Compared with other state-of-the-art DE variants and EAs, the highly competitive performance of SON-DE has been demonstrated.

The rest of this paper is organized as follows. Section 2 briefly introduces DE and SOM. The proposed SON-DE framework is presented in detail in Section 3. In Section 4, the experimental results and the performance comparisons are shown and discussed. Finally, the conclusions and future work are given in Section 5.

2. Background

2.1. Differential evolution (DE)

In DE, a population of *NP* individuals is used for searching the optimal solution of an optimization problem. The population at the *g*th generation is denoted as $P^g = \{\vec{x}_i^g = (x_{i,1}^g, x_{i,2}^g, \dots, x_{i,D}^g) | i = 1, \dots, NP\}$, where D is the dimension of the optimization problem. At the beginning of DE (i.e., $g = 0$), each individual of the population is randomly initialized within the pre-specified search space, as follows:

$$x_{i,j}^0 = x_{min,j} + rndreal(0,1) \times (x_{max,j} - x_{min,j}), j = 1, 2, \dots, D \tag{1}$$

where $rndreal(0,1)$ represents a uniformly distributed random number in the interval [0,1], and $x_{\max,j}$ ($x_{\min,j}$) represents the upper (lower) bounds of the jth variable.

After the initialization, three main operators, i.e., mutation, crossover, and selection, are carried out iteratively. The general framework of DE is shown in Algorithm 1, where *MS*, *CS*, *SS* mean the used strategy in the corresponding operator, and $G_{\max}$ is the maximal number of generations.

Algorithm 1 The general framework of DE

1: *Initialization*: Randomly generate the initial population P^0, and set $g = 0$;
2: *Evaluation*: Calculate the fitness value of each individual in P^0;
3: **WHILE** $g < G_{\max}$
4: **FOR** $i = 1$ to *NP* **DO**
5: *Mutation*: $\vec{v}_i^g = MS(\vec{x}_i^g, \overrightarrow{bv}_i^g, \overrightarrow{dv}_i^g)$;
6: *Crossover*: $\vec{u}_i^g = CS(\vec{x}_i^g, \vec{v}_i^g)$;
7: *Selection*: $\vec{x}_i^{g+1} = SS(\vec{x}_i^g, \vec{u}_i^g)$;
8: **END FOR**
9: $g = g + 1$;
10: **END WHILE**
11: *Output*: Return the individual with the best fitness value.

Mutation: Each individual $\vec{x}_i^g$ (called target vector) uses a specified mutation strategy (*MS*) to generate a mutant vector $\vec{v}_i^g$ by combining a base vector ($\overrightarrow{bv}_i^g$) with a difference vector ($\overrightarrow{dv}_i^g$). The general formula of *MS* for $\vec{x}_g^i$ can be described below [53]:

$$\vec{v}_i^g = \overrightarrow{bv}_i^g + F \times \overrightarrow{dv}_i^g \tag{2}$$

where F is a positive control parameter for scaling $\overrightarrow{dv}_i^g$. In Eq. (2), $\overrightarrow{bv}_i^g$ can be the randomly selected vector, the best vector, or the combined vector, and $\overrightarrow{dv}_i^g$ can be constructed in a random or directed manner [53]. In the literature, several commonly and frequently used *MS*s are shown below:

- DE/rand/1

$$\vec{v}_i^g = \vec{x}_{r1}^g + F \times (\vec{x}_{r2}^g - \vec{x}_{r3}^g) \tag{3}$$

- DE/rand/2

$$\vec{v}_i^g = \vec{x}_{r1}^g + F \times (\vec{x}_{r2}^g - \vec{x}_{r3}^g) + F \times (\vec{x}_{r4}^g - \vec{x}_{r5}^g) \tag{4}$$

- DE/best/1

$$\vec{v}_i^g = \vec{x}_{best}^g + F \times (\vec{x}_{r1}^g - \vec{x}_{r2}^g) \tag{5}$$

- DE/best/2

$$\vec{v}_i^g = \vec{x}_{best}^g + F \times (\vec{x}_{r1}^g - \vec{x}_{r2}^g) + F \times (\vec{x}_{r3}^g - \vec{x}_{r4}^g) \tag{6}$$

- DE/current-to-best/1

$$\vec{v}_i^g = \vec{x}_i^g + F \times (\vec{x}_{best}^g - \vec{x}_i^g) + F \times (\vec{x}_{r1}^g - \vec{x}_{r2}^g) \tag{7}$$

- DE/rand-to-best/1

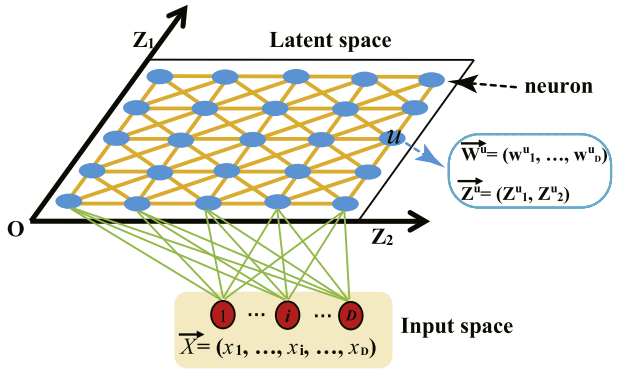

Fig. 1. An illustration of a 2-dimensional SOM.

$$\vec{v}_i^g = \vec{x}_{r1,g} + F \times (\vec{x}_{best}^g - \vec{x}_{r1}^g) + F \times (\vec{x}_{r2}^g - \vec{x}_{r3}^g) \tag{8}$$

where $r1$, $r2$, $r3$, $r4$, and $r5 \in \{1, 2, \ldots, NP\} \backslash \{i\}$ are mutually different and randomly selected indices and $\vec{x}_{best}^g$ is the best individual at generation g. During the mutation process, if the variable is out of the pre-specified interval, it will be reinitialized with Eq. (1). More details can refer to Ref. [1,4].

Crossover: To enhance the diversity of the population, the crossover operator is performed with a crossover strategy (*CS*) to generate a trial vector $\vec{u}_i^g$ for each pair of $\vec{x}_i^g$ and $\vec{v}_i^g$. The *CS* can be binomial, exponential, or arithmetic. The binomial crossover, as the frequently used in the literature, is expressed as follows:

$$u_{i,j}^g = \begin{cases} v_{i,j}^g, & \text{if } rndreal(0,1) \le Cr \text{ or } j = j_{rand} \\ x_{i,j}^g, & \text{otherwise.} \end{cases} \tag{9}$$

where $Cr \in [0, 1]$ is the crossover rate and $j_{rand} \in [1, D]$ is a randomly selected integer.

Selection: The selection strategy (*SS*) is carried out to select the better one between $\vec{x}_i^g$ and $\vec{u}_i^g$ for the next generation. In the original DE algorithm, the one-to-one greedy selection is used as *SS*, which is performed as follows:

$$\vec{x}_i^{g+1} = \begin{cases} \vec{u}_i^g, & \text{if } f(\vec{u}_i^g) \le f(\vec{x}_i^g) \\ \vec{x}_i^g, & \text{otherwise.} \end{cases} \tag{10}$$

where $f(\vec{a})$ is the fitness value of $\vec{a}$ for the optimization problem to be minimized.

2.2. Self-organizing map (SOM)

SOM, as an unsupervised learning method, is to learn a feature map from the high dimensional input space to the low dimensional map space [49,54]. In SOM, a neighborhood function is used to preserve the neighborhood relationships within the input data set, which is the major merit of SOM [49]. Here, a brief description of SOM will be given, and more details can be found in Refs. [49,54].

Suppose that $TD = \{\vec{td}_1, \ldots, \vec{td}_{|TD|}\}$ is the set of training data in the D-dimensional input space, and a neural network consists of N neurons in the m-dimensional latent space. In the latent space, each neuron u has a position vector $\vec{z}^u = (z_1^u, z_2^u, \ldots, z_m^u)$ and a weight vector $\vec{w}^u = (w_1^u, w_2^u, \ldots, w_D^u)$, $u = 1, \ldots, N$. Note that the number of dimensions of $\vec{w}^u$ is equal to that of the problem to be optimized.

Fig. 1 depicts an illustration of a two-dimensional SOM that is made up of 25 neurons arranged in a rectangular grid. Besides, the general framework of SOM is shown in Algorithm 2, where $G_{\max}$ is the maximal number of iterations.

As shown in Algorithm 2, the weight vector of each neuron, the neighborhood radius σ_0, and the learning rate τ_0 will be first initialized. Then, SOM enters the training process which includes three stages: competition, cooperation, and adjustment.

Competition stage: A vector $\vec{td}_i$ is randomly selected from *TD*, and then the distance between $\vec{td}_i$ and the weight vector of each neuron is calculated. After that, the neuron whose weight vector is nearest to $\vec{td}_i$ is determined as the winning neuron (*wu*).

Cooperation stage: The set of the neighboring neurons (N_{wu}) for *wu* is created based on the distances between it and other neurons in the latent space. Then, the weight vectors of the neurons in N_{wu} are rewarded and updated based on the learning rule.

Adjustment stage: The neighborhood radius (σ) that decides the number of neighboring neurons and the learning rate (τ) that decides how much the weight vector of neighbor can learn are adjusted with the iterations.

Algorithm 2 The general framework of SOM

1: *Initialization*: Randomly initialize the weight vector for each neuron, the neighborhood radius σ_0, the learning rate τ_0, and $g = 0$;
2: **WHILE** $g < G_{\max}$
3: *Competition*: Randomly select $\vec{td}_i \in TD$ and determine the winning neuron *wu* that is nearest to $\vec{x}$;
4: *Cooperation*: Determine the set of neighboring neurons N_{wu} for *wu* and update their weight vectors;
5: *Adjustment*: Adjust σ and τ values along with g;
6: $g = g + 1$;
7: **END WHILE**
8: *Output*: Return the weight vectors of all the neurons.

3. Proposed SON-DE algorithm

In this section, the self-organizing neighborhood (SON) is first presented with three key components: neighborhood learning strategy, neighborhood adaption mechanism, and neighborhood utilization strategy. Followed that, the proposed SON-DE framework by incorporating SON into DE is shown. Finally, some discussions about the differences between SON-DE and other NUT-based DE variants are given.

3.1. Self-organizing neighborhood (SON)

As reviewed above, various NUTs have been proposed for DE, which leads to significant improvements in the performance of DE when solving complex optimization problems. However, there are still some weaknesses in most NUTs, which inspires the emergence of SON. To elaborate, the motivations of SON mainly come from the following aspects.

1) A larger population with improved solutions can provide more valuable information for learning the neighborhood relationships between individuals. In most existing NUTs, the neighborhood relationships are learned only from the current population or defined by the fixed topology, resulting in the loss of the evolutionary information from the historical population. Hence, it is necessary to design an effective NUT to incrementally learn the neighborhood information from both current and historical populations.
2) The importance of distinguishing the search roles of individuals with different fitness values have been recognized by many researchers. However, in most NUTs, different individuals are usually equipped with the neighborhoods of the same size, which might not effectively play their distinct roles in guiding the search. Consequently, considering the different search role of each individual in the design of NUT will be a promising way to further enhance the performance of DE.
3) How to effectively use the neighborhood information of population has a great impact on the performance of NUT-based DE variants.

In most NUT-DE variants, the parents for mutation are randomly selected from the neighborhood, leading to the under-utilization of neighborhood information of population. Therefore, to enhance the capability in using the neighborhood information is an important topic for the NUT-based DE.

The purpose of the SON is to address the above issues. In SON, a neighborhood learning strategy based on self-organizing map (SOM) is first introduced to learn the neighborhood relationships of the population. Then, a neighborhood adaptation mechanism is designed to adaptively assign the neighborhoods of distinct sizes to different individuals. Followed that, a neighborhood utilization strategy is presented to construct the evolution direction from the neighborhood to guide the mutation process. Besides, the procedure of the SON is illustrated in Fig. 2.

3.1.1. Neighborhood learning strategy

Due to its advantage in preserving the neighborhood relationships of the input data, the classical 2-dimensional SOM is employed. In the neighborhood learning strategy, first, the weight vectors of neurons are iteratively updated with the training data *TD*. When the update is completed, each neuron will be linked with a unique individual of the population. In this way, the neighborhood relationships between individuals can be established based on the topological relationships between the corresponding neurons. Specifically, the neighborhood learning strategy is composed of two operators: the SOM model updating (Fig. 2 (A)) and the neighborhood relationship building (Fig. 2 (B)). Besides, the procedure of the neighborhood learning strategy with these two operators is shown in Algorithm 3.

1) *SOM model updating operator*: For each training point ($\vec{td}_i \in DT$), the neuron *wu* whose weight vector is nearest to it is first determined as follows:

$$wu = \arg \max_{1 \le u \le NP} \text{Similar}(\vec{td}_i, \vec{w}^u) \tag{11}$$

where $\text{Similar}(\vec{td}_i, \vec{w}^u)$ is used to measure the similarity between $\vec{td}_i$ and $\vec{w}^u$. As stated in Ref. [55], the Euclidean and cosine distances are the most commonly used measures to calculate the similarity between two vectors. In this study, the cosine distance is used, which is shown below:

$$\text{Similar}(\vec{td}_i, \vec{w}^u) = \frac{\vec{td}_i \cdot \vec{w}^u}{\|\vec{td}_i\|_2 \times \|\vec{w}^u\|_2} \tag{12}$$

where $\vec{td}_i \cdot \vec{w}^u$ is the inner product between $\vec{td}_i$ and $\vec{w}^u$.
Then, the set of neighboring neurons for *wu* (N_{wu}) is constructed below:

$$N_{wu} = \{u | Dis(u, wu) < \sigma \wedge 1 \le u \le NP\} \tag{13}$$

where $Dis(u, wu)$ is the topological distance between *u* and *wu* in the latent space and can be calculated as

$$Dis(u, wu) = \|\vec{z}^u, \vec{z}^{wu}\|_2. \tag{14}$$

Next, the weight vector of $u \in N_{wu}$ is updated with the following learning rule [54]:

$$\vec{w}^u = \vec{w}^u + \tau \cdot exp(-Dis(u, wu))(\vec{td}_i - \vec{w}^u) \tag{15}$$

At last, the values of σ and τ are adjusted as follows [56]:

$$\sigma = \sigma_0 \times (1 - \frac{g}{G_{max}} - \frac{i}{NP \times G_{max}}) \tag{16}$$

$$\tau = \tau_0 \times (1 - \frac{g}{G_{max}} - \frac{i}{NP \times G_{max}}) \tag{17}$$

where $i = 1, 2, \ldots |TD|$.

2) *Neighborhood relationship building operator*: For each individual ($\vec{x}_j^g \in P^g$), the neuron *u* that is linked to it will be determined by:

$$u = \arg \max_{1 \le c \le NP \wedge lb_c = null} \text{Similar}(\vec{x}_j^g, \vec{w}^c) \tag{18}$$

where $\vec{lb} = (lb_1, \ldots, lb_{NP})$ is used to indicate whether the neuron has been linked to an individual. $lb_u = j$ means the neuron *u* has been linked to $\vec{x}_j^g$; otherwise, $lb_u = null$ means it has not been linked to any individuals.

