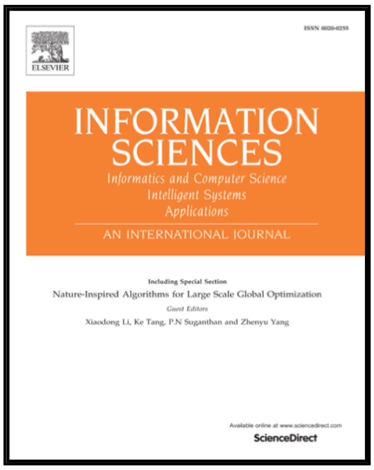
Accepted Manuscript



Differential evolution with neighborhood-based adaptive evolution mechanism for numerical optimization

Mengnan Tian, Xingbao Gao

PII: S0020-0255(18)30909-5
DOI: https://doi.org/10.1016/j.ins.2018.11.021
Reference: INS 14065

To appear in: *Information Sciences*

Received date: 16 April 2018
Revised date: 6 November 2018
Accepted date: 11 November 2018

Please cite this article as: Mengnan Tian, Xingbao Gao, Differential evolution with neighborhood-based adaptive evolution mechanism for numerical optimization, *Information Sciences* (2018), doi: https://doi.org/10.1016/j.ins.2018.11.021

Differential evolution with neighborhood-based adaptive evolution mechanism for numerical optimization

Mengnan Tian, Xingbao Gao*†

Abstract: This paper presents a novel differential evolution algorithm for numerical optimization by designing the neighborhood-based mutation strategy and adaptive evolution mechanism. In the proposed strategy, two novel neighborhood-based mutation operators and an individual-based selection probability are developed to adjust the search performance of each individual suitably. Meanwhile, the evolutionary dilemmas of the neighborhood are identified by tracking its performance and diversity, and alleviated by designing a dynamic neighborhood model and two exchanging operations in the proposed mechanism. Furthermore, the population size is adaptively adjusted by a simple reduction method. Differing from differential evolution variants based on neighborhood and evolutionary state, the proposed algorithm makes full use of the characteristics of individuals, identifies and alleviates the neighborhood evolutionary dilemmas of each individual. Compared with 21 typical algorithms, the numerical results on 30 benchmark functions from CEC2014 show that the proposed algorithm is reliable and has better performance.

Keywords: Differential evolution, dynamic neighborhood, evolutionary state, population reduction, numerical optimization.

1. Introduction

Over the last decades, the global optimization has attracted a great interest of researchers, and many nature-inspired intelligent algorithms have been developed such as genetic algorithm (GA), differential evolution (DE), particle swarm optimization (PSO), artificial bee colony algorithm and tabu search algorithm [8, 13, 19, 35, 45]. Because of the simple idea and facile realization, they have been successfully applied to a variety of engineering contexts including engineering design, signal processing, parameter estimation and pattern

*School of Mathematics and Information Science, Shaanxi Normal University, Xi'an, Shaanxi 710062, China. Email: {mengnan_tian, xinbaog}@snnu.edu.cn.
†Corresponding Author

recognition [7,8,17,30,33,34]. Among them, DE algorithm [35] is proved to be an accurate, reasonably fast and robust optimizer for numerical optimization. However, similar to other stochastic optimization algorithms [13, 19], it is also common and challenging for DE to find the global optimum. In particular, for complicated problems, many local optima are more likely to cause the premature convergence and stagnation [10]. Thus, it is necessary to further improve DE performance.

As pointed out in [10], the performance of DE depends heavily on the appropriate balance between exploration and exploitation. In particular, they access the new regions of search space and those within the neighborhood of previously visited points, respectively. According to diversity measure, maintenance, control and learning, researchers developed many direct and indirect measures to evaluate them such as distance-based measure, external archives, estimation of distribution and so on [6, 22]. Although these methods can adaptively adjust the search capability of algorithm, it is often too difficult for them to distinguish or control the exploration and exploitation. In general, the influences of the evolution strategies and mechanisms on the search process are employed to indirectly measure the exploration and exploitation, *i.e.*, there must be a better balance between them if better results are obtained. Thus, to improve the search quality of DE, many methods have been developed to achieve the balance between exploration and exploitation over the last decades [1–5, 12, 21, 23, 24, 26, 27, 31, 36–38, 40, 41, 43, 44, 46–50]. Among them, the performance of the synthesized algorithms [44,48] are mainly determined by the basic algorithm, and the control parameters settings [1–3,12,26,31,36,37,40,41,50] are closely related to the corresponding strategies or mechanisms. Then they are often difficult for problems at hand. Moreover, the trial vector generation strategies [1, 4, 5, 21, 23, 24, 26, 27, 31, 40, 41, 43, 47, 49] always control the search ability of algorithm directly, and the operations based on evolutionary state [27, 38, 46] could effectively alleviate the evolutionary dilemmas. However, the underlying and useful information among individuals are still not adequately utilized. Therefore, it is necessary and important to design some new strategies and operations to further improve DE performance.

It is well known that the trial vector generation strategy, including mutation and crossover, plays an important role in the search capability of DE. In general, different mutation and crossover operators always have quite different search characteristics and effects. Then a number of methods have been developed to enhance the performance of trial vector generation strategy [1, 4, 5, 21, 23, 24, 26, 27, 31, 40, 43, 47, 49]. Some of them combine several typical strategies with various search characteristics [26, 27, 31, 40, 47], and others properly incorporate the neighborhood topology [1, 4, 5, 21, 23, 24, 43, 49]. Specially, the neighborhood topology is always used to restrict the scope of interaction among individuals such that the search capability can be adjusted effectively. For example, Ali *et al.* [1] divided the population into equal-sized tribes and utilized the mutation strategy with different parameter settings to alleviate the stagnation and premature convergence. Liao *et al.* [21] used cellular topology as the neighborhood topology for each individual and incorporated the direction of information flow into the mutation operation. Cai *et al.* [4] employed the neighborhood guided selection method to choose the parent individuals and introduced the direction information of best/worst nearby neighbor in the mutation process. Meanwhile, Cai *et al.* [5] proposed a DE framework with the concept of index-based neighborhood by extracting the promising search directions from the neighborhood to guide the mutation process. Although these methods make great progress in improving DE performance, the mutation operation in each method always remains unchanged even for different individuals in the same neighborhood, and the characteristic of each individual is not considered in its mutation process. Thus, they cannot adaptively adjust the search performance of each individual. To overcome this shortcoming, it is vital to design some new neighborhood-based adaptive strategies.

Besides, another common way to enhance the search performance is to incorporate the evolutionary state-based operations into the framework of DE. In this way, the evolutionary dilemmas are dealt with by delineating the evolutionary states and designing special operations [27, 38, 46]. Mohamed [27] proposed a restart mechanism to avoid the premature convergence by tracking the performance of individual. Yang *et al.* [46] designed an auto-enhanced population diversity mechanism to resolve the issues of premature convergence and stagnation by measuring the distribution of the population in each dimension. Even though the experimental results show that the operations based on evolutionary state improve the balance between exploration and exploitation, the evolutionary states of the neighborhood are not considered and employed. It should be pointed out that the evolutionary states of the neighborhood might be helpful to improve the search capability and avoid a large number of invalid searches. Thus, it is necessary to develop some new operations by considering the neighborhood evolutionary state.

Based on the above important considerations and motivated by the information of neighborhood being helpful to enhance the performance of the algorithm, this paper presents a novel differential evolution algorithm (NDE) to achieve a proper balance between exploration and exploitation. The main contributions of the paper are as follows.

1) To adjust the search performance of each individual adaptively, we propose a neighborhood-based mutation (NM) strategy by designing two novel mutation operators with different search characteristics based on neighborhood and an individual-based probability parameter to choose a more suitable operator. Differing from the neighborhood-based

DE variants [1,4,5,21,23,24,26,27,31,40,41,43,47,49], NM strategy uses neighborhood information and individual information to design mutation operators and probability parameter, respectively. Then the worse or better individuals can suitably choose an explorative or exploitative mutation operator to search the decision space. Thus, NM strategy could effectively preserve a proper ratio between exploration and exploitation according to the performance of each individual.

2) To identify and relieve the evolutionary dilemmas of neighborhood, we propose a neighborhood-based adaptive evolution (NAE) mechanism by tracking its performance and diversity and presenting a dynamic neighborhood model and two exchanging operations, respectively. The proposed model guides the search to a promising region and helps to jump out of the local optimum by adding new individuals to the neighborhood. Meanwhile, two exchanging operations deals with the premature convergence and stagnation by using the binomial crossover operation to intercross the current individual with one randomly generated from the search space and the best one in the neighborhood, respectively. Unlike the evolutionary state-based DE variants [27,38,46] that always investigate the evolutionary states of the whole population, NAE mechanism employs the performance and diversity of the neighborhood to identify its evolutionary states, and deals with the different evolutionary dilemmas by the dynamic neighborhood model and two exchanging operations. Then NAE mechanism could effectively identify and alleviate the different evolutionary dilemmas of the neighborhood to adjust the search capability and improve the search efficiency.

3) A simple reduction method is employed to adaptively adjust the population size such that the diversity and exploitation capability can be maintained and enhanced at the earlier and later evolutionary processes, respectively.

Therefore, the proposed algorithm could not only adjust suitably the search performance of each individual, but also maintain a proper balance between exploration and exploitation. Finally, numerical experiments are carried out to evaluate the performance of NDE by comparing it with 21 typical algorithms on 30 benchmark functions from CEC2014 [20]. Meanwhile, NDE is also applied to Parameter Estimation for Frequency-Modulated Sound Waves. Experimental results show that the proposed algorithm is very competitive.

The reminder of this paper is organized as follows. In Section 2, the classical DE algorithm is briefly introduced. A novel differential evolution with NAE mechanism is proposed in Section 3. The experimental results of the proposed algorithm are reported and discussed in Section 4. Finally, conclusions are drawn in Section 5.

2. Classical DE algorithm

The basic DE includes initialization, mutation, crossover and selection. Specially, consider the minimization problem $\min\{f(\vec{x})|x_j^{min} \leq x_j \leq x_j^{max} \text{ for } j = 1, 2, \cdots, D\}$, where $\vec{x} = (x_1, x_2, \cdots, x_D)$ represents the solution vector, D is the dimension of the solution space, x_j^{min} and x_j^{max} are the lower and upper bounds of the j-th component of solution space, respectively. At the beginning of DE algorithm, initial population $P^0 = \{\vec{x}_i^0 = (x_{i,1}^0, x_{i,2}^0, \cdots, x_{i,D}^0)|i = 1, 2, \cdots, NP\}$ is randomly generated by

$$x_{i,j}^0 = x_j^{min} + rand(0,1) \cdot (x_j^{max} - x_j^{min}), \quad (1)$$

where $x_{i,j}^0$ is the j-th component of the i-th vector $\vec{x}_i^0$, NP is the population size and $rand(0,1) \in [0,1]$ is a uniform random number. Then the mutation, crossover and selection operators will be executed in turn until the termination criterion is met.

At each generation g, the mutation operation is applied to each individual $\vec{x}_i^g$ to generate its mutant individual $\vec{v}_i^g$. In particular, the operator "DE/rand/1"

$$\vec{v}_i^g = \vec{x}_{r1}^g + F \cdot (\vec{x}_{r2}^g - \vec{x}_{r3}^g) \quad (2)$$

is only used in this paper, where F is a scaling factor, the indices r_1, r_2 and r_3 are the distinct integers randomly generated from $[1, NP]$ and not equal to i. Then the crossover operation is performed for $\vec{x}_i^g$ and $\vec{v}_i^g$ to generate its offspring $\vec{u}_i^g$. Specially, the binomial crossover operator [10] can be described as follows:

$$u_{i,j}^g = \begin{cases} v_{i,j}^g, & \text{if } rand \leq Cr \text{ or } j = randn(i), \\ x_{i,j}^g, & \text{otherwise,} \end{cases} \quad (3)$$

where $Cr \in [0,1]$ is the crossover rate, and $randn(i)$ is an integer randomly generated from the range $[1, NP]$ to ensure that $\vec{u}_i^g$ has at least one component from $\vec{v}_i^g$. Finally, the following selection operation [10] is executed to decide whether $\vec{x}_i^g$ or $\vec{u}_i^g$ can survive in the next generation

$$\vec{x}_i^{g+1} = \begin{cases} \vec{u}_i^g, & \text{if } f(\vec{u}_i^g) \leq f(\vec{x}_i^g), \\ \vec{x}_i^g, & \text{otherwise.} \end{cases} \quad (4)$$

Note that DE with (4) will get better or remain the same fitness, but never deteriorate. The detail procedure of the classical DE can be found in [35].

3. Proposed algorithm

Even though the classical DE algorithm is simple and strongly robust, it is often difficult to deal with some practical or complicated problems. Then various DE variants have achieved to strengthen its performance and great progress has been made as mentioned in Section 1, yet there are still several shortcomings. For example, DE variants with neighborhood information rarely use the characteristics of individuals in the same neighborhood during mutation [1, 4, 5, 21, 23, 24, 43, 49]. The variants based on evolutionary state might not be suitable for adjusting the search capability of algorithm for complex problems since they only focus on the evolutionary states of the whole population [27, 38, 46]. To overcome these drawbacks, we shall propose a novel DE variant with adaptive evolution mechanism based on neighborhood in this section. Specially, we design two novel NM operators with different search characteristics and choose a suitable one for each individual according to its characteristic. Meanwhile, the proposed algorithm identifies the evolutionary states of neighborhood by tracking its fitness value and diversity, and relieves the different evolutionary dilemmas by presenting three operations.

For the convenience of the later discussions, let $N(i)$ denote the neighborhood of $\vec{x}_i^g$, N_{size_i} and N_{rsize_i} denote the size and radius of $N(i)$ respectively, $\vec{x}_{nbest_i}^g$ denote the best individual among $N(i)$, fit_{nworst_i}, fit_{nbest_i} and fit_{naver_i} denote the worst, best and average fitness values among $N(i)$ respectively, $Numg_i$ and $Nums_i$ denote the number of the successive unsuccessful update of $\vec{x}_{nbest_i}^g$ and fit_{naver_i} respectively, Std_{nf_i} denote the standard deviation of the fitness values of individuals in $N(i)$ and Std_{nfaver} denote the average value of Std_{nf_i} for all individuals.

3.1. NM strategy

As pointed out in [24], population topology is helpful to balance the exploration and exploitation by controlling the scope of interaction between particles and affecting the dissemination of search information. However, the existing neighborhood-based DE variants [4, 5, 21] do not consider the characteristics of individuals within the same neighborhood, and always use the unchanged mutation strategy such that the search performance of each individual cannot be adaptively adjusted. Thus, to alleviate this shortcoming, we propose the following NM strategy by designing two novel NM operators and an individual-based probability parameter:

$$\vec{v}_i^g = \begin{cases} \vec{x}_{nr_1}^g + F(\vec{x}_{r_1}^g - \vec{x}_{r_2}^g), & \text{if } rand(0,1) < \xi_{1,i}, \\ \vec{x}_i^g + F(\vec{x}_{nbest}^g - \vec{x}_i^g) + F(\vec{x}_{nr_1}^g - \vec{x}_{nr_2}^g) + F(\vec{x}_{r_1}^g - \vec{x}_{r_2}^g), & \text{otherwise}, \end{cases} \quad (5)$$

where F is a scaling factor, r_1 and $r_2 \in [1, NP]$ are two random integers and not equal to i, the neighborhood $N(i)$ of the i-th individual $\vec{x}_i^g$ is constructed by ring topology [24], nr_1 and nr_2 are two random integers from $N(i)$ and not equal to i, $\xi_{1,i}$ is a probability parameter based on the performance of $\vec{x}_i^g$.

Obviously, the first strategy in Eq. (5) takes the individual randomly chosen from the neighborhood $N(i)$ as the base individual and searches around it. But another one uses the current individual as the base individual and searches the search space along the best individual in its corresponding neighborhood. Meanwhile, a difference vector from the whole population is employed to enhance their global search capability. Then they can make full use of the neighborhood and whole population information, and the former has stronger exploration ability than the latter. Thus, NM strategy could effectively improve the balance between exploration and exploitation by choosing a suitable strategy based on a probability for each individual.

From Eq. (5), the probability parameter $\xi_{1,i}$ plays an important role in its performance since an unreasonable setting will lead to explore or exploit ineffectively the information of each individual. To choose a suitable mutation operator for each individual and make full use of its characteristic, let

$$\xi_{1,i} = (1 + \exp(20\frac{fit_{naver_i} - fit(i)}{fit_{nworst_i} - fit_{nbest_i}}))^{-1}, \tag{6}$$

where $fit(i)$ is the fitness value of $\vec{x}_i^g$, and

$$fit_{naver_i} = \frac{1}{N_{size_i}} \sum_{k \in N(i)} fit(k) \tag{7}$$

with N_{size_i} being the size of $N(i)$. From Eqs. (6) and (7), $\xi_{1,i}$ becomes smaller or larger if $\vec{x}_i^g$ has better or worse fitness. Then the individual with worse or better performance has more chances to employ the mutation operator with more explorative or exploitative in Eq. (5). Thus, the proposed strategy can adaptively adjust the search performance of each individual.

In summary, the proposed strategy in Eq. (5) develops two novel NM operators with different search characteristics, and an individual-based probability parameter to choose a suitable one for each individual. Unlike the methods [4, 5, 21] that do not consider the differences between individuals in the same neighborhood, NM strategy searches the broader region or the more promising position around the worse or better individual. Thus, it could not only make full use of the neighborhood information, but also adaptively adjust the search performance for each individual. Therefore, the proposed strategy effectively adjusts the exploration and exploitation, which is shown by the experiments in Subsection 4.2.1.