When all the neurons have been linked to all the individuals of the population, the neighborhood relationships between individuals can be established according to the position relationships between their linked neurons in the latent space.

As shown in Fig. 2(A) and (B), after executing the SOM model updating operator, each individual of the population is linked to a unique neuron by the neighborhood relationship building operator. Based on these established links, the neighbors of each individual can be determined by the neighboring neurons of its linked neuron.

Algorithm 3 Neighborhood learning strategy

```
/****SOM model updating****/
FOR i = 1, 2, ..., |TD| DO
    Determine the winning neuron wu whose weight vector is closest to td_i ∈ TD using Eq. (11);
    Construct the set of neighboring neurons N_wu for wu using Eq. (13);
    Update the weight vector of each neuron in N_wu using Eq. (15);
    Adjust σ and τ using Eqs. (16) and (17), respectively;
END FOR
/****Neighborhood relationship building****/
Set lb_u = null for each neuron u, u = 1, 2, ..., NP;
FOR j = 1, 2, ..., NP DO
    Determine the neuron u for linking x_j^g using Eq. (18);
    Set lb_u = j;
END FOR
```

3.1.2. Neighborhood adaption mechanism

As discussed above, individuals with different fitness values have a distinct influence on the search of population. Thus, the search roles of different individuals should be distinguished, which is beneficial for guiding the process of evolution. For this purpose, a neighborhood adaption mechanism is designed, which consists of two operators: neighborhood size adapting (Fig. 2 (C)) and neighbors locating (Fig. 2 (D)). The procedure of the neighborhood adaption mechanism is also given in Algorithm 4.

1) *Neighborhood size adapting operator*: To evaluate the search role of each individual, all the individuals of the current population are first sorted in descending order based on their fitness values. Then, the neighborhood size of $\vec{x}_i^g$ (r_i^g) is adaptively assigned based on its ranking value, namely:

$$r_i^g = \max\{r_{max} - \Delta \times rank_i, r_{min}\} \tag{19}$$

where $rank_i$ is the ranking value of $\vec{x}_i^g$ in the sorted population, $r_{\max}$ ($r_{\min}$) represents the maximal (minimal) neighborhood size, and Δ is the parameter for controlling the step size of adaption.
As shown in Eq. (19), on the one hand, the individual with better ranking value will be allocated to a smaller neighborhood size. On the other hand, the neighborhood sizes are set to the same value (i.e., $r_{\min}$) for the best individuals, which can further promote the exploitive search.

2) *Neighbors locating operator*: For each individual ($\vec{x}_i^g \in P^g$), its linked neuron *u* (i.e., $lb_u = i$) is first identified. Then, the topological

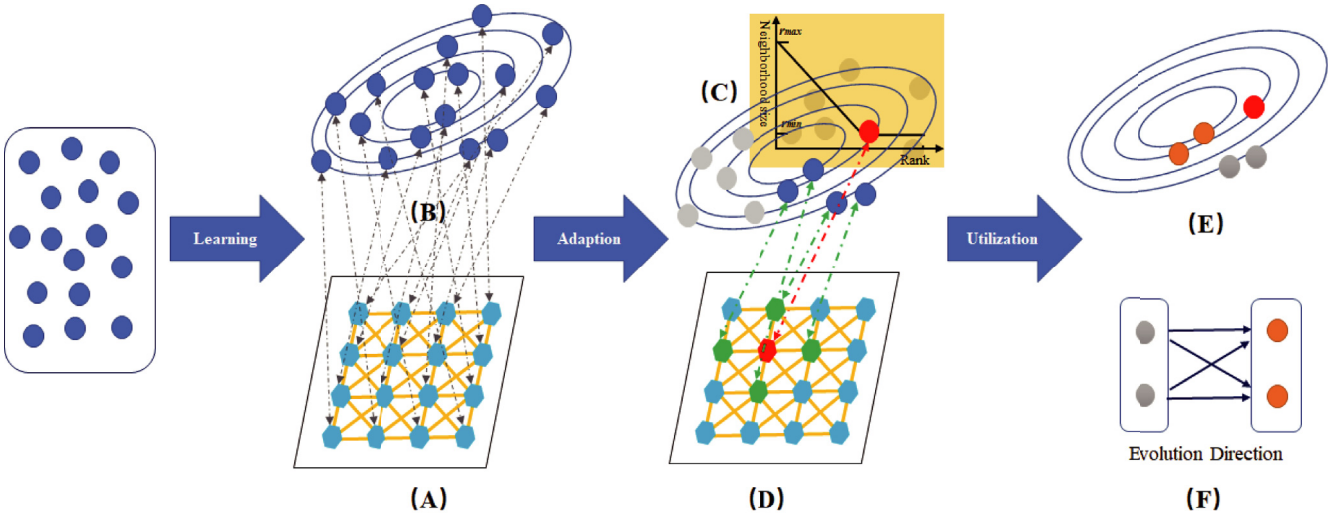

Fig. 2. The procedure of self-organizing neighborhood (SON). Neighborhood learning strategy: (A) SOM model updating, (B) neighborhood relationship building. Neighborhood adaption mechanism: (C) neighborhood size adapting, (D) neighbors locating. Neighborhood utilization strategy: (E) neighbors grouping, (F) evolution direction constructing.

distances between all the other neurons and u are calculated by Eq. (14). After that, these distances are sorted in ascending order, and the ranking value of neuron k ($k \in \{1,2,\ldots,NP\}\backslash\{u\}$) for u is denoted as $sort_{u,k}$. Note that the structure of latent space in SOM will remain unchanged and thus the *sort* value for each neuron can be calculated in advance, which can effectively reduce the computational overhead.

Next, the set of the r_i^g neighboring neurons for u (NN_u) is built as follows:

$$NN_u = \{k | sort_{u,k} \le r_i^g\} \tag{20}$$

After that, the neighborhood of $\vec{x}_i^g$ (Nei_i^g) for guiding the search is constructed below:

$$Nei_i^g = \{\vec{x}_j^g | k \in NN_u \wedge lb_k = j\} \tag{21}$$

In general, each individual is assigned to a distinct neighborhood size to match with its search role, as shown in Fig. 2 (C), and the neighborhood of each individual is constructed by locating the individuals that are linked to its neighboring neurons, as illustrated in Fig. 2 (D).

Algorithm 4 Neighborhood adaption mechanism

1:/****Neighborhood size adapting****/
2: Sort all the individuals of P^g in descending order based on their fitness values;
3: **FOR** $i = 1,2,\ldots,NP$ **DO**
4: Calculate the neighborhood size of $\vec{x}_i^g$ (r_i^g) using Eq. (19);
5: **END FOR**
6:/****Neighbors locating****/
7: **FOR** $i = 1,2,\ldots,NP$ **DO**
8: Identify the neuron u that is linked to $\vec{x}_i^g$ using $lb_u = i$;
9: Build NN_u for u using Eq. (20);
10: Construct Nei_i^g of $\vec{x}_i^g$ using Eq. (21);
11: **END FOR**

3.1.3. Neighborhood utilization strategy

To effectively use the learned neighborhood information of population, the neighborhood utilization strategy is employed, which includes two operators: neighbors grouping (Fig. 2 (E)) and evolution direction constructing (Fig. 2 (F)). Also, the procedure of neighborhood utilization strategy is described in Algorithm 5.

1) *Neighbors grouping operator*: For each individual ($\vec{x}_i^g \in P^g$), all the neighbors in Nei_i^g are first divided into two groups, i.e., starting group (SG_i^g) and terminal group (TG_i^g), based on their fitness values. These two groups are obtained by:

$$TG_i^g = \{\vec{x}_k^g | \vec{x}_k^g \in Nei_i^g \wedge f(\vec{x}_k^g) \le f(\vec{x}_i^g)\} \tag{22}$$

$$SG_i^g = Nei_i^g - TG_i^g \tag{23}$$

From Eqs. (22) - (23), the neighbors in TG_i^g have the better fitness values than $\vec{x}_i^g$, while the neighbors in SG_i^g perform worse than $\vec{x}_i^g$.

2) *Evolution direction constructing operator*: With the two groups of neighbors, a set of evolution direction vectors for X_i^g (ED_i^g) is created as follows:

$$ED_i^g = \{\vec{x}_p^g - \vec{x}_q^g | \vec{x}_p^g \in TG_i^g \wedge \vec{x}_q^g \in SG_i^g\} \tag{24}$$

From Eq. (24), the individuals in TG_i^g and SG_i^g are selected as the terminal point and the starting point, respectively, of evolution direction. If $TG_i^g = \varphi$ or $SG_i^g = \varphi$, the vector in ED_i^g will be constructed by randomly choosing two individuals from Nei_i^g and pointing to the better one from the worse one. Consequently, each evolution direction vector in ED_i^g can provide a promising search direction for X_i^g to generate the new offspring.

Algorithm 5 Neighborhood utilization strategy

1: **FOR** $i = 1,2,\ldots,NP$ **DO**
2:/****Neighbors grouping****/
3: Partition Nei_i^g into two groups using Eqs. (22) - (23);
4:/****Evolution direction constructing****/
5: Create ED_i^g for X_i^g using Eq. (24);
6: **END FOR**

3.2. SON-DE

By incorporating SON into DE, a novel DE framework, termed SON-based DE (SON-DE), is proposed for numerical optimization. The general framework of SON-DE is described in Algorithm 6.

As Algorithm 6 shows, SON (lines 4–8) that is applied for learning the neighborhood relationships between individuals and the DE operators (lines 9–12) that are used for producing offspring for the next generation are implemented alternately in SON-DE. To elaborate, SON starts with the *neighborhood learning strategy* to update the SOM model and building the neighborhood relationships between individuals (line 5). Then, the *neighborhood adaption mechanism* is carried out to adaptively adjust the neighborhood size and locate the neighbors for each individual (line 6). Followed that, the neighbors grouping and constructing evolution direction operators are implemented by the *neighborhood utilization strategy* (line 7). With these three components of the SON, the neighborhood information of the current population is learned and extracted for guiding the search of DE.

After executing SON, SON-DE enters the loop of generating offspring with the DE operators. For each individual, the base and difference vectors are selected from its defined neighborhood (i.e., Nei_i^g) and the set of evolution direction vectors (i.e., ED_i^g), respectively (line 10). Afterward, the DE operators (i.e., mutation, crossover, and selection) are executed with the selected vectors to obtain the new population for the next generation (line 11).

Besides, when the training data set is updated with the improved individuals (line 13), the SOM model for the *neighborhood learning strategy* will be incrementally updated with the new data set, rather than being training from scratch. In this way, the distributed information of the previously generated solutions can be reused to learn the neighborhood relationships of the current population.

Concerning the time complexity of SON-DE, the additional computation mainly lies in the three components of SON. In Algorithm 3, the SOM model updating operator will take $O(D \times NP \times |TD|)$ where $|TD| \leq NP$, while the neighborhood relationship building operator will take $O(D \times NP^2)$. Therefore, the time complexity of the *neighborhood learning strategy* is $O(D \times NP^2)$. In Algorithm 4, there are two operators, neighborhood size adapting and neighbors locating. The time complexity of the former and the latter are $O(NP \times \log NP)$ and $O(NP^2)$, respectively. Thus, the complexity of the *neighborhood adapting mechanism* is $O(NP^2)$. From Algorithm 5, the *neighborhood adapting mechanism*, which includes the neighbors grouping and evolution direction constructing operators, will take $O(NP^2)$. Based on the above analysis, the time complexity of SON is $O(D \times NP^2)$. Note that if no improved solutions are generated (i.e., $T = \varphi$), the SON will not be activated. For the DE operators in SON-DE, the time complexity is $O(D \times NP)$, which is similar to that of original DE operators. Overall, the total complexity of SON-DE is $O(D \times NP^2 \times G_{\max})$.

Algorithm 6 The SON-DE framework

1: *Initialization*: Randomly generate the initial population P^0; Set the training data set $TD = P^0$, the initial weight vector of each neuron $\vec{w}^u$, the neighborhood radius σ_0, the learning rate τ_0, and $g = 0$;
2: *Evaluation*: Calculate the fitness values of each individual in P^0;
3: **WHILE** $g \leq G_{\max}$
4: **IF** $TD \neq \varphi$ **THEN**
5: Execute the *neighborhood learning strategy* with TD (see **Algorithm 3**);
6: Execute the *neighborhood adaption mechanism* (see **Algorithm 4**);
7: Execute the *neighborhood utilization strategy* (see **Algorithm 5**);
8: **END IF**
9: **FOR** $i = 1$ to NP **DO**

(*continued on next page*)

Algorithm 6 *(continued)*

10: Select $\overrightarrow{bv}_i^g$ and $\overrightarrow{dv}_i^g$ for $\vec{x}_i^g$ from Nei_i^g and ED_i^g, respectively;
11: Execute the DE operators (Steps 5–7 in **Algorithm 1**);
12: **END FOR**
13: Update TD with the improved individuals as $TD = P^{g+1} \backslash P^g$;
14: $g = g + 1$;
15: **END WHILE**
16: *Output*: Return the individual with the best fitness value.

3.3. Differences between SON-DE and other NUT-based DE variants

As reviewed in Section 1, many NUTs have been developed for DE to improve its search ability. Compared with the existing NUT-based DE variants, SON-DE has the following advantages:

1) Both current and historical populations are involved in learning the neighborhood relationships between individuals. In this way, the evolutionary information from the previously generated solutions can be reused in updating the SOM model. On the contrary, the historical information is usually ignored in extracting neighborhood information by most NUT-based DE variants.
2) The differences between individuals in search role are considered in SON-DE to achieve the individual-dependent guidance with distinct neighborhoods. The neighborhoods of different sizes are allocated to different individuals to accord with their distinct search roles. By contrast, most NUT-based DE variants assign the neighborhood of the same size to each solution, without considering their different search roles.
3) The neighborhood for each individual is constructed based on the cosine distance, which can effectively guide the search towards the promising directions with similar angles. Besides, the evolution direction vectors constructed from the neighborhood are incorporated into the mutation operator to further enhance the effectiveness of using neighborhood information. Relatively, in most NUT-based DE variants, the neighbors defined by the population topology, Euclidean distance or the fitness information are usually randomly selected as parents for the mutation operator.

Recently, SOM has been incorporated into DE to solve different optimization problems [56,57]. However, there are some main differences between them, which are shown as follows:

1) In Ref. [56], SMEA is a multiobjective EA that only employs the DE operators (mutation and crossover) for offspring reproduction. Relatively, SON-DE is a general DE framework for global optimization that is enhanced by incorporating the SOM-based neighborhood. Further, constructing and using the neighborhood information are carried out in a completely different way. In SMEA, a neighborhood mating pool with the same size is constructed for each solution with a preset probability, and the neighbors involved in the DE operators are randomly selected from the mating pool. In contrast, SON-DE assigns the cosine similarity-based neighborhoods with different sizes to different solutions to match with their search roles, and the search for different solutions is further guided by the evolution direction constructed from the neighborhood.
2) In Ref. [57], a SOM-based DE variant (DE-SOM) is also introduced into DE to extract the neighborhood information of the population. Compared with DE-SOM, SON-DE has some significant improvements. First, the cosine distance is used in SON-DE to define the similarity between individuals, by replacing the Euclidean distance in DE-SOM. Second, the neighborhood adaption mechanism in SON-DE is adopted to adjust the neighborhood sizes for different individ-

uals based on their search roles, instead of selecting a neighborhood size for the entire population with a probability. Third, the neighborhood utilization strategy in SON-DE is employed to further promote the use of neighborhood information, rather than only randomly selecting the neighbors for the mutation in DE-SOM.