3.2. NAE mechanism

The existing neighborhood models [4, 5, 21, 24] are always fixed, and their evolutionary states are not identified and employed to improve the algorithm performance. Then they will waste a great number of computational resources whenever the neighborhood is in an evolutionary dilemma, and cannot properly adjust the search capability of each individual, especially for complicated problems. To identify and overcome the evolutionary dilemmas of neighborhood effectively, we propose a NAE mechanism by using the performance and diversity of the neighborhood and designing a dynamic neighborhood model and two exchanging operations in the following.

In the proposed mechanism, the neighborhood evolutionary state is characterized by its performance and diversity. To evaluate the performance of the neighborhood of $\vec{x}_i^g$, we employ two counters, $Numg_i$ and $Nums_i$, as the indicators to record the number of the successive unsuccessful update of $\vec{x}_{nbest_i}^g$ and the number of the unsuccessful update of fit_{naver_i} during $Numg_i$ iterations, respectively. Set them to 0 at the beginning, increase by 1 when the best individual $\vec{x}_{nbest_i}^g$ and the average fitness value fit_{naver_i} of $N(i)$ are not improved respectively, and return to 0 when a better $\vec{x}_{nbest_i}^g$ is obtained. On the other hand, the diversity of the neighborhood is characterized by the standard deviation (Std_{nf_i}) of the fitness values of the individuals in $N(i)$. In general, a larger or smaller Std_{nf_i} means that the individuals in $N(i)$ are relatively scattered or crowded. Then the neighborhood with smaller or larger Std_{nf_i} is more likely to suffer from the premature convergence or stagnation whenever no individual is updated after several successive generations. Clearly, it requires less computational costs to evaluate the diversity of neighborhood in the objective space than that in the search space.

According to the counters $Numg_i$ and $Nums_i$, the following two evolutionary dilemmas of the neighborhood might be encountered when $Numg_i$ meets a prescribed limited value gm.

(i) The ratio $Nums_i/Numg_i$ is close to 0, *i.e.*, fit_{naver_i} is not improved within few iterations during $Numg_i$ iterations. This might be due to the fact that the best individual in the neighborhood might be located at the local optimum, but the other individuals do not converge to it. Then it is useless to further search in the current neighborhood, and the neighborhood topology should be reconstructed to guide the individuals toward a more promising region. To do this, we develop the following dynamic neighborhood model to enlarge the neighborhood $N(i)$ of $\vec{x}_i^g$ by adding new individuals.

$$N_{rsize_i} = N_{rsize_i} + 1, \tag{8}$$

where $N_{rsize_i} = (N_{size_i} - 1)/2$ is the radius of $N(i)$. At the beginning, let N_{rsize_i} be 1 to ensure the exploration of the algorithm in the early evolutionary stage. Furthermore, to ensure the rationality of N_{rsize_i}, let

$$N_{rsize_i} = \min(N_{rsize_i}, floor(0.5 \cdot (NP - 1))), \tag{9}$$

where $\min(a, b)$ returns the minimum one between a and b, and $floor(c)$ is the nearest integer smaller than c. Clearly, N_{size_i} is increased and $\vec{x}_i^g$ searches within a more promising region when the dilemma occurs. Thus, the proposed model could help to jump out of local optimum, and effectively adjust the search performance of $\vec{x}_i^g$.

(ii) The ratio $Nums_i/Numg_i$ is close to 1, *i.e.*, there is almost no progress on fit_{naver_i} during $Numg_i$ iterations, which may be due to the premature convergence or stagnation. According to [46], the evolutionary state of $N(i)$ shall be regarded as the premature convergence or stagnation when Std_{nf_i} is smaller or larger than the average diversity Std_{nfaver} of all neighborhoods. In general, they can be alleviated by enhancing the diversity of neighborhood and making full use of the information of the promising individuals, respectively. To do this, we design the following two exchanging operations.

Regenerate $\vec{x}_i^g$ as

$$\vec{x'}_i^g = \begin{cases} \vec{x'}_{I,i}^g, & \text{if } Std_{nf_i} < Std_{nfaver}, \\ \vec{x'}_{B,i}^g, & \text{otherwise}, \end{cases} \tag{10}$$

where $\vec{I} = \{I_1, I_2, \cdots, I_D\}$ with $I_j = x_j^{min} + rand(0,1) \cdot (x_j^{max} - x_j^{min})$ for $j = 1, 2, \cdots, D$, $\vec{x'}_{I,i}^g = (x'^g_{I,i,1}, x'^g_{I,i,2}, \cdots, x'^g_{I,i,D})$ and $\vec{x'}_{B,i}^g = (x'^g_{B,i,1}, x'^g_{B,i,2}, \cdots, x'^g_{B,i,D})$ are generated by

$$x'^g_{I,i,j} = \begin{cases} I_j, & \text{if } rand(0,1) < \xi_{2,i}, \\ x^g_{i,j}, & \text{otherwise} \end{cases} \tag{11}$$

and

$$x'^g_{B,i,j} = \begin{cases} x^g_{nbest_i,j}, & \text{if } rand(0,1) < \xi_{2,i}, \\ x^g_{i,j}, & \text{otherwise} \end{cases} \tag{12}$$

for $j = 1, 2, \cdots, D$ respectively, $\xi_{2,i}$ is the crossover parameter.

To make full use of the information of $\vec{x}_i^g$ and ensure the convergence of algorithm during the later evolutionary process, the possibility of intercrossing $\vec{x}_i^g$ with $\vec{x}_{nbest_i}^g$ or $\vec{I}$ should be smaller as the iteration proceeds or it has better performance. Then, let

$$\xi_{2,i} = 1 - \min\left(\frac{FES}{FES_{max}}, \frac{fit_{max} - fit(i)}{fit_{max} - fit_{min}}\right), \tag{13}$$

where FES and FES_{max} are the current and maximum number of fitness evaluations respectively, $fit(i)$, fit_{max} and fit_{min} are the fitness values of $\vec{x}_i^g$, the worst and best individuals among the whole population, respectively. From Eqs. (10)-(13), the diversity or the promising information of the neighborhood $N(i)$ can be enhanced or exploited by exchanging $\vec{x}_i^g$ with $\vec{I}$ or $\vec{x}_{nbest_i}^g$. Thus, these proposed operations could effectively alleviate the premature convergence and stagnation.

Obviously, the neighborhood is more likely to fall into the local optimum, or suffer from the premature convergence and stagnation when $Numg_i$ exceeds gm. Then the parameter gm plays an important role in the identification of evolutionary states, and should not be too large for simple functions, and not too small or too large for the complicated problems. In fact, for simple problems, a small gm will lead to a rapid increase of the size of neighborhood so that the promising information can be exploited to improve convergence. For complicated problems, a too small gm could cause a premature judgement of dilemmas on the evolutionary states such that some promising information in the current neighborhood cannot be fully utilized. Meanwhile, a too large gm will waste a large amount of computational resources due to the ineffective searches after the neighborhood is truly in the evolutionary dilemmas. Thus, let $gm = 10$ from the sensitivity analysis in Subsection 4.1.

From the above discussions, the proposed mechanism identifies the evolutionary states of the neighborhood by using its fitness value and diversity, and deals with its different evolutionary dilemmas by developing a dynamic neighborhood model and two exchanging operations. In particular, when $Numg_i$ exceeds gm and $Nums_i/Numg_i$ approaches 0, new individuals are added in the current neighborhood to enhance its diversity. This is helpful to jump out of local optimum and guide the search toward a more promising region. On the other hand, when $Nums_i/Numg_i$ approaches 1, the current individual $\vec{x}_i^g$ is exchanged with $\vec{I}$ or $\vec{x}_{nbest_i}^g$ to enhance the diversity or utilize the promising information of better individuals. Meanwhile, the exchanging probability becomes smaller as the iteration proceeds, or when $\vec{x}_i^g$ has better performance. Unlike the DE variants [4, 5, 21], the proposed mechanism can identify neighborhood dilemmas, and alleviate them by enhancing its diversity and making full use of promising information. Therefore, the proposed mechanism effectively adjusts the search performance of each individual and improves the search efficiency. Furthermore, its effectiveness is illustrated by experiments in Subsection 4.2.2.

3.3. Parameter setting

It is well known that the control parameters, including scaling factor F, crossover rate Cr and populations size NP, also influence the search capability of algorithm mainly, and appropriate parameter settings can enhance its performance [1–3, 10, 35, 37]. In particular, the constant method in [35] improves the running efficiency of DE algorithm, yet it always takes more time to tune and is unsuitable for all problems. The random method [10] can enhance the robustness, but it could not adapt to the different evolutionary processes. Unlike the constant and random methods [10, 35], the adaptive methods [2, 37] can dynamically adjust parameters and effectively balance the exploration and exploitation. To make full use of feedback information, we set F and Cr by employing the weighted adaptive method [37] as follows.