4. Experimental study

In this section, a set of benchmark functions from the CEC2013 special session on real-parameter optimization [50] are chosen to evaluate the performance of SON-DE. The 28 functions in the CEC2013 test set can be classified into three categories: unimodal functions ($F1$ – $F5$), basic multimodal functions ($F6$ – $F20$), and composition functions ($F21$ – $F28$). Besides, 30 test functions from the CEC2017 special session on real-parameter optimization ($NF1$ – $NF30$) [51] and 17 real-world numerical optimization problems from the CEC2011 competition ($RW1$ - $RW17$) [52] are used to further test the effectiveness of SON-DE on different optimization problems.

In the experiments, each algorithm is run independently on each test function for 30 times. For each run, it will be terminated when the maximum number of fitness evolutions is reached, which is limited to $10^4 \times D$ in this paper, and the final function error value ($f(X_{best}) - f(X^*)$) for each test function is recorded for comparison. Here, $f(X_{best})$ and $f(X^*)$ represent the fitness value obtained by the best solution and the optimal solution, respectively. In addition, the parameters for the basic DE algorithms and SON-DE are summarized below if no change is mentioned: 1) basic DE algorithm: $NP = 100$, $F = 0.5$, $Cr = 0.9$; 2) SON-DE: 2-dimensional 10×10 structure, $\sigma_0 = 5$, $\tau_0 = 0.9$, $r_{\max} = 70$, $r_{\min} = 5$, $\Delta = 1$.

To show the significant differences among the competitive algorithms, two non-parametric statistical tests are used by the KEEL software[1]: 1) the single- and multiple-problem analysis by Wilcoxon signed-rank test, which is used to evaluate the difference between two algorithms for single function and multiple functions, respectively; 2) the Friedman's test, which is used to obtain the ranking values of different algorithms for a set of functions [58–60]. The results of single-problem analysis by the Wilcoxon test with the significance level of 0.05 are summarized as "+/ = /-" in the tables, which represents that the considered algorithm significantly outperforms, equals to and is significantly worse than the compared algorithm on the corresponding number of functions, respectively. In the multiple-problem analysis by Wilcoxon test, $R+$ and $R-$ represent the sum of ranks that the considered algorithm performs significantly better than and worse than its competitor for the test functions, respectively [58]. In this paper, only the statistical results for the comparisons are presented for brevity, and the detailed numerical values of simulations can be found in the supplementary file.

The experiments in this study are divided into the following parts:

1) Sections 4.1 - 4.3 analyze the effectiveness of the three key components in SON, i.e., neighborhood learning strategy, neighborhood adapting mechanism, and neighborhood utilization strategy, respectively.
2) Section 4.4 investigates the performance sensitivity of SON-DE to its parameter of neighborhood size.
3) Section 4.5 studies what benefits can be obtained by applying SON-DE to original DE algorithms and advanced DE variants.
4) Section 4.6 compares the performance of SON-DE with several state-of-the-art DE and EAs.
5) Sections 4.7 and 4.8 investigate the advantages of SON-DE when compared with several clustering-based DE variants and three NUT-based DE frameworks, respectively.
6) Sections 4.9 and 4.10 further test the performance of SON-DE on other optimization problems, including the CEC2017 real-parameter optimization problems and the CEC2011 real-world numerical optimization problems.

4.1. Effect analysis of neighborhood learning strategy

To identify the benefit of the neighborhood learning strategy with cosine distance-based SOM, two SON-DE variants are investigated in this section. The first variant is named SON/KM-DE, where the k-means clustering technique [61] is used to replace SOM for learning the neighborhood information. The second variant is named SON/ED-DE, where the cosine distance is replaced by the Euclidean distance in calculating the similarity between two vectors. The statistically significant results are summarized in Tables 1and2.

As shown in Table 1, SON-DE obtains significantly better results than SON/KM-DE in most cases. Specifically, in the cases of original DE algorithms (i.e., DE/rand/, DE/rand/2, DE/best/1, DE/best/2, DE/current-to-best/1, and DE/rand-to-best/1), SON-DE significantly outperforms SON/KM-DE on 15, 26, 24, 23, 20, and 21 functions, respectively, while is outperformed by it on 2, 0, 4, 0, 0, and 1 function, respectively. In the cases of advanced DE variants (i.e., CoDE [15], ODE [62], SaDE [13], JADE [5], and SHADE [6]), SON-DE is significantly better than SON/KM-DE on 74 (= 21 + 8 + 9 + 19 + 17) functions and is significantly worse than it on 21(= 2 + 4 + 8 + 4 + 3) functions. Also, based on the multiple-problem statistical analysis, SON-DE obtains the higher $R+$ than $R-$ in 10 out of 11 cases. The p-value indicates that the significant difference between SON-DE and SON/KM-DE can be observed in 9 cases both at $\alpha = 0.05$ and $\alpha = 0.1$.

From Table 2, in the cases of DE/rand/1, DE/rand/2, DE/best/2, and ODE, SON-DE is significantly better than SON/ED-DE on 18, 25, 21 and 15 functions, respectively. Moreover, the significant difference between SON-DE and SON/ED-DE can be observed in all these cases according to the p-value. In the cases of DE/rand-to-best/1, CoDE, SaDE, JADE and SHADE, SON-DE also obtains the higher $R+$ than $R-$, which indicates that SON-DE performs better than SON/ED-DE on the test functions overall.

In general, from the above comparisons, SON-DE performs obviously better than SON/KM-DE and SON/ED-DE in most cases, which demonstrates that: 1) neighborhood learning strategy with SOM can provide a more effective way in learning and extracting neighborhood information of population for guiding the search; 2) the cosine distance is capable of further improving the performance of SON-DE by considering the similarity of solutions in search direction.

4.2. Effect analysis of neighborhood adapting mechanism

To verify the effectiveness of the neighborhood adaption mechanism, the performance comparisons between SON-DE and its variant with the fixed neighborhood size (denoted as SON/r-DE) are made. The values of r considered in this experiment include $r \in \{10, 40, 70, 90\}$. The statistical comparison results on the test functions are shown in Table 3.

According to Table 3, some observations can be obtained. First, in the case of DE with the explorative strategy (i.e., DE/rand/1), SON-DE can perform better than SON/r-DE overall. When r is set to a smaller value (e.g., 10 or 40), SON-DE gets the higher $R+$ than $R-$, although no significant difference between SON-DE and SON/r-DE. When r is set to a larger value (e.g., 70 or 90), SON-DE is significantly better than SON/r-DE based on both single- and multiple-problem statistical analysis. Second, for the DE variants with the exploitative strategy (i.e., DE/best/1 and ODE), SON-DE can achieve significantly better results than SON/r-DE when r is set to a greater value (e.g., 40, 70, or 90), while SON-DE

[1] KEEL: a software tool to assess EAs to data mining problems, which can be available from http://www.keel.es/.

Table 1
Results of single- and multiple-problem analysis by the Wilcoxon test between SON-DE and SON/KM-DE (i.e., the SON-DE variant with k-means clustering technique) for the CEC2013 functions at 30D.

SON-DE *vs.* SON/KM-DE	+ /=/−	$R+$	$R-$	*p*-value	$\alpha = 0.05$	$\alpha = 0.1$
DE/rand/1	15/11/2	344.5	61.5	1.22E-03	yes	yes
DE/rand/2	26/2/0	365.0	13.0	2.20E-05	yes	yes
DE/best/1	24/0/4	354.0	52.0	5.61E-04	yes	yes
DE/best/2	23/5/0	369.0	37.0	1.43E-04	yes	yes
DE/current-to-best/1	20/8/0	384.0	22.0	2.90E-05	yes	yes
DE/rand-to-best/1	21/6/1	335.0	43.0	4.15E-04	yes	yes
CoDE	21/5/2	339.5	66.5	1.81E-03	yes	yes
ODE	8/16/4	272.0	134.0	1.12E-01	no	no
SaDE	9/11/8	189.0	189.0	9.90E-01	no	no
JADE	19/5/4	358.5	47.5	3.67E-04	yes	yes
SHADE	17/8/3	322.5	55.5	1.29E-03	yes	yes

Table 2
Results of single- and multiple-problem analysis by the Wilcoxon test between SON-DE and SON/ED-DE (i.e., the SON-DE variant with Euclidean distance-based SOM) for the CEC2013 functions at 30D.

SON-DE *vs.* SON/ED-DE	+ /=/−	$R+$	$R-$	*p*-value	$\alpha = 0.05$	$\alpha = 0.1$
DE/rand/1	18/8/2	347.5	30.5	1.33E-04	yes	yes
DE/rand/2	25/3/0	357.0	21.0	2.60E-05	yes	yes
DE/best/1	6/11/11	151.5	254.5	1.00E+00	no	no
DE/best/2	21/6/1	363.5	14.5	5.20E-05	yes	yes
DE/current-to-best/1	7/10/11	167.0	239.0	1.00E+00	no	no
DE/rand-to-best/1	7/18/3	197.0	181.0	8.38E-01	no	no
CoDE	13/11/4	230.5	147.5	3.11E-01	no	no
ODE	15/13/0	373.0	33.0	9.90E-05	yes	yes
SaDE	10/14/4	273.0	133.0	1.07E-01	no	no
JADE	3/24/1	262.5	143.5	1.72E-01	no	no
SHADE	5/23/0	194.0	184.0	8.95E-01	no	no

performs worse than SON/r-DE when r is set to a smaller value (e.g., 10). Third, in the case of SHADE, SON-DE can achieve the better results than SON/r-DE overall in all the cases based on the multiple-problem statistical analysis. Besides, significantly better results can be got by SON-SHADE when $r = 10$ and $r = 40$.

From the above observations, it is clear that SON-DE can perform better than SON/r-DE in most cases, which demonstrates the advantages of neighborhood adapting mechanism in 1) effectively alleviating the influence of setting neighborhood size in SON-DE while further improving its performance, and 2) adaptively adjusting the neighborhood sizes for different solutions based on their search roles.

4.3. Effect analysis of neighborhood utilization strategy

To evaluate the effectiveness of neighborhood utilization strategy, the comparison between SON-DE and its variant without the neighborhood utilization strategy (denoted as SON-DEw/oD) is made. The statistical comparison results are shown in Table 4.

From Table 4, SON-DE performs significantly better than SON-DEw/oD in most cases of original DE algorithms. Specifically, SON-DE significantly outperforms SON-DEw/oD on 93 (= 9 + 12 + 19 + 9 + 20 + 24) functions and is outperformed by it on 13 (= 1 + 6 + 0 + 4 + 2 + 0) functions. Besides, SON-

Table 3
Results of single- and multiple-problem analysis by the Wilcoxon test between SON-DE and SON/r-DE (i.e., the SON-DE variant with the fixed neighborhood size r) for the CEC2013 functions at 30D.

SON-DE *vs.* SON/r-DE		+ /=/−	$R+$	$R-$	*p*-value	$\alpha = 0.05$	$\alpha = 0.1$
DE/rand/1	$r = 10$	5/19/4	271.5	134.5	1.15E-01	no	no
	$r = 40$	3/24/1	228.0	150.0	3.41E-01	no	no
	$r = 70$	8/18/2	310.5	95.5	1.37E-02	yes	yes
	$r = 90$	16/9/3	277.0	101.0	3.35E-02	yes	yes
DE/best/1	$r = 10$	1/10/17	49.5	365.5	1.00E+00	yes*	yes*
	$r = 40$	7/20/1	329.5	76.5	3.83E-03	yes	yes
	$r = 70$	21/7/0	353.0	25.0	7.40E-05	yes	yes
	$r = 90$	22/6/0	338.0	40.0	3.28E-04	yes	yes
ODE	$r = 10$	3/18/7	159.5	218.5	1.00E+00	no	no
	$r = 40$	7/17/4	260.5	145.5	1.85E-01	no	no
	$r = 70$	13/11/4	288.5	117.5	4.76E-02	yes	yes
	$r = 90$	16/10/2	298.0	80.0	8.52E-03	yes	yes
SHADE	$r = 10$	2/25/1	282.0	124.0	6.87E-02	no	yes
	$r = 40$	4/22/2	292.5	113.5	3.87E-02	yes	yes
	$r = 70$	5/22/1	242.0	164.0	3.68E-01	no	no
	$r = 90$	7/20/1	237.0	141.0	2.43E-01	no	no

* means that SON/r-DE is significantly better than SON-DE at the corresponding level of significance.

Table 4
Results of single- and multiple-problem analysis by the Wilcoxon test between SON-DE and SON-DEw/oD (i.e., the SON-DE variant without evolution direction) for the CEC2013 functions at 30D.

SON-DE *vs.* SON-DEw/oD	+ /=/−	$R+$	$R-$	*p*-value	$\alpha = 0.05$	$\alpha = 0.1$
DErand/1	9/18/1	329.5	48.5	6.79E-04	yes	yes
DE/rand/2	12/10/6	208.0	170.0	6.36E-01	no	no
DE/best/1	19/9/0	351.0	27.0	9.50E-05	yes	yes
DE/best/2	9/15/4	215.0	163.0	5.24E-01	no	no
DE/current-to-best/1	20/6/2	317.5	88.5	8.64E-03	yes	yes
DE/rand-to-best/1	24/4/0	385.0	21.0	3.20E-05	yes	yes
CoDE	12/16/0	316.5	61.5	1.86E-03	yes	yes
ODE	4/19/5	191.0	215.0	1.00E+00	no	no
SaDE	7/13/8	209.5	168.5	6.14E-01	no	no
JADE	6/20/2	201.5	176.5	7.53E-01	no	no
SHADE	10/16/2	257.5	148.5	2.08E-01	no	no

Table 5
Parameter setting for SON-DE with different combinations of $r_{\max}$ and $r_{\min}$.

Algorithm		SON-DE1	SON-DE2	SON-DE3	SON-DE4	SON-DE5	SON-DE6	SON-DE7	SON-DE8
Parameter Setting	$r_{\max}$	100	100	100	100	90	90	90	90
$r_{\min}$	20	15	10	5	20	15	10	5	
Algorithm		SON-DE9	SON-DE10	SON-DE11	SON-DE12	SON-DE13	SON-DE14	SON-DE15	SON-DE16
Parameter Setting	$r_{\max}$	80	80	80	80	70	70	70	70
$r_{\min}$	20	15	10	5	20	15	10	5	

DE obtains the higher $R+$ than $R-$ in all these cases. Based on the *p*-value, significant differences between SON-DE and SON-DEw/oD can be observed in 4 out of 6 cases. In the cases of advanced DE variants, SON-DE is significantly better than and worse than SON-DEw/oD on 39 ($= 12 + 4 + 7 + 6 + 10$) and 17 ($= 0 + 5 + 8 + 2 + 2$) functions, respectively. According to the multiple-problem analysis, SON-DE obtains the higher $R+$ than $R-$ in 4 out of 5 cases.