For the individual $\vec{x}_i^g$, its corresponding scale factor

$$F_i^g = rand_C(F_{loc}^g, 0.1), \quad i = 1, 2, \cdots, NP, \tag{14}$$

where $rand_C(F_{loc}^g, 0.1)$ is the cauchy distribution with location parameter

$$F_{loc}^g = (1-c) \cdot F_{loc}^{g-1} + c \cdot mean_{WL}(S_F^{g-1}), \tag{15}$$

$c \in (0, 1]$ is a constant, S_F^{g-1} is the set of successful F values at $g-1$ generation,

$$mean_{WL}(S_F^{g-1}) = \frac{\sum_{k=1}^{|S_F^{g-1}|} w_k \cdot F_k^2}{\sum_{k=1}^{|S_F^{g-1}|} w_k \cdot F_k}, \tag{16}$$

$$w_k = \frac{\triangle f_k}{\sum_{k=1}^{|S_F^{g-1}|} \triangle f_k} \tag{17}$$

and $\triangle f_k = |f(\vec{u}_k^{g-1}) - f(\vec{x}_k^{g-1})|$. Similarly, the corresponding crossover rate is set as

$$Cr_i^g = rand_n(Cr_{mean}^g, 0.1), \quad i = 1, 2, \cdots, NP, \tag{18}$$

where $rand_n(Cr_{mean}^g, 0.1)$ is the normal distribution with standard deviation 0.1 and mean

$$Cr_{mean}^g = (1-c) \cdot Cr_{mean}^{g-1} + c \cdot mean_{WA}(S_{Cr}^{g-1}), \tag{19}$$

S_{Cr}^{g-1} is the set of all successful Cr values at $g-1$ generation,

$$mean_{WA}(S_{Cr}^{g-1}) = \sum_{k=1}^{|S_{Cr}^{g-1}|} w_k \cdot Cr_k \tag{20}$$

and w_k is defined in (17). To ensure the validity of F_i^g and Cr_i^g, let F_i^g be truncated to 1 if $F_i^g > 1$ and be regenerated by (14) if $F_i^g < 0$, and

$$Cr_i^g = \begin{cases} 0, & \text{if } Cr_i^g < 0, \\ 1, & \text{if } Cr_i^g > 1. \end{cases} \tag{21}$$

Similar to [37], c is set to 0.1, F_{loc} and Cr_{mean} are initialized to 0.5.

Moreover, as pointed out in [1, 3, 37], population size reduction can effectively improve the performance of algorithm. To further enhance the performance of the proposed method, we employ a reduction method [37] to adjust dynamically the population size. In particular, the current population size NP is first calculated by

$$NP = round[(\frac{NP^{min} - NP^{ini}}{FES_{max}}) \cdot FES + NP^{ini}], \tag{22}$$

where $round(a)$ is the nearest integer around a, NP^{min} and NP^{ini} are the smallest and initial size of population, respectively. Then we delete the individual with the worst fitness value when the population size is reduced. From Eq. (22), a too large or too small NP^{ini} could cause a large amount of invalid searches during the earlier evolutionary process or weaken the global search ability. Thus, let $NP^{ini} = 10D$, which is a suitable choice by experiments in Subsection 4.1. In addition, set NP^{min} to 5 since Eq. (5) requires at least five individuals. Clearly, the population size is gradually reduced and the better individuals are retained as the number of iterations increases. Therefore, it is helpful to enhance the exploitation at the later evolutionary stage, and the above parameter settings could adaptively adjust the search capability and balance the exploration and exploitation effectively.

In summary, a novel DE variant (NDE) can be proposed and described in Algorithm 1 by integrating NM strategy, NAE mechanism and the parameter adaptation method in this subsection.

From Algorithm 1, one can see that for each target individual $\vec{x}_i^g$, a suitable NM operator is chosen to generate its mutant individual according to the individual-based probability $\xi_{i,1}$ (lines 9-15 in Algorithm 1). After each generation, the neighborhood evolutionary state of each individual is identified by tracking the performance and diversity of its corresponding neighborhood (lines 26-36). When the evolutionary dilemmas occur, they are alleviated by a dynamic neighborhood model and two exchanging operations, respectively (lines 38-52). Finally, the linear reduction method is further applied to delete the worst individual from the current population as the number of iterations increases (lines 53-56). Thus, the proposed algorithm could not only take full advantage of the neighborhood information and the characteristic of each individual, but also effectively adjust the search capability of the population.

It should be mentioned that the DE variant [4] employs a probability to produce neighbors for each individual and selects the best individual from them as the base vector to accelerate convergence. However, it might not exploit the promising information around the true neighborhood and does not consider the differences between individuals in the mutation process. On the contrary, for each individual, the proposed NDE employs the index-based ring topology to construct the neighborhood, and chooses a more suitable mutation operator by developing two novel NM operators with different search capabilities. Meanwhile, the PSO variant [28] uses the historical information of neighborhood to update the learner particle, and dynamically produces the neighborhood after a certain interval, which might not be suitable for the evolutionary process. Unlike this PSO variant, the proposed NDE adaptively adjusts the neighborhood of each individual to alleviate

Algorithm 1 (The framework of NDE)

Input: the initial and minimum size of population NP^{ini} and NP^{min}, the maximum number of fitness evaluations FES_{max}, the initial location parameter F^0_{loc}, the initial average crossover rate Cr^0_{mean}, the weighted parameter c and the limit parameter gm.
Set population size $NP = NP^{ini}$, the current generation $g = 0$; initialize the population $P^g = \{\vec{x}^g_1, \vec{x}^g_2, \cdots, \vec{x}^g_{NP}\}$ and evaluate its fitness; fitness evaluation counter $FES = NP$; initialize neighborhood radius $N_{rsize_i} = 1$, $Numg_i = 0$ and $Nums_i = 0$ for $\vec{x}^g_i$ with $i = 1, 2, \cdots, NP$;
while $FES \leq FES_{max}$ **do**
  $S_F = \emptyset$ and $S_{Cr} = \emptyset$;
  for $i = 1 : NP$ **do**
    Construct $N(i)$ based on ring topology structure, and calculate fit_{nbest_i}, fit_{nworst_i}, fit_{naver_i} and Std_{nf_i};
    Let $oldfit_{nbest_i} = fit_{nbest_i}$, $oldfit_{nworst_i} = fit_{nworst_i}$, $oldfit_{naver_i} = fit_{naver_i}$ and $oldStd_{nf_i} = Std_{nf_i}$;
    Calculate F^g_i by Eq. (14), and correct it; Calculate Cr^g_i by Eqs. (18) and (21), and $\xi_{1,i}$ by Eqs. (6) and (7);
    if $rand \leq \xi_{i,1}$ **then**
      Randomly select $\vec{x}^g_{nr_1}$ from $N(i)$, $\vec{x}^g_{r_1}$ and $\vec{x}^g_{r_2}$ from P^g with $nr_1 \neq r_1 \neq r_2 \neq i$;
      $\vec{v}^g_i = \vec{x}^g_{nr_1} + F^g_i \cdot (\vec{x}^g_{r_1} - \vec{x}^g_{r_2})$;
    else
      Randomly select $\vec{x}^g_{nr_1}$ and $\vec{x}^g_{nr_2}$ from $N(i)$, $\vec{x}^g_{r_1}$ and $\vec{x}^g_{r_2}$ from P^g with $nr_1 \neq nr_2 \neq r_1 \neq r_2 \neq i$;
      $\vec{v}^g_i = \vec{x}^g_i + F^g_i \cdot (\vec{x}^g_{nbest} - \vec{x}^g_i) + F^g_i \cdot (\vec{x}^g_{nr_1} - \vec{x}^g_{nr_2}) + F(\vec{x}^g_{r_1} - \vec{x}^g_{r_2})$;
    end if
    Execute the crossover operation for $\vec{x}^g_i$ and $\vec{v}^g_i$ to generate its offspring $\vec{u}^g_i$ by Eq.(3);
    Evaluate $\vec{u}^g_i$; $FES = FES + 1$;
    if $f(\vec{u}^g_i) \leq f(\vec{x}^g_i)$ **then**
      $\vec{x}^{g+1}_i = \vec{u}^g_i$; $F^g_i \to S_F$ and $Cr^g_i \to S_{Cr}$;
    else
      $\vec{x}^{g+1}_i = \vec{u}^g_i$;
    end if
  end for
  Calculate $mean_{WL}(S_F)$ and $mean_{WA}(S_F)$ by Eqs. (16), (17) and (20);
  Update F^{g+1}_{loc} and Cr^{g+1}_{mean} by Eqs. (15) and (19), respectively;
  for $i = 1 : NP$ **do**
    Construct $N(i)$ based on ring topology structure, and calculate fit_{nbest_i}, fit_{nworst_i}, fit_{naver_i} and Std_{nf_i};
    if $fit_{nbest_i} < oldfit_{nbest_i}$ **then**
      $Numg_i = 0$; $Nums_i = 0$;
    else
      $Numg_i = Numg_i + 1$;
      if $fit_{naver_i} \geq oldfit_{naver_i}$ **then**
        $Nums_i = Nums_i + 1$;
      end if
    end if
  end for
  Calculate $Std_{nfaver} = \sum_{i=1}^{NP} Std_{nf_i}/NP$;
  for $i = 1 : NP$ **do**
    if $Numg_i = gm$ **then**
      if $rand > Nums_i/Numg_i$ **then**
        $N_{rsize_i} = N_{rsize_i} + 1$; $N_{rsize_i} = \min(N_{rsize_i}, floor(0.5 * (NP - 1)))$;
      else
        if $Std_{nf_i} < Std_{nfaver}$ **then**
          Generate $\vec{x'}^g_i$ by exchanging $\vec{x}^g_i$ with $\vec{I}$ by Eqs.(11) and (13);
        else
          Generate $\vec{x'}^g_i$ by exchanging $\vec{x}^g_i$ with $\vec{x}^g_{nbest_i}$ by Eqs.(12) and (13);
        end if
        $\vec{x}^g_i = \vec{x'}^g_i$; Evaluate $\vec{x}^g_i$; $FES = FES + 1$;
      end if
      $Numg_i = 0$; $Nums_i = 0$;
    end if
  end for
  $NP_{new} = round[(\frac{NP^{min} - NP^{ini}}{FES_{max}}) \cdot FES + NP^{ini}]$;
  if $NP_{new} < NP$ **then**
    Delete the worst $NP - NP_{new}$ individuals from P^g based on the fitness and their corresponding records;
    $NP = NP_{new}$;
  end if
  $g = g + 1$;
end while
Output: The best individual and its fitness value.

the evolutionary dilemmas by designing a dynamic neighborhood model and two exchanging operations according to its evolutionary state. Moreover, the proposed NDE adopts a linear reduction method to adaptively reduce the population size with the increase of iterations, while each population size in [4] and [28] is fixed. Therefore, NDE has more promising performance to adjust the search capabilities of different individuals and adapt to the different evolutionary stages.

3.4. Complexity analysis

In this subsection, we shall analyze the complexity of NDE, which is a very important criterion for evaluating the performance of an algorithm. Obviously, the main differences between NDE and the classical DE algorithm are NM strategy, NAE mechanism and the parameter setting method.

As discussed in the above subsections, the main operations of NM strategy and NAE mechanism are to sort the neighbors of each individual and calculate the diversity of all neighborhoods based on fitness values, respectively. Similar to [4,21,28], their complexities are $O(G \cdot (NP^{ini})^2 \cdot \log_2 NP^{ini})$ and $O(G \cdot (NP^{ini})^3)$ respectively, where G is the maximum number of iterations. According to [10,37], the complexities of the classical DE algorithm and the parameter setting method are $O(G \cdot NP^{ini} \cdot D)$ and $O(NP^{ini} \cdot (2 \cdot G + NP^{ini} - NP^{min}) + 2 \cdot G \cdot NP^{min})$, respectively. Thus, the complexity of NDE is $O(G \cdot NP^{ini} \cdot (D + 2) + NP^{min} \cdot (2 \cdot G - NP^{ini}) + (NP^{ini})^2 \cdot (G \cdot (\log_2 NP^{ini} + NP^{ini}) + 1)$.

It should be pointed out that the diversity of all neighborhoods does not require to be calculated at each generation, and the population size is gradually reduced as the iteration proceeds. Therefore, the complexity of NDE is more expensive, but not severe, than that of the classical DE algorithm.

4. Numerical experiments

In this section, we shall evaluate the performance of NDE by numerical experiments on 30 well-known benchmark functions f_1-f_{30} from CEC 2014 [20] as listed in Table 1, where search range and bias value for each function are also provided. Meanwhile, we will also analyze the sensitivities of parameters in NDE, illustrate the effectiveness of NM strategy and NAE mechanism. Finally, we shall compare NDE with the classical DE, 14 variants of DE and 6 non-DE algorithms, discuss the reliability and efficiency of NDE, and give an application. All experiments are conducted in MATLAB R2014a on a PC (Intel i3-4570 CUP 3.20GHz. RAM 4.00 GB).

In all experiments, the stopping criterion is that the number of function evaluations

Table 1: The benchmark functions of CEC2014

Type	Name	Search range	$f(\vec{x}^*)$ (f_bias)
Unimodal functions	f_1: Rotated high conditioned elliptic function	$[-100, 100]^D$	100
	f_2: Rotated bent cigar function	$[-100, 100]^D$	200
	f_3: Rotated discus function	$[-100, 100]^D$	300
Simple multimodal functions	f_4: Shifted and rotated rosenbrock's function	$[-100, 100]^D$	400
	f_5: Shifted and rotated ackley's function	$[-100, 100]^D$	500
	f_6: Shifted and rotated weierstrass function	$[-100, 100]^D$	600
	f_7: Shifted and rotated griewank's function	$[-100, 100]^D$	700
	f_8: Shifted rastrigin's function	$[-100, 100]^D$	800
	f_9: Shifted and rotated rastrigin's function	$[-100, 100]^D$	900
	f_{10}: Shifted schwefel's function	$[-100, 100]^D$	1000
	f_{11}: Shifted and rotated schwefel's function	$[-100, 100]^D$	1100
	f_{12}: Shifted and rotated katsuura function	$[-100, 100]^D$	1200
	f_{13}: Shifted and rotated happycat function	$[-100, 100]^D$	1300
	f_{14}: Shifted and rotated hgbat function	$[-100, 100]^D$	1400
	f_{15}: Shifted and rotated expanded griewank's plus rosenbrock's function	$[-100, 100]^D$	1500
	f_{16}: Shifted and rotated expanded scaffer's function	$[-100, 100]^D$	1600
Hybrid functions	f_{17}: Hybrid function 1 (N=3)	$[-100, 100]^D$	1700
	f_{18}: Hybrid function 2 (N=3)	$[-100, 100]^D$	1800
	f_{19}: Hybrid function 3 (N=4)	$[-100, 100]^D$	1900
	f_{20}: Hybrid function 4 (N=4)	$[-100, 100]^D$	2000
	f_{21}: Hybrid function 5 (N=5)	$[-100, 100]^D$	2100
	f_{22}: Hybrid function 6 (N=5)	$[-100, 100]^D$	2200
Composition functions	f_{23}: Composition function 1 (N=5)	$[-100, 100]^D$	2300
	f_{24}: Composition function 2 (N=3)	$[-100, 100]^D$	2400
	f_{25}: Composition function 3 (N=3)	$[-100, 100]^D$	2500
	f_{26}: Composition function 4 (N=5)	$[-100, 100]^D$	2600
	f_{27}: Composition function 5 (N=5)	$[-100, 100]^D$	2700
	f_{28}: Composition function 6 (N=5)	$[-100, 100]^D$	2800
	f_{29}: Composition function 7 (N=3)	$[-100, 100]^D$	2900
	f_{30}: Composition function 8 (N=3)	$[-100, 100]^D$	3000

is less than the maximum number of function evaluations (FES_{max}), and set $FES_{max} = 10000D$ for all algorithms in Subsections 4.1-4.4. All algorithms are run 30 times independently except for NDE in Subsection 4.3.4. The average value (Mean Error) and standard deviation (Std Dev) of the function errors $f(\vec{x}) - f(\vec{x}^*)$ are recorded to measure the performance of algorithm, where $\vec{x}$ and $\vec{x}^*$ are the best solution found by the algorithm in a run and the global optimum of test function, respectively. To have statistically sound conclusions, we adopt a) Wilcoxon rank sum test [42] at 0.05 significance level to show the difference between two algorithms on a single problem; b) the multiproblem Wilcoxon signed-rank test [11] at 0.05 significance level to identify the differences between a pair of algorithms; and c) the Friedman test [11] to show overall rankings of all algorithms according to their performances on all problems.