Overall, compared with SON-DEw/oD, the high performance of SON-DE is mainly owing to the incorporation of evolution direction into the mutation operator, which demonstrates the effectiveness of neighborhood utilization strategy in SON-DE.

4.4. Sensitivity analysis of parameters

In the neighborhood adaption mechanism, there are three parameters: the maximal neighborhood size ($r_{\max}$), the minimal neighborhood size ($r_{\min}$), and the step size of adaption (Δ). To investigate the performance sensitivity of SON-DE to these parameters, two experiments are carried out as follows: 1) study the sensitivity of SON-DE to both $r_{\max}$ and $r_{\min}$ while keeping the value of Δ (i.e., $\Delta = 1$); 2) study the sensitivity of SON-DE to Δ when keeping the values of $r_{\max}$ and $r_{\min}$. Also, DE/rand/1 and JADE are chosen as the instance algorithms, and three different types of functions from CEC2013 are used, i.e., the unimodal function *F*3, the basic multimodal function *F*13, and the composition function *F*26.

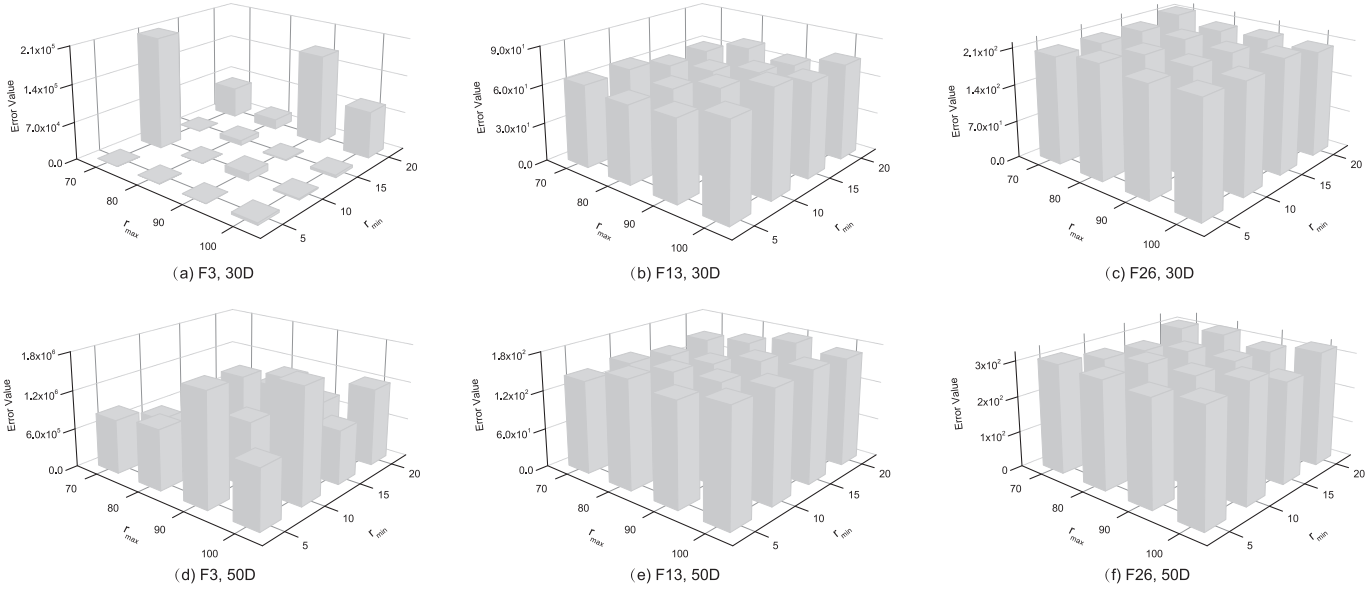

Fig. 3. Average error value obtained by SON-DE/rand/1 with different combinations of $r_{\max}$ and $r_{\min}$.

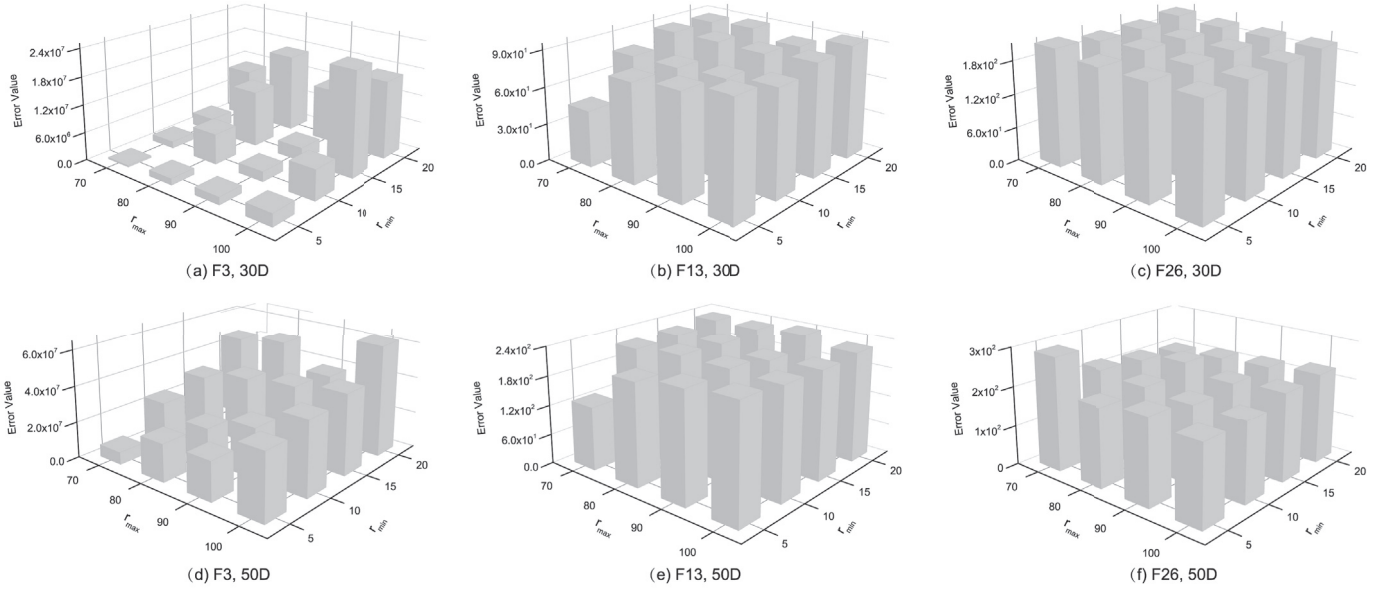

Fig. 4. Average error value obtained by SON-JADE with different combinations of $r_{\max}$ and $r_{\min}$.

4.4.1. Influence of r_{max} and r_{min}

The values of $r_{\max}$ and $r_{\min}$ considered in this experiment are as follows: $r_{\max} \in \{70, 80, 90, 100\}$ and $r_{\min} \in \{5, 10, 15, 20\}$. Besides, different combinations of $r_{\max}$ and $r_{\min}$ for SON-DE are shown in Table 5. Figs. 3and4 show the results obtained by SON-DE/rand/1 and SON-JADE with different combinations of $r_{\max}$ and $r_{\min}$, and the numerical results can be found in Tables S1–S2 of the supplemental file.

From the results shown in Figs. 3and4, the following observations can be obtained. First, for the unimodal function *F*3 at 30D, both SON-DE/rand/1 and SON-JADE get the best average value when $r_{\min} = 5$ and $r_{\max} = 70$. The similar results are also obtained by them for *F*3 at 50D. Second, for the basic multimodal function *F*13, both SON-DE/rand/1 and SON-JADE perform best averagely when $r_{\max} = 70$. Besides, when $r_{\max} = 70$, the performance of SON-JADE will get worse as the increase of the $r_{\min}$ value. Third, for the composition functions *F*26, both SON-DE/rand/1 and SON-JADE can achieve the similar results with most combinations of parameters.

In general, these results indicate that: 1) the performance of SON-DE for the complicated problems is not sensitive to the setting of $r_{\max}$ and $r_{\min}$; 2) the combination with a relatively small $r_{\max}$ and $r_{\min}$ (e.g., $r_{\max} = 70$ and $r_{\min} = 5$) is recommended for SON-DE when solving the simple problems.

4.4.2. Influence of Δ

The values of Δ in this experiment are $\Delta \in \{1, 3, 5, 7, 10\}$ while keeping $r_{\max} = 70$ and $r_{\min} = 5$. The results of SON-DE/rand/1 and SON-JADE with different Δ values are shown in Figs. 5and6 and Table S3 of the supplemental file.

According to Figs. 5and6, SON-DE performs best with $\Delta = 7$ for *F*3 at 30D in both cases. For the *F*3 at 50D, SON-DE/rand/1 with $\Delta = 10$ and SON-JADE with $\Delta = 5$ can obtain the best results. For *F*13 at 30D and 50D, SON-DE/rand/1 with $\Delta = 10$ achieves the best results, while SON-JADE with $\Delta = 1$ performs best. For *F*26 at 30D and 50D, both SON-DE/rand/1 and SON-JADE can obtain the better results with a larger Δ value.

The above comparisons demonstrate that: 1) a larger Δ value is appropriate to solve the composition functions, while a smaller Δ value is suitable to solve the unimodal and basic multimodal functions; 2) the performance of SON-DE with a smaller Δ value is more robust than that with a larger Δ value; 3) there are no significant differences among the SON-DE variants with different Δ values in most cases.

4.5. Performance enhancement of the DE algorithms

To verify the effectiveness of the proposed framework, SON-DE is applied to several original DE algorithms and advanced DE variants. The comparisons are made between SON-DE and the corresponding DE algorithm on the CEC2013 test functions at 30D and 50D.

4.5.1. Applying SON-DE to original DE algorithms

The statistical comparison results between SON-DE and the corresponding original DE algorithms are shown in Table 6, and the numerical results are given in Tables S4–S5 of the supplemental file. Also, the box plots of the final solutions obtained by DE/rand/1 and its augmented SON-DE on three different types of functions (i.e., the unimodal function F2, the basic multimodal function F13, and the composition function F2) are shown in Fig. 7.

From Table 6, SON-DE performs significantly better than the corresponding DE algorithms in all the cases. To elaborate, SON-DE is significantly better than and worse than the corresponding original DE algorithms on 139 (= 16 + 26 + 24 + 23 + 24 + 26) and 8 (= 1 + 2 + 1 + 2 + 1 + 1) functions at 30D, respectively. For the functions at 50D, SON-DE significantly outperforms the corresponding DE algorithms on 131 (= 14 + 24 + 24 + 21 + 24 + 24) functions, while is outperformed by them on 11 (= 8 + 1 + 1 + 0 + 0 + 1) functions.

Considering the features of the test functions, the following observations can be obtained from Tables S4–S5. First, for the unimodal functions (*F*1 − *F*5), SON-DE can significantly improve the performance of most original DE algorithms. In the case of DE/rand/1, the performance of SON-DE deteriorates on 2 functions at 30D and 3 functions at 50D. In the cases of DE/best/1 and DE/best/2, SON-DE is significantly outperformed by them only on *F*4. Second, for the basic multimodal functions (*F*6 − *F*20), SON-DE obtains the significant improvements in most cases, except *F*7 and *F*10 in the case of DE/rand/1 and *F*8 in the case of DE/best/1. Third, for the composition functions (*F*21 − *F*28), SON-DE shows the superiority over the corresponding algorithm on most test functions.

According to the multiple-problem statistical analysis in Table 6, SON-DE can obtain the higher $R+$ than $R-$ in all the cases. Besides, the significant differences between SON-DE and the corresponding DE algo-

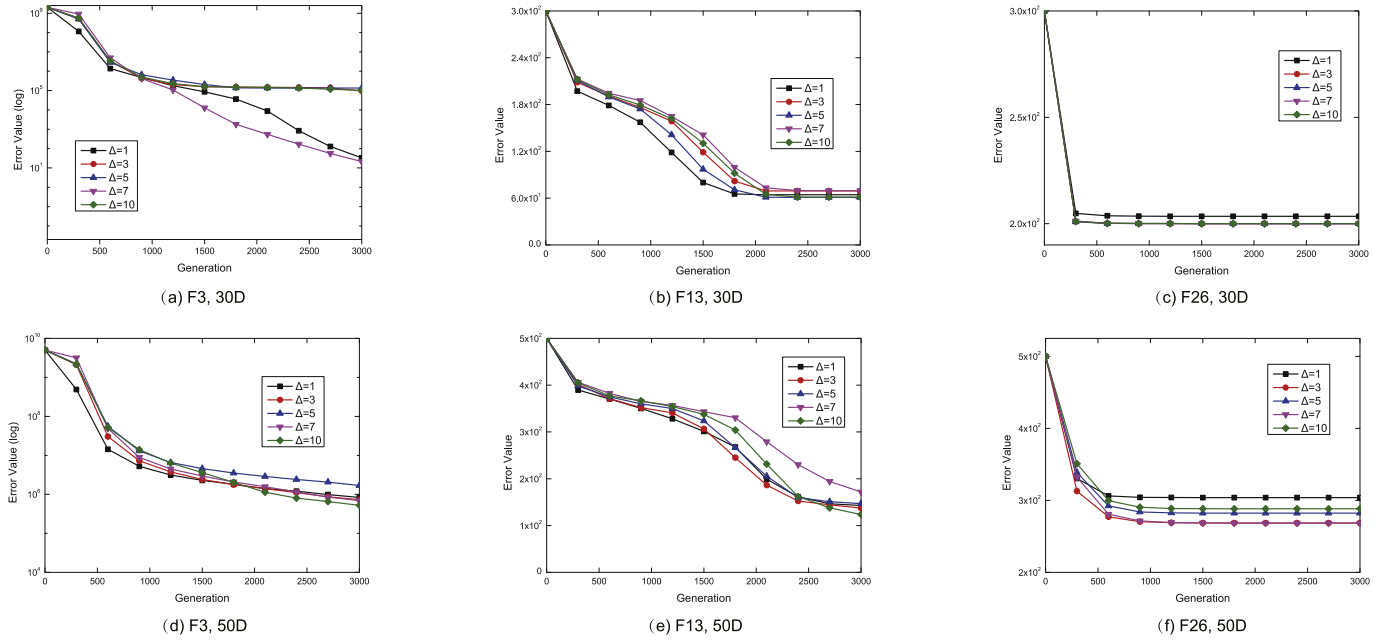

Fig. 5. Convergence graphs of SON-DE/rand/1 with different Δ values.