4.1. The sensitivities of parameters gm and NP^{ini}

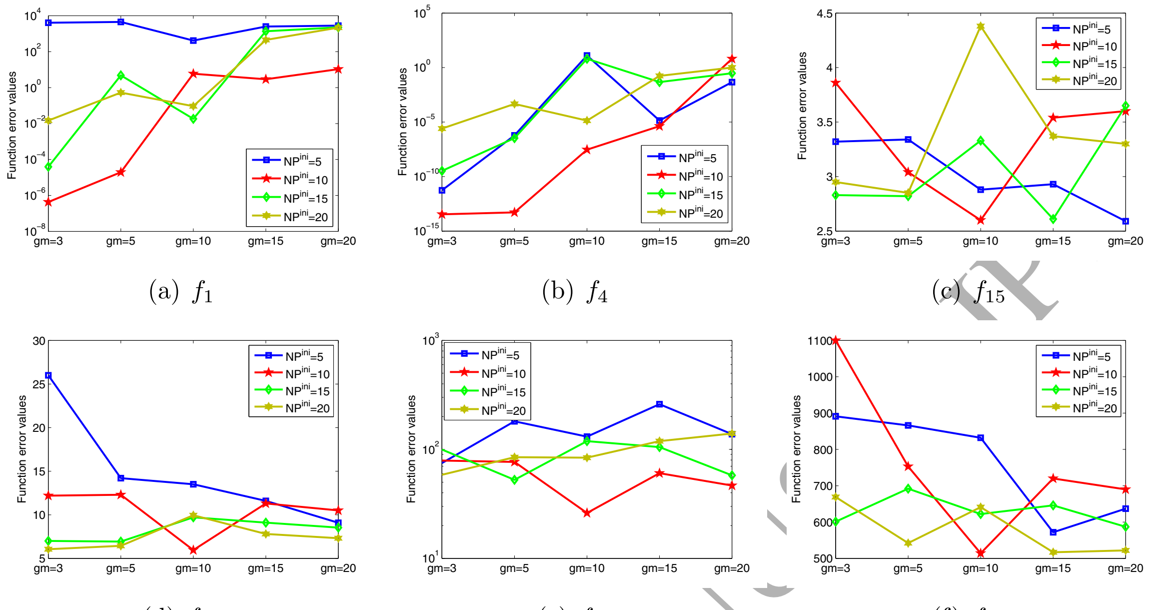
Now, we study the sensitivities and interactions between the prescribed limited value gm and initial population size NP^{ini} in NDE on 6 typical functions f_1, f_4, f_{15}, f_{18}, f_{22} and f_{30} in Table 1. Among many sensitivity analysis methods [16, 18, 29, 32, 39], the full factorial design (FFD) [18, 29] is adopted because it is simple and can demonstrate the interaction between parameters more accurately.

Table 2: Experimental results of NDE with various values of gm and NP^{ini}

Function			f_1	f_4	f_{15}	f_{18}	f_{22}	f_{30}
D	NP^{ini}	gm	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)
30		3	4.08E+03(6.95E+03)	5.49E-12(3.36E-12)	3.32E+00(3.31E-01)	2.60E+01(1.73E+01)	7.50E+01(6.43E+01)	8.91E+02(3.35E+02)
		5	4.52E+03(4.71E+03)	5.99E-07(1.09E-06)	3.34E+00(7.92E-01)	1.42E+01(8.37E+00)	1.81E+02(1.11E+02)	8.66E+02(4.82E+02)
	5D	10	4.14E+02(1.15E+03)	1.27E+01(2.67E+01)	2.88E+00(7.37E-01)	1.35E+01(5.30E+00)	1.31E+02(5.44E+01)	8.32E+02(3.01E+02)
		15	2.50E+03(2.71E+03)	1.38E-05(1.79E-05)	2.93E+00(6.92E-01)	1.16E+01(7.20E+00)	2.60E+02(2.25E+02)	5.72E+02(1.17E+02)
		20	2.80E+03(4.26E+03)	4.58E-02(9.62E-02)	2.59E+00(8.96E-01)	9.08E+00(2.18E+00)	1.38E+02(1.05E+02)	6.37E+02(1.67E+02)
		3	**4.30E-07(8.90E-07)**	**3.41E-14(7.19E-14)**	3.86E+00(1.04E+00)	1.22E+01(4.19E+00)	7.90E+01(7.61E+01)	1.10E+03(5.72E+02)
		5	1.95E-05(5.65E-05)	5.12E-14(7.31E-14)	3.04E+00(7.40E-01)	1.23E+01(3.42E+00)	7.68E+01(7.24E+01)	7.53E+02(2.69E+02)
	10D	10	5.91E+00(5.58E+00)	2.94E-08(4.84E-08)	**2.60E+00(4.45E-01)**	**5.95E+00(1.50E+00)**	**2.61E+01(4.46E+00)**	**5.14E+02(6.93E+01)**
		15	2.92E+00(7.02E+00)	4.22E-06(9.34E-06)	3.54E+00(8.03E-01)	1.13E+01(4.05E+00)	6.05E+01(6.26E+01)	7.20E+02(2.24E+02)
		20	1.06E+01(2.32E+01)	6.34E+00(2.00E+01)	3.60E+00(1.03E+00)	1.05E+01(3.95E+00)	4.66E+01(3.61E+01)	6.90E+02(1.46E+02)
		3	3.94E-05(2.84E-05)	3.32E-10(4.92E-10)	2.83E+00(8.75E-01)	7.00E+00(2.74E+00)	9.97E+01(6.93E+01)	6.01E+02(1.64E+02)
		5	4.81E+00(9.83E+00)	3.36E-07(2.04E-07)	2.82E+00(4.88E-01)	6.93E+00(1.85E+00)	5.26E+01(6.04E+01)	6.92E+02(2.79E+02)
	15D	10	1.82E-02(5.42E-02)	6.34E+00(2.00E+01)	3.33E+00(9.16E-01)	9.71E+00(3.60E+00)	1.19E+02(1.38E+02)	6.22E+02(1.19E+02)
		15	1.36E+03(1.81E+03)	4.69E-02(5.09E-02)	2.61E+00(7.84E-01)	9.10E+00(6.13E+00)	1.05E+02(9.78E+01)	6.46E+02(2.71E+02)
		20	2.24E+03(2.24E+03)	3.02E-01(4.30E-01)	3.65E+00(6.23E-01)	8.53E+00(5.00E+00)	5.78E+01(5.66E+01)	5.87E+02(8.43E+01)
		3	1.47E-02(1.77E-02)	2.55E-06(3.48E-06)	2.95E+00(3.22E-01)	6.05E+00(1.54E+00)	5.85E+01(6.32E+01)	6.69E+02(3.14E+02)
		5	5.25E-01(4.55E-01)	4.62E-04(9.34E-04)	2.85E+00(1.15E+00)	6.44E+00(2.97E+00)	8.48E+01(6.49E+01)	5.42E+02(1.11E+02)
	20D	10	9.30E-02(2.68E-01)	1.35E-05(2.68E-05)	4.38E+00(1.09E+00)	9.95E+00(4.22E+00)	8.39E+01(8.61E+01)	6.41E+02(1.87E+02)
		15	4.48E+02(3.84E+02)	1.72E-01(2.13E-01)	3.37E+00(1.09E+00)	7.80E+00(4.71E+00)	1.19E+02(8.03E+01)	5.17E+02(4.38E+01)
		20	2.18E+03(9.37E+02)	1.01E+00(2.41E-01)	3.30E+00(1.17E+00)	7.32E+00(5.65E+00)	1.40E+02(1.05E+02)	5.22E+02(6.07E+01)
50		3	6.10E+04(1.79E+04)	3.33E+01(2.22E+01)	6.60E+00(1.40E+00)	7.09E+01(1.56E+01)	4.60E+02(1.97E+02)	9.00E+03(4.14E+02)
		5	8.38E+04(3.89E+04)	7.93E+01(4.19E+01)	5.78E+00(1.01E+00)	8.06E+01(2.86E+01)	6.66E+02(1.45E+02)	8.32E+03(3.70E+02)
	5D	10	7.40E+04(3.35E+04)	3.65E+01(4.43E+01)	5.44E+00(3.02E-01)	5.09E+01(2.13E+01)	3.42E+02(2.81E+02)	8.71E+03(6.84E+02)
		15	1.17E+05(6.93E+04)	3.77E+01(4.31E+01)	5.75E+00(1.04E+00)	4.03E+01(9.77E+00)	4.04E+02(1.20E+02)	8.81E+03(5.82E+02)
		20	1.09E+05(4.37E+04)	3.95E+01(4.39E+01)	5.19E+00(1.03E+00)	3.69E+01(2.42E+01)	4.58E+02(2.24E+02)	9.50E+03(3.44E+02)
		3	1.09E+05(4.37E+04)	3.95E+01(4.39E+01)	5.19E+00(1.03E+00)	3.69E+01(2.42E+01)	4.58E+02(2.24E+02)	9.50E+03(3.44E+02)
		5	6.54E+04(3.17E+04)	5.50E+01(4.71E+01)	5.96E+00(1.65E+00)	5.04E+01(1.32E+01)	4.25E+02(1.58E+02)	8.45E+03(5.09E+02)
	10D	10	6.30E+04(2.54E+04)	**8.19E+00(6.55E-01)**	**4.72E+00(6.11E-01)**	**2.40E+01(5.41E+00)**	**2.11E+02(1.34E+02)**	**8.16E+03(1.70E+02)**
		15	9.25E+04(3.94E+04)	9.99E+00(1.15E+00)	7.08E+00(1.82E+00)	2.84E+01(1.26E+01)	5.70E+02(2.27E+02)	8.15E+03(2.24E+02)
		20	1.23E+05(3.63E+04)	4.92E+01(4.47E+01)	4.92E+00(1.23E+00)	3.84E+01(1.14E+01)	4.58E+02(1.41E+02)	8.60E+03(7.03E+02)
		3	**2.60E+04(1.32E+04)**	2.48E+01(4.10E+01)	6.50E+00(3.02E+00)	2.42E+01(7.75E+00)	6.56E+02(1.80E+02)	8.57E+03(3.58E+02)
		5	7.86E+04(4.80E+04)	4.46E+01(4.88E+01)	5.36E+00(1.78E+00)	3.52E+01(6.17E+00)	3.23E+02(3.89E+02)	8.61E+03(4.73E+02)
	15D	10	6.03E+04(2.56E+04)	6.27E+01(4.86E+01)	5.91E+00(1.25E+00)	3.60E+01(1.43E+01)	2.94E+02(3.36E+02)	8.64E+03(6.78E+02)
		15	1.16E+05(4.49E+04)	5.75E+01(3.73E+01)	5.42E+00(9.83E-01)	3.18E+01(8.89E+00)	5.05E+02(1.98E+02)	8.51E+03(2.94E+02)
		20	1.65E+05(3.36E+04)	6.57E+01(4.44E+01)	5.27E+00(8.08E-01)	4.88E+01(2.99E+01)	3.59E+02(2.70E+02)	8.74E+03(3.26E+02)
		3	3.40E+04(1.78E+04)	4.47E+01(4.87E+01)	5.76E+00(1.12E+00)	2.51E+01(1.10E+01)	9.42E+02(3.67E+02)	8.43E+03(4.67E+02)
		5	5.88E+04(2.69E+04)	4.50E+01(4.85E+01)	5.61E+00(1.02E+00)	3.85E+01(9.79E+00)	4.45E+02(3.78E+02)	8.41E+03(6.27E+02)
	20D	10	9.41E+04(1.01E+05)	6.57E+01(4.45E+01)	5.49E+00(1.48E+00)	3.75E+01(1.28E+01)	9.07E+02(1.71E+02)	8.60E+03(7.03E+02)
		15	1.46E+05(6.44E+04)	9.21E+01(8.21E+00)	6.12E+00(1.19E+00)	3.34E+01(1.12E+01)	6.45E+02(3.87E+02)	8.78E+03(3.81E+02)
		20	3.44E+05(1.30E+05)	7.60E+01(3.19E+01)	6.21E+00(1.95E+00)	4.60E+01(1.43E+01)	7.82E+02(2.94E+02)	8.64E+03(3.73E+02)

In the experiment, gm and NP^{ini} are first set to five and four different levels, *i.e.*, $gm \in \{3, 5, 10, 15, 20\}$ and $NP^{ini} \in \{5D, 10D, 15D, 20D\}$, respectively, and all possible combinations of each level are then run. Other parameters in NDE are consistent with Section 3. Table 2 reports their experimental results when $D = 30$ and 50, where the best results are marked by bold on each function (the same below).

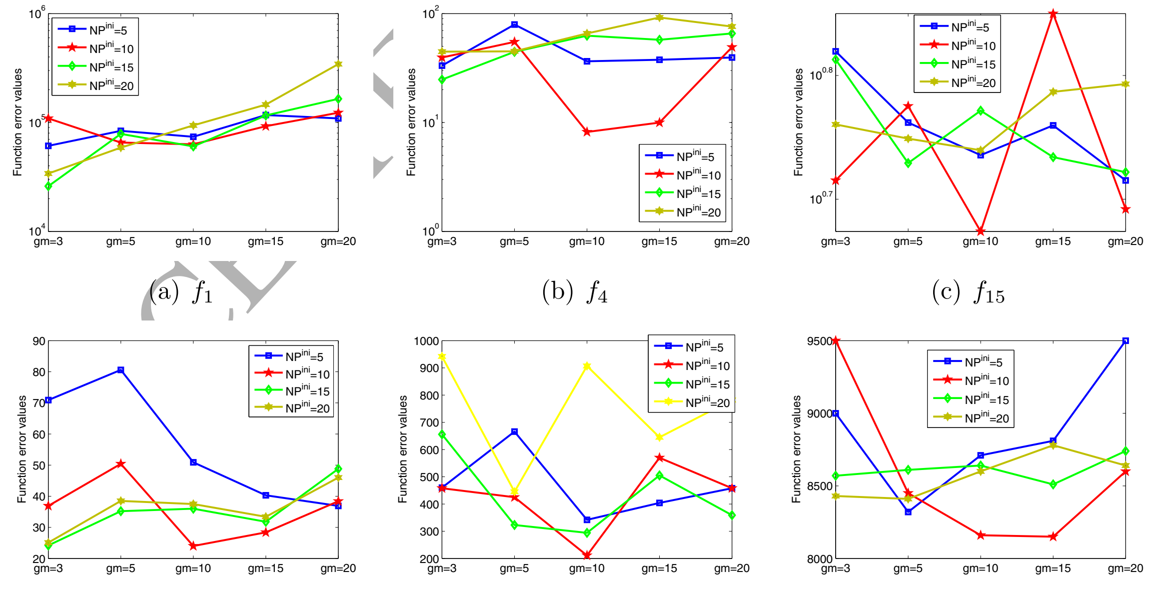
From Table 2, NDE gets the best results on these functions when $NP^{ini} = 10D$ and $gm = 10$ except for f_1 and f_4 when $NP^{ini} = 10D$ and $gm = 3$ for $D = 30$, and f_1 when $NP^{ini} = 15D$ and $gm = 3$ for $D = 50$. To see the interaction between NP^{ini} and gm clearly, Figures 1 and 2 depict the performance of NDE with various values of NP^{ini} and gm on these functions when $D = 30$ and 50, respectively. From Figures 1 and 2, we see that NDE is sensitive to NP^{ini} and gm. In particular, whether $D = 30$ or 50, different values of NP^{ini} or gm result in significant difference on each function for the same gm or NP^{ini}. Then NP^{ini} should not be too small or too large for all problems, while gm should be small for simple functions, and not too small or too large for complicated problems.



(a) f_1 (b) f_4 (c) f_{15}

(d) f_{18} (e) f_{22} (f) f_{30}

Figure 1: Performance of NDE with various values of NP^{ini} and gm when $D = 30$. (a) f_1, (b) f_4, (c) f_{15}, (d) f_{18}, (e) f_{22} and (f) f_{30}.



(a) f_1 (b) f_4 (c) f_{15}

(d) f_{18} (e) f_{22} (f) f_{30}

Figure 2: Performance of NDE with various values of NP^{ini} and gm when $D = 50$. (a) f_1, (b) f_4, (c) f_{15}, (d) f_{18}, (e) f_{22} and (f) f_{30}.

These are consistent with the analysis in Subsections 3.2 and 3.3, respectively. Thus, let $NP^{ini} = 10D$ and $gm = 10$ in the following experiments since the more promising performance is achieved on these functions at this case.

4.2. The effectiveness of the proposed strategies

In this subsection, we illustrate the effectiveness of NM strategy and NAE mechanism.

4.2.1. The effectiveness of the NM strategy

To show the effectiveness of NM strategy, we design three NDE variants, NDE$_{1-1}$, NDE$_{1-2}$ and NDE$_{1-3}$, and compare them with NDE on f_1-f_{30} in Table 1 when $D = 30$. Three variants are NDE with $\vec{v}_i^g = \vec{x}_{nr_1}^g + F(\vec{x}_{r_1}^g - \vec{x}_{r_2}^g)$, $\vec{v}_i^g = \vec{x}_i^g + F(\vec{x}_{nbest}^g - \vec{x}_i^g) + F(\vec{x}_{nr_1}^g - \vec{x}_{nr_2}^g) + F(\vec{x}_{r_1}^g - \vec{x}_{r_2}^g)$ and $\xi_{1,i} = 0.5$, respectively. Obviously, the variant with only one mutant operator or constant probability parameter can illustrate the influence of the combination of mutant operators or individual-based probability parameter setting.

In this experiment, the other parameters in NDE and three variants are consistent with Section 3. Table 3 reports their experimental results, as well as statistical and comparison results of the three tests, and the last five rows summarize them. Here and in the following, "Rank" represents the overall performance ranking of each algorithm, "+", "-" and "≈ " denote that the performance of NDE is better than, worse than, and similar to that of the corresponding method respectively, "R+" and "R-" are the rank sum that NDE is better and worse than the compared algorithm, respectively.

From Table 3, we see that NM strategy is helpful to improve the performance of NDE. According to the statistical results of three tests in Table 3, a) NDE significantly outperforms NDE$_{1-1}$, NDE$_{1-2}$ and NDE$_{1-3}$ on 20, 15 and 18 test functions respectively; b) the overall performance rankings of NDE, NDE$_{1-1}$, NDE$_{1-2}$ and NDE$_{1-3}$ are 1.7, 3.08, 2.72, and 2.5, respectively; and c) R+ values are bigger than R- values in all cases and the significant differences can be observed at 0.05 significant level. Then the combination of mutant operators can enhance the performance of single mutation operator effectively, and the individual-based probability parameter setting makes great progress in improving the performance of the random combination of mutant operators. This might be because the dynamical selection of two mutation operators with different search characteristics is helpful to balance exploration and exploitation of NDE, and the individual-based probability parameter setting suitably adjusts the search ability of each individual. Thus, NM strategy effectively balances the exploration and exploitation of NDE and improves its performance.