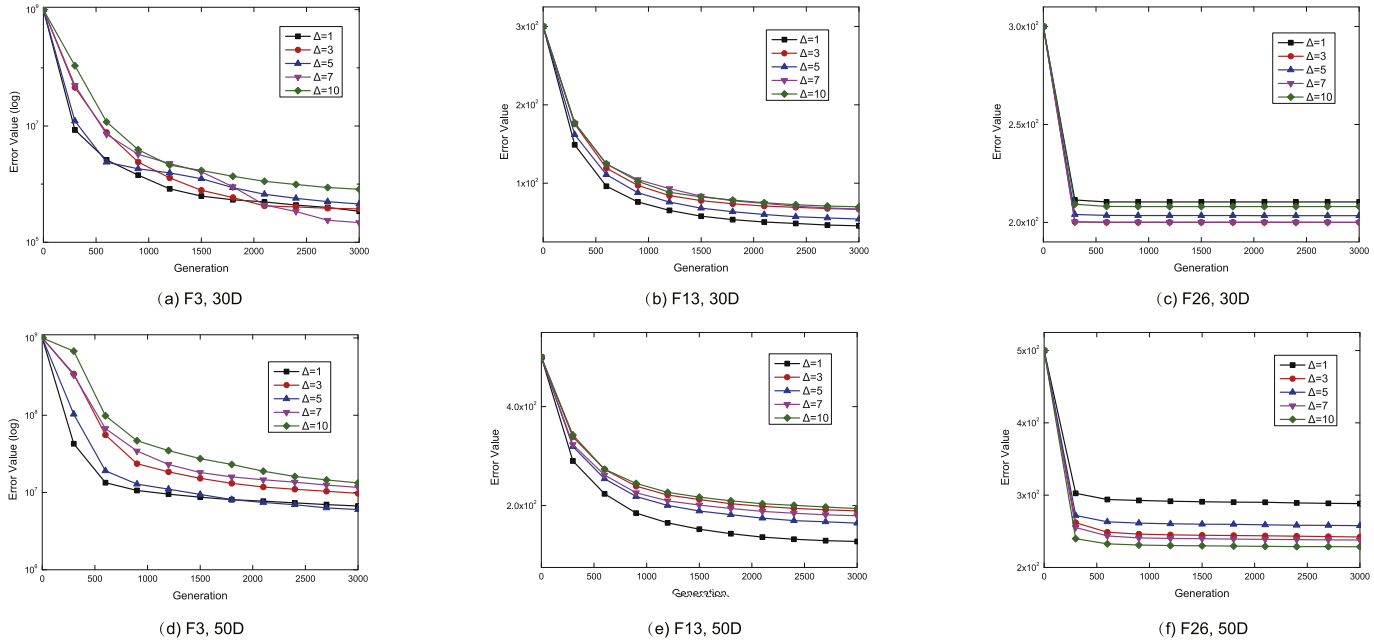

Fig. 6. Convergence graphs of SON-JADE with different Δ values.

rithm can be observed in all the cases both at $\alpha = 0.1$ and $\alpha = 0.05$.

Moreover, Fig. 7 clearly shows that SON-DE/rand/1 exhibits lower error value than DE/rand/1 on all the three functions and the improvements in the solution quality can be obtained by applying SON-DE to the original DE algorithm.

From the above results, SON-DE can significantly improve the performance of the corresponding DE algorithms on most test functions, which clearly demonstrates the advantages of applying SON-DE to original DE algorithms.

4.5.2. *Applying SON-DE to advanced DE variants*

The comparisons between SON-DE and the advanced DE variants are also made, and the results are shown in Tables 7 and S6-S7 (in the supplemental file). Furthermore, Fig. 8 shows the box plots of the final solutions obtained by JADE and its augmented SON-DE on the selected functions.

From Table 7, SON-DE significantly improves the performance of the DE variants on most test functions. Specifically, compared with CoDE, ODE, SaDE, JADE, and SHADE, SON-DE obtains significantly better results on 21, 12, 11, 10, and 13 functions at 30D, respectively, and 21, 15, 13, 14, and 14 functions at 50D, respectively. In contrast, CoDE, ODE, SaDE, JADE, and SHADE beat their augmented SON-DE on 1, 8, 6, 1 and 1 function at 30D, respectively, and no more than 6 functions at 50D. Besides, as illustrated in Fig. 8, SON-JADE can obtain the better results than JADE on all three functions.

Table 6
Results of single- and multiple-problem analysis by the Wilcoxon test between SON-DE and the original DE algorithm for the CEC2013 functions at 30D and 50D.

SON-DE at 30D vs.	+ /=/−	$R+$	$R-$	***p*-value**	$\alpha = 0.05$	$\alpha = 0.1$
DE/rand/1	16/5/7	309.5	68.5	3.65E-03	yes	yes
DE/rand/2	26/2/0	406.0	0.0	4.00E-06	yes	yes
DE/best/1	24/0/4	362.0	44.0	2.81E-04	yes	yes
DE/best/2	23/3/2	344.5	33.5	1.70E-04	yes	yes
DE/current-to-best/1	24/4/0	364.0	14.0	2.50E-05	yes	yes
DE/rand-to-best/1	26/2/0	362.0	16.0	3.10E-05	yes	yes
SON-DE at 50D vs.	+ /=/−	$R+$	$R-$	***p*-value**	$\alpha = 0.05$	$\alpha = 0.1$
DE/rand/1	14/6/8	288.5	117.5	5.02E-02	yes	yes
DE/rand/2	24/3/1	367.0	11.0	1.90E-05	yes	yes
DE/best/1	24/3/1	370.0	8.0	1.20E-05	yes	yes
DE/best/2	21/7/0	398.5	7.5	8.00E-06	yes	yes
DE/current-to-best/1	24/4/0	355.5	22.5	5.70E-05	yes	yes
DE/rand-to-best/1	24/3/1	361.0	17.0	3.40E-05	yes	yes

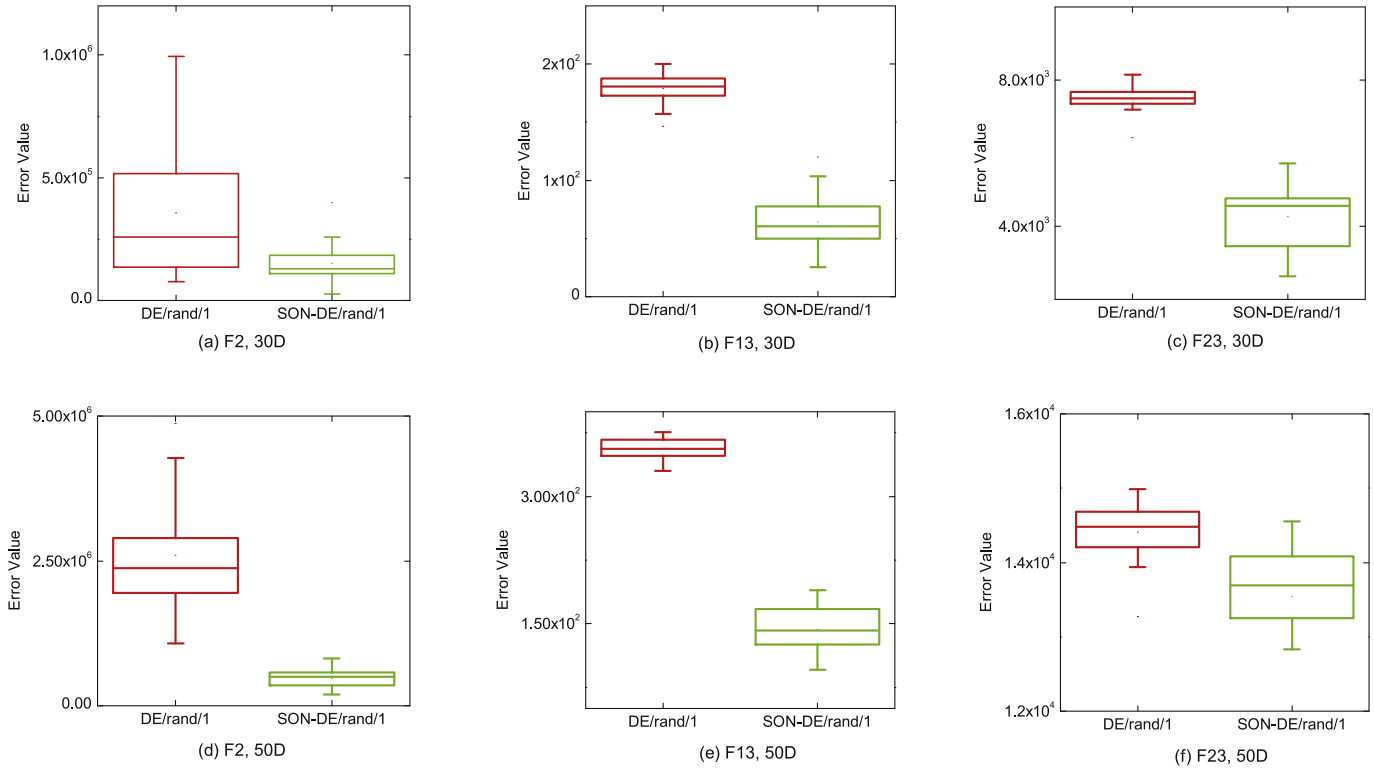

Fig. 7. Box plots of the final solutions obtained by DE/rand/1 and SON-DE/rand/1 on the selected functions at 30D and 50D over 30 independent runs.

From the results in Tables S6–S7, the following comments are made: 1) SON-DE significantly improves the performance of the corresponding competitors on most unimodal functions ($F1 - F5$) both at 30D and 50D, except $F1$ in the case of ODE; 2) for the basic multimodal functions ($F6 - F20$), SON-DE obtains the great improvements on most functions in the cases of CoDE, JADE, and SHADE, while SON-DE performs a little worse than ODE and SaDE on 4 and 5 functions at 30D respectively; 3) for the composition functions ($F21 - F28$), the superiority of SON-DE is shown in the cases of CoDE and SaDE, while the significant improvements cannot be obtained for ODE.

Furthermore, the results of the multiple-problem statistical analysis show that SON-DE achieves the higher $R+$ than $R-$ in all the cases. The *p*-value also indicates that the performance of SON-DE is significantly better than the advanced DE variants in most cases at $\alpha = 0.1$ or/and $\alpha = 0.05$. These results clearly demonstrate the effectiveness of SON-DE in improving the performance of most advanced DE variants considered.

4.5.3. *Overall comparisons*

To further compare the performance of different SON-DE versions overall, the Friedman test is conducted. The results are shown in Fig. 9, where the bar charts of the SON-DE versions are with stripes and the bar charts of the DE algorithm have the same color with its augmented SON-DE.

As illustrated in Fig. 9, some interesting observations can be obtained: 1) each SON-DE version gets the smaller ranking value than the corresponding DE algorithm for the functions both at 30D and 50D, which further demonstrates the benefit of the proposed framework to the DE algorithms; 2) the SON-DE versions with original strategy (i.e., DE/rand/1, DE/rand/2 and DE/best/2) can achieve the better ranking value than most advanced DE variants, which suggests that SON-DE is an effective framework for original DE algorithms to obtain the improved and promising results; 3) SON-SHADE obtains the first rank on the functions both at 30D and 50D, followed by SON-JADE. Besides, SON-SHADE is significantly better than SHADE overall according to the

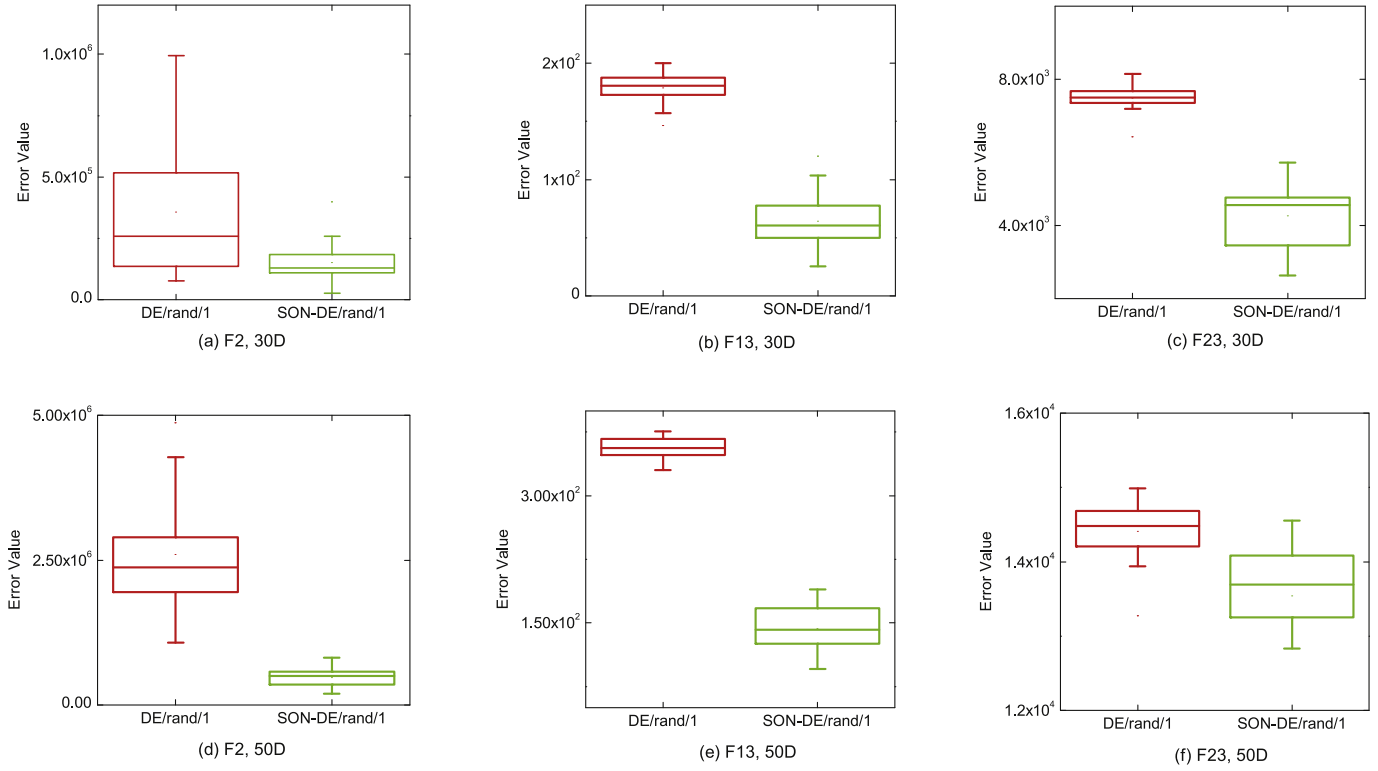

Fig. 8. Box plots of the final solutions obtained by JADE and SON-JADE on the selected functions at 30D and 50D over 30 independent runs.

Table 7
Results of single- and multiple-problem analysis by the Wilcoxon test between SON-DE and the advanced DE algorithm for the CEC2013 functions at 30D and 50D.

SON-DE at 30D vs.	**+ /=/−**	***R*+**	***R*−**	***p*-value**	**$\alpha = 0.05$**	**$\alpha = 0.1$**
CoDE	21/6/1	351.0	55.0	7.21E-04	yes	yes
ODE	12/8/8	297.0	81.0	9.14E-03	yes	yes
SaDE	11/11/6	268.0	138.0	1.36E-01	no	no
JADE	10/17/1	326.5	79.5	4.63E-03	yes	yes
SHADE	13/14/1	269.5	108.5	5.10E-02	no	yes
SON-DE at 50D vs.	**+ /=/−**	***R*+**	***R*−**	***p*-value**	**$\alpha = 0.05$**	**$\alpha = 0.1$**
CoDE	21/7/0	375.0	3.0	5.00E-06	yes	yes
ODE	15/7/6	277.5	128.5	8.77E-02	no	yes
SaDE	13/15/0	273.0	105.0	4.05E-02	yes	yes
JADE	14/12/2	357.0	49.0	4.35E-04	yes	yes
SHADE	14/12/2	300.5	77.5	7.13E-03	yes	yes

results in Table 7. Consequently, SON-SHADE will be selected as the representative to compare with other EAs in the following section.

4.6. Comparison with state-of-the-art EAs

In this section, SON-SHADE is compared with 13 state-of-the-art EAs, including 9 algorithms from the CEC'2013 special session and competition (i.e., NBIPOPaCMA [63], iCMAESILS [64], MVMO [65], SMADE [66], TLBSaDE [67], DEcfbLS [68], b6e6rl [69], SPSRDEMMS [70], and CMAES-RIS [71]) and 4 recently proposed DE-based algorithms (i.e., DMPSADE [72], MPEDE [14], IIN-SHADE [73], and AGDE [74]). These algorithms have been tested on the CEC2013 benchmark functions and can obtain the promising results. The results of these EAs are directly obtained from their original papers. The performance comparisons on the test functions are made, and the statistically significant results are summarized in Tables 8–10 and S8 - S9 (in the supplemental file).

As shown in Table 8, SON-SHADE obtains the higher $R+$ than $R-$ in 10 out of 13 cases for the functions at 30D, except in the cases of NBIPOPaCMA, iCMAESILS, and IIN-SHADE. Based on the p-value, SON-SHADE significantly outperforms six DE-based algorithms (i.e., SMADE, b6e6rl, DEcfbLS, SPSRDEMMS, DMPSADE, and AGDE) and one hybrid algorithms based on covariance matrix adaptation evolution strategy (CMAES) (i.e., CMAES-RIS) both at $\alpha = 0.05$ and $\alpha = 0.1$, while is outperformed by two CMAES-based hybrid algorithms (i.e., NBIPOPaCMA and iCMAESILS) at $\alpha = 0.1$. For the functions at 50D, Table 9 clearly suggests that SON-SHADE can obtain the consistent good results. Concretely, SON-SHADE achieves the $R+$ than $R-$ in 10 cases and significantly outperforms five compared EAs according to the multiple-problem statistical analysis.

Moreover, based on the results of the Friedman test in Table 10, SON-SHADE achieves the third rank for the functions both at 30D and 50D and is only outperformed by NBIPOPaCMA and iCMAESILS. Overall, NBIPOPaCMA and iCMAESILS are the first and second best respectively in terms of the meaning ranking, followed by the proposed SON-SHADE algorithm as the third best among all the compared algorithms.

From the above results, some conclusions can be drawn: 1) SON-SHADE obtains the best results overall among all the DE-based algo-

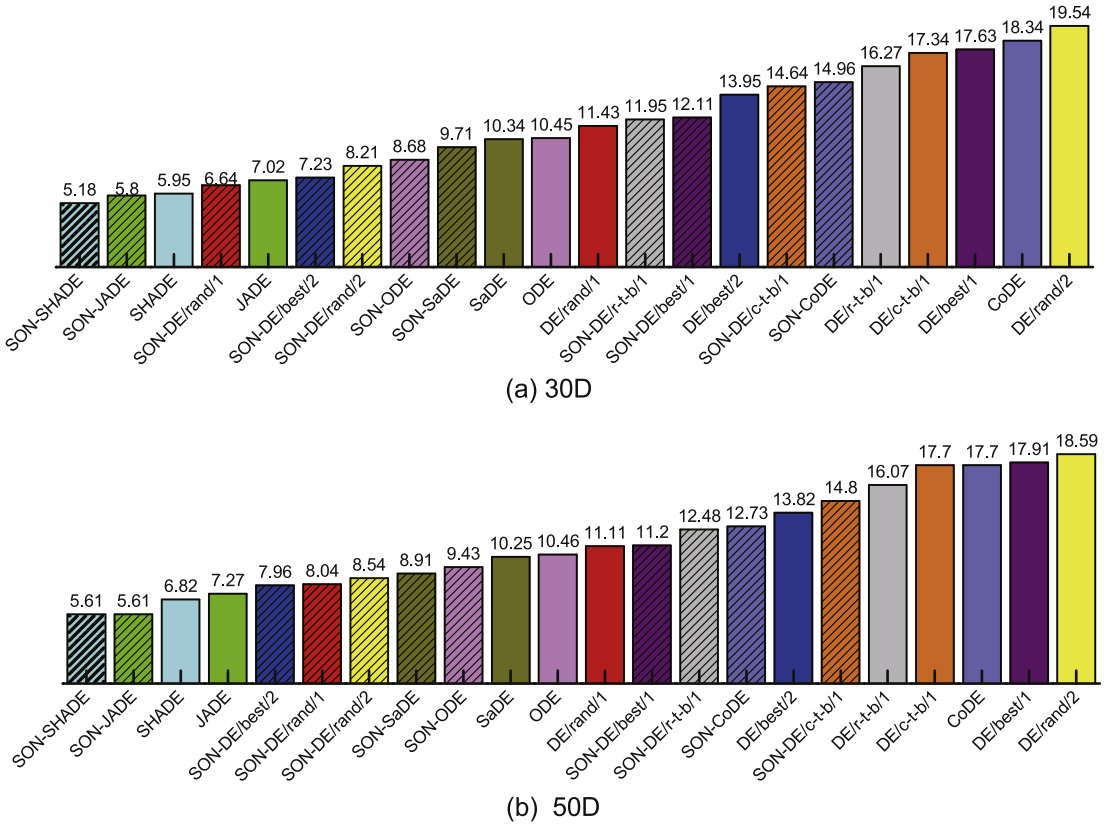

Fig. 9. Average ranking values of all the DE algorithms and their augmented SON-DE by Friedman test for the CEC2013 functions at 30D and 50D. DE/r-*t*-b/1 and DE/c-*t*-b/1 are the abbreviation of DE/rand-to-best/1 and DE/current-to-best/1, respectively.

Table 8
Results of multiple-problem analysis by the Wilcoxon test between SON-SHADE and the state-of-the-art EAs for the CEC2013 functions at 30D.

SON-SHADE *vs.*	$R+$	$R-$	*p*-value	$\alpha = 0.05$	$\alpha = 0.1$
NBIPOPaCMA	123.5	282.5	6.85E-02	no	yes*
iCMAESILS	123	283	6.60E-02	no	yes*
MVMO	214.5	191.5	7.85E-01	no	no
SMADE	320.5	85.5	7.05E-03	yes	yes
b6e6rl	301.5	76.5	6.63E-03	yes	yes
DEcfbLS	311	95	1.35E-02	yes	yes
TLBSaDE	269.5	136.5	1.27E-01	no	no
SPSRDEMMS	287.5	90.5	1.74E-02	yes	yes
CMAES-RIS	293.5	112.5	3.82E-02	yes	yes
DMPSADE	336.5	41.5	3.77E-04	yes	yes
MPEDE	231.5	146.5	3.02E-01	no	no
IIN-SHADE	174.5	203.5	7.19E-01	no	no
AGDE	307	99	1.73E-02	yes	yes

* means that the competitor is better than SON-DE at the corresponding level of significance.

rithms in this comparison, which demonstrates its outstanding performance for the complex problems; 2) although both NBIPOPaCMA and iCMAESILS outperform SON-SHADE, SON-SHADE is also proven to be a powerful optimization algorithm with the promising and competitive performance; 3) compared to the complicated structures and sophisticated search operators in NBIPOPaCMA and iCMAESILS, SON-SHADE only uses the learned neighborhood information of population to guide the evolutionary process of DE, which exhibits its advantages in simple structure and is easy to implement; 4) the main purpose of this study is to enhance the performance of DE with SON and not to present the best algorithm that defeats all the other EAs, and the effectiveness of SON-DE is further confirmed by the superior performance of SON-SHADE over other state-of-the-art EAs.

4.7. Comparison with clustering-based DE variants

To demonstrate the superiority of the proposed framework, the comparisons between SON-DE and other clustering-based DE variants are made. Here, several recently proposed clustering-based DE variants, including LeDE/bin [37], LeDE/exp [37], Self-CCDE [39], GRCDE [38], and MGRCDE [75], are used for comparison. Due to that the "DE/rand/1" strategy is employed in these variants, SON-DE with DE/rand/1 is employed here. The statistically significant results are shown in Tables 11–12 and S10-S11 (in the supplemental file).

From Table 11, SON-DE exhibits the better performance than other clustering-based DE variants for the functions both at 30D and 50D. Specifically, compared with GRCDE, MGRCDE, Self-CCDE, LeDE/bin

Table 9
Results of multiple-problem analysis by the Wilcoxon test between SON-SHADE and the state-of-the-art EAs for the CEC2013 functions at 50D.

SON-SHADE *vs.*	$R+$	$R-$	p-value	$\alpha = 0.05$	$\alpha = 0.1$
NBIPOPaCMA	115.5	290.5	4.51E-02	yes*	yes*
iCMAESILS	74	304	5.52E-03	yes*	yes*
MVMO	169	209	6.22E-01	no	no
SMADE	278.5	99.5	3.06E-02	yes	yes
b6e6rl	313.5	92.5	1.15E-02	yes	yes
DEcfbLS	256.5	121.5	1.02E-01	no	no
TLBSaDE	300.5	105.5	2.56E-02	yes	yes
SPSRDEMMS	347.5	58.5	9.60E-04	yes	yes
CMAES-RIS	236	142	2.54E-01	no	no
DMPSADE	207.5	170.5	6.47E-01	no	no
MPEDE	206.5	171.5	6.65E-01	no	no
IIN-SHADE	235	143	2.62E-01	no	no
AGDE	319.5	86.5	7.72E-03	yes	yes

* means that the compared algorithm is better than SON-DE at the corresponding level of significance.

Table 10
Average ranking value for SON-SHADE and the state-of-the-art EAs on the CEC2013 functions at 30D and 50D.

Algorithm	30D	50D	Mean Ranking	Final Ranking
NBIPOPaCMA	4.84	4.66	4.75	1
iCMAESILS	5.25	4.41	4.83	2
SON-SHADE	6.05	6.66	6.36	3
MVMO	6.38	6.66	6.52	4
MPEDE	6.63	6.80	6.71	5
IIN-SHADE	6.20	7.45	6.82	6
DMPSADE	8.27	6.93	7.60	7
TLBSaDE	7.80	8.80	8.30	8
SMADE	8.36	8.32	8.34	9
SPSRDEMMS	8.55	8.61	8.58	10
DEcfbLS	8.98	8.29	8.63	11
b6e6rl	8.98	9.16	9.07	12
CMAES-RIS	9.71	8.63	9.17	13
AGDE	9.00	9.63	9.31	14

and LeDE/exp, SON-DE is significantly better on 14, 12, 17, 18, and 16 functions at 30D, respectively, and on 13, 17, 13, 11, and 14 functions at 50D, respectively. According to the multiple-problem statistical analysis, SON-DE obtains the higher $R+$ than $R-$ in all the cases. The p-value also indicates that SON-DE obtains the significantly better results overall in 4 cases at $\alpha = 0.05$ and 8 cases at $\alpha = 0.1$. Besides, the results of the Friedman test in Table 12 show that SON-DE achieves the first rank among all the competitors for the functions both at 30D and 50D.

From the above comparisons, SON-DE significantly outperforms other clustering-based DE variants considered, which suggests that SON is more effective than other clustering-based mechanisms in learning and utilizing the neighborhood information of the population.

4.8. Comparison with NUT-based DE frameworks

In this section, SON-DE is compared with three NUT-based DE frameworks: ProDE [34], MSDE [45], and MTDE [18]. The statistical comparison results are summarized in Table 13. Also, the box plots of the final solutions obtained by SON-DE, ProDE, MSDE, and MTDE with DE/rand/1 and JADE on the selected functions are shown in Fig. 10.

From Table 13, some observations can be obtained:

- Compared to ProDE, SON-DE significantly outperforms it on 114 (= 12 + 23 + 10 + 23 + 22 + 24) functions and is significantly outperformed by it on 7 (= 3 + 1 + 3 + 0 + 0 + 0) functions in the cases of original DE algorithms. In the cases of advanced DE variants, SON-DE is significantly better than and worse than ProDE on 54 (= 19 + 8 + 9 + 10 + 8) and 17 (= 2 + 3 + 7 + 4 + 1) functions, respectively. Based on the multiple-problem statistical analysis, SON-DE obtains the higher $R+$ than $R-$ in all the cases. Besides, the p-value shows that the overall performance of SON-DE is significantly better than ProDE in 6 and 8 cases at $\alpha = 0.05$ and $\alpha = 0.1$, respectively.
- Compared to MSDE, SON-DE consistently achieves the better results than it. Specifically, SON-DE is significantly better than and worse than MSDE on 166 and 57 functions, respectively. Furthermore, the multiple problem statistical analysis demonstrates the significantly better performance of SON-DE in 6 and 7 cases at $\alpha = 0.05$ and $\alpha = 0.1$, respectively.
- Compared to MTDE, the overall performance of SON-DE is significantly better than it in 6 cases (i.e., DE/rand/1, DE/rand/2, DE/best/2, CoDE, JADE, and ODE) both at $\alpha = 0.05$ and $\alpha = 0.1$. In the cases of DE/current-to-best/1 and DE/rand-to-best/1, SON-DE is significantly outperformed by MTDE.

Moreover, Fig. 10 clearly shows that SON-DE achieves the lowest error value among the four DE frameworks in most cases, which further exhibits the outstanding performance of SON-DE.

In general, the above results demonstrate the superior performance of SON-DE when compared with the three NUT-based DE frameworks. Besides, the effectiveness of SON for DE is further verified by comparing it with other NUTs.

Table 11
Results of multiple-problem analysis by the Wilcoxon test between SON-DE/rand/1 and different clustering-based DE variants for the CEC2013 functions at 30D and 50D.

SON-DE/rand/1 at 30D *vs.*	+ /=/−	***R*+**	***R*−**	***p*-value**	**$\alpha = 0.05$**	**$\alpha = 0.1$**
GRCDE	14/8/6	304.5	73.5	5.32E-03	no	yes
MGRCDE	12/12/4	306.0	72.0	4.76E-03	yes	yes
Self-CCDE	17/3/8	285.5	120.5	5.88E-02	no	yes
LeDE/bin	18/8/2	323.0	55.0	1.23E-03	yes	yes
LeDE/exp	16/4/8	335.0	71.0	2.48E-03	yes	yes
SON-DE/rand/1 at 50D *vs.*	+ /=/−	***R*+**	***R*−**	***p*-value**	**$\alpha = 0.05$**	**$\alpha = 0.1$**
GRCDE	13/8/7	281.5	124.5	7.20E-02	no	yes
MGRCDE	17/5/6	324.5	81.5	5.49E-03	no	yes
Self-CCDE	13/3/12	216.5	189.5	7.50E-01	no	no
LeDE/bin	11/11/6	253.5	152.5	2.46E-01	no	no
LeDE/exp	14/6/8	271.5	106.5	4.61E-02	yes	yes

Table 12
Average ranking value (ARV) of SON-DE/rand/1 and the clustering-based DE variants by Friedman test for the CEC 2013 functions at 30D and 50D.

Algorithm at 30D	**ARV**	**Rank**	**Algorithm at 50D**	**ARV**	**Rank**
SON-DE/rand/1	2.411	1	SON-DE/rand/1	2.893	1
MGRCDE	3.286	2	Self-CCDE	3.357	2
Self-CCDE	3.607	3	LeDE/bin	3.393	3
GRCDE	3.607	3	GRCDE	3.625	4
LeDE/bin	4.000	5	MGRCDE	3.750	5
LeDE/exp	4.089	6	LeDE/exp	3.982	6

Table 13
Results of multiple-problem analysis by the Wilcoxon test for SON-DE *vs.* ProDE, SON-DE *vs.* MSDE, and SON-DE *vs.* MTDE for the CEC2013 functions at 30D.

Mutation	**SON-DE *vs.* ProDE**			**SON-DE *vs.* MSDE**			**SON-DE *vs.* MTDE**		
	+ /=/−	***R*+ /*R*−**	**Sig.⋄**	+ /=/−	***R*+ /*R*−**	**Sig.⋄**	+ /=/−	***R*+ /*R*−**	**Sig.⋄**
DE/rand/1	12/13/3	284.5/121.5	n/y	16/4/8	326.5/79.5	y/y	13/14/1	324/82	y/y
DE/rand/2	23/4/1	404.5/1.5	y/y	26/2/0	378/0	y/y	26/2/0	367.5/10.5	y/y
DE/best/1	10/15/3	171/7	y/y	8/9/11	147/231	n/n	3/22/3	148.5/257.5	n/n
DE/best/2	23/5/0	335/71	y/y	25/2/1	366/12	y/y	21/7/0	363/15	y/y
DE/c-*t*-b/1	22/6/0	327/51	y/y	11/8/9	191/187	n/n	3/9/16	128/278	n/y*
DE/r-*t*-b/1	24/4/0	406/0	y/y	14/5/9	284/122	n/y	5/4/19	91/287	y*/y*
CoDE	19/7/2	344/62	y/y	21/7/0	369.5/8.5	y/y	22/5/1	381/25	y/y
JADE	8/17/3	233/145	n/n	13/13/2	289.5/116.5	y/y	6/20/2	296.5/109.5	y/y
SaDE	9/12/7	253.5/124.5	n/n	11/11/6	250.5/155.5	n/n	4/13/11	169/237	n/n
SHADE	10/14/4	269/109	n/y	14/11/3	292/86	y/y	11/12/5	248/130	n/n
ODE	8/19/1	257.7/148.5	n/n	7/13/8	201.5/176.5	n/n	11/13/4	291/115	y/y

⋄ The values (a/b) in the "Sig." column are used to show whether the significant difference can be observed between SON-DEand its compared algorithm at $\alpha = 0.05$ and $\alpha = 0.1$, respective. "y" and "n" mean "yes" and "no", respectively. * means that the compared algorithm is significantly better than SON-DE at the corresponding level of significance.

4.9. More comparisons on the CEC2017 test functions

To further test the performance of SON-DE on different optimization functions, the CEC2017 test suite with 30 functions [51] is used for comparison. In this section, jSO [76], which has shown its high performance in the CEC2017 competition, is also included. The results are shown in Table 14 and S12-S13 (in the supplemental file).

As shown in Table 14, SON-DE consistently exhibits the better performance on the CEC2017 test functions when compared with the corresponding DE algorithms. To elaborate, SON-DE performs better than the DE variants on 84 (= 28 + 11 + 17 + 10 + 11 + 7) and 84 (= 28 + 15 + 16 + 9 + 14 + 2) functions at 30D and 50D, respectively, while SON-DE is outperformed by them on 18 and 24 functions at 30D and 50D, respectively. Besides, SON-DE can obtain the higher $R+$ than $R-$ in most cases except the jSO case at 50D. According to the *p*-value, SON-DE is significantly better than the corresponding variants overall in 3 cases (i.e., CoDE, JADE, and SaDE) for the functions at 30D both at $\alpha = 0.05$ and $\alpha = 0.1$. For the functions at 50D, SON-DE significantly outperforms the corresponding variants in 4 and 5 cases at $\alpha = 0.05$ and $\alpha = 0.1$, respectively. However, when applying SON-DE to jSO,[2] significant improvements cannot be obtained on most functions. The reason might be that the population size of jSO reduces linearly with the generations, and SON-DE with the fixed network structure cannot effectively preserve and learn the neighborhood relationships of the decreasing population. In the future, new neighborhood learning strategy in SON-DE will be investigated to further enhance the performance of DE with dynamic population size.

In summary, the above observations indicate that SON-DE is also an effective algorithmic framework in improving the performance of most advanced DE variants when solving the CEC2017 test functions.

[2] Since the linear population size reduction mechanism is used in jSO, we herein still set the number of neurons to 100 in SON-DE if the population size is larger than 100. Otherwise, we will remove the neuron that is not linked to any solutions after neighborhood learning strategy.

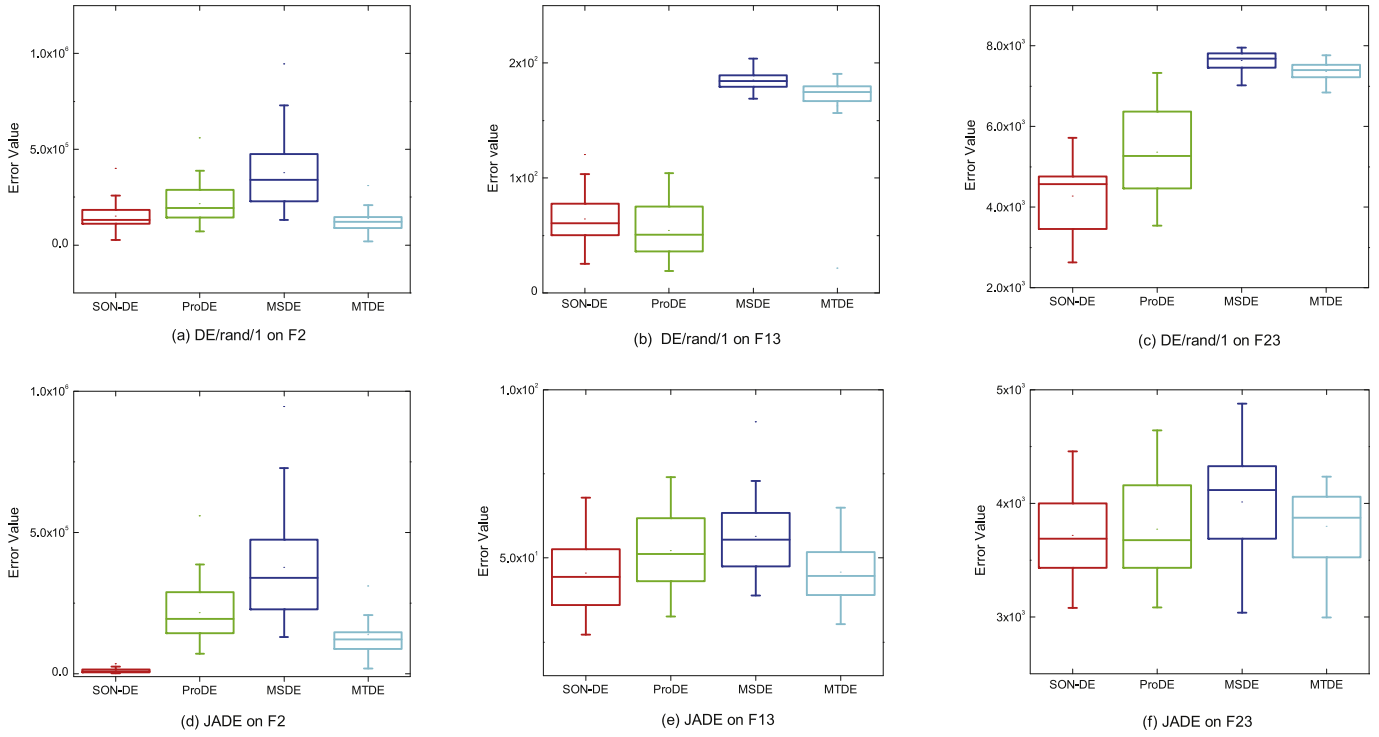

Fig. 10. Box plots of the final solutions obtained by SON-DE, ProDE, MSDE, and MTDE with DE/rand/1 and JADE on the selected functions at 30D over 30 independent runs.

Table 14
Results of multiple-problem analysis by the Wilcoxon test between SON-DE and the corresponding DE algorithms for the CEC2017 functions at 30D and 50D.

SON-DE at 30D *vs.*	+ /=/−	$R+$	$R-$	***p*-value**	$\alpha = 0.05$	$\alpha = 0.1$
CoDE	28/2/0	435.0	0.0	2.00E-06	yes	yes
JADE	11/16/3	371.0	94.0	4.25E-03	yes	yes
SaDE	17/13/0	377.0	58.0	5.04E-04	yes	yes
ODE	10/14/6	309.0	156.0	1.12E-01	no	no
SHADE	11/15/4	264.5	170.5	2.99E-01	no	no
jSO	7/18/5	252.0	183.0	4.47E-01	no	no
SON-DE at 50D *vs.*	+ /=/−	$R+$	$R-$	***p*-value**	$\alpha = 0.05$	$\alpha = 0.1$
CoDE	28/2/0	463.5	1.5	2.00E-06	yes	yes
JADE	15/11/4	332.0	103.0	1.26E-02	yes	yes
SaDE	16/13/1	417.0	48.0	1.07E-04	yes	yes
ODE	9/12/9	324.0	141.0	5.85E-02	no	yes
SHADE	14/11/5	344.5	120.5	2.03E-02	yes	yes
jSO	2/23/5	195.0	270.0	1.00E+00	no	no

4.10. Application of SON-DE for real-world problems

To evaluate the performance of SON-DE on the real-world problems, 17 problems from the CEC2011 competition (see Table S14 in the supplemental file) [52] are used in this experiment. The comparison results are given in Table 15 and S15-S16. Besides, the average ranking values of all the compared algorithms by the Friedman test are illustrated in Fig. 11.

From Table 15, SON-DE significantly improves the performance of the corresponding DE algorithms on most problems. To elaborate, in the cases of original DE algorithms, SON-DE is significantly better than them on 78 (= 16 + 15 + 11 + 15 + 11 + 10) out of 102 problems and is worse than them on 4 (= 0 + 0 + 3 + 0 + 1 + 0) problems. In the cases of advanced DE variants, SON-DE significantly outperforms them on 40 (= 8 + 12 + 11 + 9) problems and is outperformed by them on 4 (= 2 + 0 + 0 + 2) problems. Based on the multiple-problem statistical analysis, the overall performance of SON-DE is significantly better than that of the corresponding DE algorithm in all the cases. Moreover, the results of the Friedman test in Fig. 11 indicate that all the SON-DE versions can achieve the better ranking values than the corresponding DE algorithms.

Generally, the above comparisons demonstrate that SON-DE can further enhance the performance of the considered DE algorithms for complex real-world numerical problems.

5. Conclusion and future research

This paper proposed a novel neighborhood utilization technique, termed self-organizing neighborhood (SON), to promote the use of neighborhood information for guiding the search of DE. In SON, a neighborhood learning strategy was designed to incrementally learn the neighborhood information of population by using the SOM technique, a neighborhood adapting mechanism was used to adaptively adjust the neighborhood sizes for different solutions based on their search roles, and a neighborhood utilization strategy was also employed to intelligently select neighbors for mutation with the promising evolution

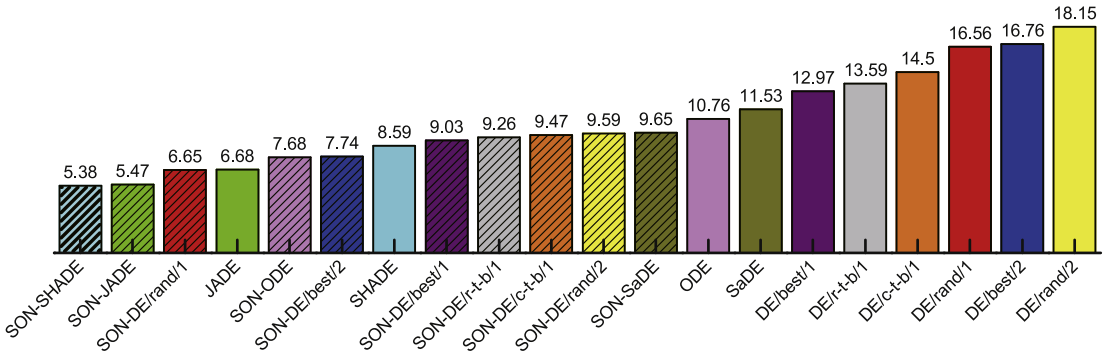

Fig. 11. Average ranking values of all the DE algorithms and their augmented SON-DE by Friedman test for the real-world problems from CEC2011.

Table 15
Results of multiple-problem analysis by the Wilcoxon test between SON-DE and the corresponding DE algorithm for the real-world problems from CEC2011.

SON-DE *vs.*	+ /=/−	$R+$	$R-$	*p*-value	$\alpha = 0.05$	$\alpha = 0.1$
DE/rand/1	16/1/0	136.0	0.0	3.64E-04	yes	yes
DE/rand/2	15/2/0	151.5	1.5	3.24E-04	yes	yes
DE/best/1	11/3/3	106.5	29.5	4.10E-02	yes	yes
DE/best/2	15/2/0	133.0	3.0	6.52E-04	yes	yes
DE/current-to-best/1	11/5/1	141.5	11.5	1.70E-03	yes	yes
DE/rand-to-best/1	10/7/0	146.5	6.5	7.32E-04	yes	yes
JADE	8/7/2	117.5	35.5	4.81E-02	yes	yes
SaDE	12/5/0	131.0	5.0	9.50E-04	yes	yes
ODE	11/6/0	135.0	18.0	4.96E-03	yes	yes
SHADE	9/6/2	128.0	25.0	1.33E-02	yes	yes

directions. By applying SON to DE, the resultant DE framework, namely SON-based DE (SON-DE), was presented for numerical optimization.

To verify the effectiveness of SON-DE, extensive experiments were conducted to compare it with the original DE algorithms, the advanced DE variants, and the state-of-the-art EAs, for solving the test functions from the CEC2013 competition. The experimental results demonstrated that SON-DE can not only bring significant improvements for most DE algorithms but also achieves the competitive results when compared with the state-of-the-art EAs. Moreover, the effectiveness of the main components and the parameter sensitivity of SON-DE were also analyzed. At last, the CEC2017 real-parameter functions and the CEC2012 real-world problems were tested to further show the consistently high performance of SON-DE on different kinds of problems.

In the future, it would be beneficial to introduce the adaptive or self-adaptive techniques for setting parameters of SON. Besides, whether the proposed SON-DE framework is promising for other population-based EAs, is another interesting direction for study.

CRediT authorship contribution statement

Yiqiao Cai: Conceptualization, Methodology, Writing - review & editing. **Duanwei Wu:** Software, Data curation, Writing - original draft. **Ying Zhou:** Validation, Software. **Shunkai Fu:** Visualization, Formal analysis. **Hui Tian:** Supervision. **Yongqian Du:** Project administration.

Acknowledgments

This work was supported in part by the Natural Science Foundation of Fujian Province of China (2018J01091), the National Natural Science Foundation of China (61572204, 61502184), the Natural Science Foundation of Guangdong Province of China (2018A0303130055), the Promotion Program for Young and Middle-aged Teacher in Science and Technology Research of Huaqiao University (ZQN-PY410), and the Opening Project of Guangdong Province Key Laboratory of Computational Science at the Sun Yat-sen University.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.swevo.2020.100699.

Author declaration

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us.

We confirm that we have given due consideration to the protection of intellectual property associated with this work and that there are no impediments to publication, including the timing of publication, with respect to intellectual property. In so doing we confirm that we have followed the regulations of our institutions concerning intellectual property.

We understand that the Corresponding Author is the sole contact for the Editorial process (including Editorial Manager and direct communications with the office). He/she is responsible for communicating with the other authors about progress, submissions of revisions and final approval of proofs. We confirm that we have provided a current, correct email address which is accessible by the Corresponding Author and which has been configured to accept email from (yiqiao00@163.com).

References

[1] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[2] R.D. Al-Dabbagh, F. Neri, N. Idris, M.S. Baba, Algorithmic design issues in adaptive differential evolution schemes: review and taxonomy, Swarm Evol. Comput. 43 (2018) 284–311.
[3] S. Das, S.S. Mullick, P.N. Suganthan, Recent advances in differential evolution–an updated survey, Swarm Evol. Comput. 27 (2016) 1–30.
[4] S. Das, P. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2011) 4–31.
[5] J. Zhang, A. Sanderson, Jade: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (5) (2009) 945–958.
[6] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, 2013, pp. 71–78.
[7] X.-F. Liu, Z.-H. Zhan, Y. Lin, W.-N. Chen, Y.-J. Gong, T.-L. Gu, H.-Q. Yuan, J. Zhang, Historical and heuristic-based adaptive differential evolution, IEEE Trans. Syst. Man Cybern.: Systems (99) (2018) 1–13.
[8] S. Li, Q. Gu, W. Gong, B. Ning, An enhanced adaptive differential evolution algorithm for parameter extraction of photovoltaic models, Energy Convers. Manag. 205 (2020) 112443.
[9] S. Das, A. Abraham, U. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Trans. Evol. Comput. 13 (3) (2009) 526–553.
[10] Y. Cai, J. Wang, Differential evolution with hybrid linkage crossover, Inf. Sci. 320 (320) (2015) 244–287.
[11] A. Ghosh, S. Das, A.K. Das, L. Gao, Reusing the past difference vectors in differential evolution–a simple but significant improvement, IEEE Trans. Cybern. (2019) 1, https://doi.org/10.1109/TCYB.2019.2921602.
[12] L. Deng, H. Sun, L. Zhang, L. Qiao, _CODE η, A differential evolution with η _Cauchy operator for global numerical optimization, IEEE Access 7 (2019) 88517–88533.
[13] A. Qin, V. Huang, P. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2) (2009) 398–417.
[14] G. Wu, R. Mallipeddi, P.N. Suganthan, R. Wang, H. Chen, Differential evolution with multi-population based ensemble of mutation strategies, Inf. Sci. 329 (2016) 329–345.
[15] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (1) (2011) 55–66.
[16] X. Zhou, G. Zhang, Differential evolution with underestimation-based multimutation strategy, IEEE Trans. Cybern. 49 (4) (2019) 1353–1364.
[17] Y. Cai, M. Zhao, J. Liao, T. Wang, H. Tian, Y. Chen, Neighborhood guided differential evolution, Soft Comput. 21 (16) (2017) 4769–4812.
[18] G. Sun, Y. Cai, T. Wang, H. Tian, C. Wang, Y. Chen, Differential evolution with individual-dependent topology adaptation, Inf. Sci. 450 (2018) 1–38.
[19] B. Dorronsoro, P. Bouvry, Improving classical and decentralized differential evolution with new mutation operator and population topologies, IEEE Trans. Evol. Comput. 15 (1) (2011) 67–98.
[20] Y.-F. Ge, W.-J. Yu, Y. Lin, Y.-J. Gong, Z.-H. Zhan, W.-N. Chen, J. Zhang, Distributed differential evolution based on adaptive mergence and split for large-scale optimization, IEEE Trans. Cybern. 48 (7) (2018) 2166–2180.
[21] J. Sun, Q. Zhang, E. Tsang, DE/EDA: a new evolutionary algorithm for global optimization, Inf. Sci. 169 (3) (2005) 249–262.
[22] B. Xin, J. Chen, J. Zhang, Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: a review and taxonomy, IEEE Trans. Syst. Man Cybern. C App Rev. 42 (5) (2012) 744–767.
[23] S.S. Jadon, R. Tiwari, H. Sharma, J.C. Bansal, Hybrid artificial bee colony algorithm with differential evolution, Appl. Soft Comput. 58 (2017) 11–24.
[24] J. Kennedy, Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC), vol. 3, IEEE, 1999, pp. 1931–1938.
[25] I. De Falco, A.D. Cioppa, U. Scafuri, E. Tarantino, Exploiting diversity in an asynchronous migration model for distributed differential evolution, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, ACM, 2017, pp. 1880–1887.
[26] M. Weber, F. Neri, V. Tirronen, Distributed differential evolution with explorative–exploitative population families, Genet. Program. Evolvable Mach. 10 (4) (2009) 343–371.
[27] M. Weber, V. Tirronen, F. Neri, Scale factor inheritance mechanism in distributed differential evolution, Soft Comput. 14 (11) (2010) 1187–1207.
[28] H. Wang, S. Rahnamayan, H. Sun, M.G. Omran, Gaussian bare-bones differential evolution, IEEE Trans. Cybern. 43 (2) (2013) 634–647.
[29] V. Noroozi, A. Hashemi, M. Meybodi, CellularDE: a cellular based differential evolution for dynamic optimization problems, in: Adaptive and Natural Computing Algorithms, Springer, 2011, pp. 340–349.
[30] J. Liao, Y. Cai, T. Wang, H. Tian, Y. Chen, Cellular direction information based differential evolution for numerical optimization: an empirical study, Soft Comput. 20 (7) (2016) 2801–2827.
[31] L. Cui, G. Li, Z. Zhu, Q. Lin, K.-C. Wong, J. Chen, N. Lu, J. Lu, Adaptive multiple-elites-guided composite differential evolution algorithm with a shift mechanism, Inf. Sci. 422 (2018) 122–143.
[32] Y. Sun, Y. Li, G. Liu, J. Liu, A novel differential evolution algorithm with adaptive of population topology, in: International Conference on Information Computing and Applications, Springer, 2012, pp. 531–538.
[33] Y. Cai, G. Sun, T. Wang, H. Tian, Y. Chen, J. Wang, Neighborhood-adaptive differential evolution for global numerical optimization, Appl. Soft Comput. 59 (2017) 659–706.
[34] M. Epitropakis, D. Tasoulis, N. Pavlidis, V. Plagianakos, M. Vrahatis, Enhancing differential evolution utilizing proximity-based mutation operators, IEEE Trans. Evol. Comput. 15 (1) (2011) 99–119.
[35] B.-Y. Qu, P.N. Suganthan, J.-J. Liang, Differential evolution with neighborhood mutation for multimodal optimization, IEEE Trans. Evol. Comput. 16 (5) (2012) 601–614.
[36] W. He, W. Gong, L. Wang, X. Yan, C. Hu, Fuzzy neighborhood-based differential evolution with orientation for nonlinear equation systems, Knowl. Base Syst. 182 (2019) 104796.
[37] Y. Cai, J. Wang, J. Yin, Learning-enhanced differential evolution for numerical optimization, Soft Comput. 16 (2) (2012) 303–330.
[38] G. Liu, Z. Guo, A clustering-based differential evolution with random-based sampling and Gaussian sampling, Neurocomputing 205 (2016) 229–246.
[39] W. Gao, G.G. Yen, S. Liu, A cluster-based differential evolution with self-adaptive strategy for multimodal optimization, IEEE Trans. Cybern. 44 (8) (2014) 1314–1327.
[40] Z. Wang, Z. Zhan, Y. Lin, W. Yu, H. Wang, S. Kwong, J. Zhang, Automatic niching differential evolution with contour prediction approach for multimodal optimization problems, IEEE Trans. Evol. Comput. 24 (1) (2020) 114–128.
[41] W. Gong, Z. Cai, Differential evolution with ranking-based mutation operators, IEEE Trans. Cybern. 43 (6) (2013) 2066–2081.
[42] Y. Cai, J. Liao, T. Wang, Y. Chen, H. Tian, Social learning differential evolution, Inf. Sci. 433–434 (2018) 464–509.
[43] Y. Cai, J. Wang, Differential evolution with neighborhood and direction information for numerical optimization, IEEE Trans. Cybern. 43 (6) (2013) 2202–2215.
[44] L. Deng, L. Zhang, H. Sun, L. Qiao, DSM-DE: A differential evolution with dynamic speciation-based mutation for single-objective optimization, Memet. Comput. (2019) 1–14.
[45] J. Wang, J. Liao, Y. Zhou, Y. Cai, Differential evolution enhanced with multiobjective sorting based mutation operators, IEEE Trans. Cybern. 46 (12) (2014) 2792–2805.
[46] Y. Cai, Y. Chen, T. Wang, H. Tian, Improving differential evolution with a new selection method of parents for mutation, Front. Comput. Sci. 10 (2) (2016) 246–269.
[47] S. Biswas, S. Kundu, S. Das, An improved parent-centric mutation with normalized neighborhoods for inducing niching behavior in differential evolution, IEEE Trans. Cybern. 44 (10) (2014) 1726–1737.
[48] L. Tang, Y. Dong, J. Liu, Differential evolution with an individual-dependent mechanism, IEEE Trans. Evol. Comput. 19 (4) (2015) 560–574.
[49] T. Kohonen, The self-organizing map, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC), vol. 78, 1990, pp. 1464–1480.
[50] J. Liang, B. Qu, P. Suganthan, G. Alfredo, Problem Definitions and Evaluation Criteria for the Cec 2013 Special Session on Real-Parameter Optimization, Tech. Rep., Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Nanyang Technological University, Singapore, 2013.
[51] N.H. Awad, M.Z. Ali, B.Y. Qu, J.J. Liang, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Bound Constrained Real-Parameter Numerical Optimization, Tech. Rep., Nanyang Technological University, Singapore, November 2016.
[52] S. Das, P. Suganthan, Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems, Tech. Rep., Jadavpur Univ., West Bengal, India Nanyang Technological University, Singapore, 2010.
[53] V. Feoktistov, S. Janaqi, Generalization of the strategies in differential evolution. Proceedings of the 18th International Parallel and Distributed Processing Symposium, IEEE, 2004, p. 165.
[54] S.S. Haykin, S.S. Haykin, S.S. Haykin, K. Elektroingenieur, S.S. Haykin, Neural Networks and Learning Machines, vol. 3, Pearson education, Upper Saddle River, 2009.
[55] J. Han, J. Pei, M. Kamber, Data Mining: Concepts and Techniques, Elsevier, 2011.
[56] H. Zhang, A. Zhou, S. Song, Q. Zhang, X.-Z. Gao, J. Zhang, A self-organizing multiobjective evolutionary algorithm, IEEE Trans. Evol. Comput. 20 (5) (2016) 792–806.
[57] D. Wu, Y. Cai, J. Li, W. Luo, Enhanced differential evolution with self-organizing map for numerical optimization, in: Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing, Springer, 2018, pp. 308–318.
[58] S. García, A. Fernández, J. Luengo, F. Herrera, A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability, Soft. Comput. 13 (10) (2009) 959–977.
[59] J. Derrac, S. Garcła, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. 1 (1) (2011) 3–18.
[60] A. Salvador, D. Molina, M. Lozano, F. Herrera Garc, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization, J. Heuristics 15 (6) (2009) 617–644.
[61] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (3) (1999) 264–323.

[62] S. Rahnamayan, H. Tizhoosh, M. Salama, Opposition-based differential evolution, IEEE Trans. Evol. Comput. 12 (1) (2008) 64–79.
[63] I. Loshchilov, CMA-ES with restarts for solving CEC 2013 benchmark problems, in: Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, Ieee, 2013, pp. 369–376.
[64] T. Liao, T. Stuetzle, Benchmark results for a simple hybrid algorithm on the CEC 2013 benchmark set for real-parameter optimization, in: Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, IEEE, 2013, pp. 1938–1944.
[65] J. Rueda, I. Erlich, Hybrid mean-variance mapping optimization for solving the IEEE-CEC 2013 competition problems, in: Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, 2013, pp. 1664–1671.
[66] F. Caraffini, F. Neri, J. Cheng, G. Zhang, L. Picinali, G. Iacca, E. Mininno, Super-fit multicriteria adaptive differential evolution, in: Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, IEEE, 2013, pp. 1678–1685.
[67] S. Biswas, S. Kundu, S. Das, A.V. Vasilakos, Teaching and learning best differential evoltuion with self adaptation for real parameter optimization, in: Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, IEEE, 2013, pp. 1115–1122.
[68] I. Poikolainen, F. Neri, Differential evolution with concurrent fitness based local search, in: Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, IEEE, 2013, pp. 384–391.
[69] J. Tvrdík, R. Poláková, Competitive differential evolution applied to CEC 2013 problems, in: Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, IEEE, 2013, pp. 1651–1657.
[70] A. Zamuda, J. Brest, E. Mezura-Montes, Structured population size reduction differential evolution with multiple mutation strategies on CEC 2013 real parameter optimization, in: Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, IEEE, 2013, pp. 1925–1931.
[71] F. Caraffini, G. Iacca, F. Neri, L. Picinali, E. Mininno, A CMA-ES super-fit scheme for the re-sampled inheritance search, in: Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, 2013, pp. 1123–1130.
[72] Q. Fan, X. Yan, Self-adaptive differential evolution algorithm with discrete mutation control parameters, Expert Syst. Appl. 42 (3) (2015) 1551–1572.
[73] L.M. Zheng, L. Liu, S.X. Zhang, S.Y. Zheng, Enhancing differential evolution with interactive information, Soft Comput. 22 (23) (2018) 7919–7938.
[74] A.W. Mohamed, A.K. Mohamed, Adaptive guided differential evolution algorithm with novel mutation for numerical optimization, Int. J. Mach. Learn. Cybern. 10 (2) (2019) 253–277.
[75] W. Sun, Y. Song, A. Lin, H. Tang, Modified clustering-based differential evolution with a flexible combination of exploration and exploitation, Soft Comput. 22 (18) (2018) 6087–6098.
[76] J. Brest, M.S. Maučec, B. Bošković, Single objective real-parameter optimization: algorithm jSO, in: Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 2017, pp. 1311–1318.