Table 3: Experimental results of NDE and NDE_{1-1}, NDE_{1-2} and NDE_{1-3} on CEC 2014 functions with $D = 30$

Function	NDE_{1-1}	NDE_{1-2}	NDE_{1-3}	NDE
	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)
f_1	4.87E+04(4.09E+04)+	**4.15E-04(8.98E-04)-**	1.28E+01(1.83E+01)+	5.91E+00(5.58E+00)
f_2	**0.00E+00(0.00E+00)≈**	**0.00E+00(0.00E+00)≈**	**0.00E+00(0.00E+00)≈**	**0.00E+00(0.00E+00)**
f_3	**0.00E+00(0.00E+00)≈**	**0.00E+00(0.00E+00)≈**	**0.00E+00(0.00E+00)≈**	**0.00E+00(0.00E+00)**
f_4	1.52E+01(2.85E+01)+	**3.98E-13(8.07E-13)-**	1.09E-04(2.04E-04)+	2.94E-08(4.84E-08)
f_5	2.03E+01(6.79E-02)+	**2.01E+01(5.63E-02)≈**	2.02E+01(7.76E-02)+	**2.01E+01(4.71E-02)**
f_6	4.13E+00(1.42E+00)+	5.23E+00(1.35E+00)+	4.49E+00(1.86E+00)+	**3.37E+00(1.36E+00)**
f_7	**0.00E+00(0.00E+00)≈**	**0.00E+00(0.00E+00)≈**	**0.00E+00(0.00E+00)≈**	**0.00E+00(0.00E+00)**
f_8	**0.00E+00(0.00E+00)≈**	**0.00E+00(0.00E+00)≈**	**0.00E+00(0.00E+00)≈**	**0.00E+00(0.00E+00)**
f_9	3.45E+01(1.25E+01)-	2.80E+01(8.86E+00)-	3.18E+01(1.47E+01)-	**2.48E+01(4.48E+00)**
f_{10}	**0.00E+00(0.00E+00)≈**	5.63E-02(3.13E-02)+	**0.00E+00(0.00E+00)≈**	**0.00E+00(0.00E+00)**
f_{11}	1.49E+03(5.40E+02)+	1.67E+03(4.59E+02)+	1.52E+03(4.98E+02)+	**1.27E+03(2.41E+02)**
f_{12}	2.07E-01(8.97E-02)+	**1.22E-01(4.06E-02)≈**	1.75E-01(1.10E-01)+	**1.22E-01(2.82E-02)**
f_{13}	1.30E-01(7.00E-02)+	1.57E-01(5.35E-02)+	1.25E-01(3.45E-02)+	**6.80E-02(1.31E-02)**
f_{14}	2.58E-01(5.67E-02)+	**1.79E-01(3.30E-02)≈**	2.31E-01(5.21E-02)+	2.03E-01(2.64E-02)
f_{15}	4.02E+00(9.09E-01)+	3.06E+00(7.16E-01)+	3.58E+00(1.58E+00)+	**2.60E+00(4.45E-01)**
f_{16}	9.01E+00(6.58E-01)+	8.81E+00(5.19E-01)+	8.72E+00(5.59E-01)+	**8.38E+00(4.13E-01)**
f_{17}	1.83E+02(1.31E+02)+	4.25E+02(2.05E+02)+	1.47E+02(7.52E+01)+	**1.13E+02(5.94E+01)**
f_{18}	8.87E+00(2.18E+00)+	8.85E+00(3.62E+00)+	6.39E+00(3.58E+00)+	**5.95E+00(1.50E+00)**
f_{19}	2.71E+00(7.90E-01)+	3.45E+00(7.10E-01)+	2.86E+00(7.22E-01)+	**2.14E+00(4.61E-01)**
f_{20}	7.21E+00(2.88E+00)+	9.93E+00(2.65E+00)+	5.59E+00(1.51E+00)+	**4.05E+00(9.50E-01)**
f_{21}	5.97E+01(6.59E+01)+	2.02E+02(1.46E+02)+	2.22E+01(3.79E+01)+	**1.01E+01(5.37E+00)**
f_{22}	6.25E+01(5.46E+01)+	5.60E+01(5.63E+01)+	5.15E+01(5.42E+01)+	**2.61E+01(4.46E+00)**
f_{23}	**3.15E+02(0.00E+00)≈**	**3.15E+02(1.44E-13)≈**	**3.15E+02(2.21E-13)≈**	**3.15E+02(2.15E-13)**
f_{24}	2.23E+02(1.11E+00)+	**2.17E+02(9.05E+00)-**	2.20E+02(7.04E+00)-	2.22E+02(1.67E-01)
f_{25}	**2.03E+02(1.69E-01)≈**	**2.03E+02(1.77E-01)≈**	**2.03E+02(1.98E-01)≈**	**2.03E+02(4.91E-02)**
f_{26}	**1.00E+02(5.45E-02)≈**	**1.00E+02(5.06E-02)≈**	**1.00E+02(4.38E-02)≈**	**1.00E+02(1.79E-02)**
f_{27}	**3.61E+02(5.21E+01)-**	4.01E+02(1.54E+00)+	3.90E+02(3.18E+01)≈	3.90E+02(3.06E+01)
f_{28}	8.14E+02(2.66E+01)+	**7.86E+02(1.22E+01)-**	8.16E+02(1.96E+01)+	7.97E+02(1.63E+01)
f_{29}	7.07E+02(8.55E+01)+	7.17E+02(3.57E+00)+	**6.57E+02(1.71E+02)-**	6.66E+02(1.50E+02)
f_{30}	7.72E+02(2.95E+02)+	7.09E+02(2.21E+02)+	6.29E+02(1.73E+02)+	**5.14E+02(6.93E+01)**
+/-/≈	20/2/8	15/5/10	18/3/9	- -
R+/R-	238/15	189.5/41.5	204/27	- -
p-value	0.0003	0.0106	0.0022	- -
$\alpha = 0.05$	YES	YES	YES	- -
Rank	3.08	2.72	2.50	1.70

4.2.2. The effectiveness of the NAE mechanism

To evaluate the effectiveness of NAE mechanism, NDE is compared with its three variants, NDE_{2-1}, NDE_{2-2} and NDE_{2-3}, on f_1-f_{30} in Table 1 when $D = 30$. The variants are NDE without dynamic neighborhood, exchanging operations and NAE mechanism, respectively. Clearly, they can effectively illustrate the influences of NAE mechanism and its each component.

In this experiment, the other parameters in NDE and its variants are consistent with Section 3. Table 4 reports their experimental results, statistical and comparison results. From Table 4, one can see that NAE mechanism and its components have great influences on the performance of algorithm, and NDE is superior to its variants. According to the statistical results of three tests in Table 4, a) NDE is better than NDE_{2-1}, NDE_{2-2} and NDE_{2-3} on 27, 23 and 27 test functions, respectively; b) the overall performance rankings of NDE, NDE_{2-1}, NDE_{2-2} and NDE_{2-3} are 1.22, 2.68, 2.4 and 3.7, respectively; and c) R+ values are bigger than R- values in all cases and the significant differences can be observed at 0.05 significant level. Then NAE mechanism improves the performance of NDE effectively. These might be attributed to the following two facts. 1) The dynamic neighborhood model is helpful to jump out of local optimum. 2) The exchanging operations deal with the premature convergence and stagnation of the corresponding neighborhood.

Table 4: Experimental results of NDE and NDE_{2-1}, NDE_{2-2} and NDE_{2-3} on CEC 2014 functions with $D = 30$

Function	NDE_{2-1}	NDE_{2-2}	NDE_{2-3}	NDE
	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)
f_1	1.77E+05(1.01E+05)+	2.50E+06(6.08E+06)+	6.05E+06(1.24E+07)+	**5.91E+00(5.58E+00)**
f_2	6.18E-06(1.88E-05)+	**0.00E+00(0.00E+00)**≈	7.21E-09(1.98E-08)+	**0.00E+00(0.00E+00)**
f_3	1.06E-03(2.01E-03)+	**0.00E+00(0.00E+00)**≈	2.65E-14(3.87E-14)+	**0.00E+00(0.00E+00)**
f_4	1.06E+01(2.06E+01)+	6.34E+00(2.00E+01)+	7.31E+01(5.23E+01)+	**2.94E-08(4.84E-08)**
f_5	2.02E+01(9.15E-02)+	2.03E+01(1.92E-02)+	2.04E+01(4.72E-02)+	**2.01E+01(4.71E-02)**
f_6	7.68E+00(3.30E+00)+	1.38E+01(1.17E+00)+	1.68E+01(1.77E+00)+	**3.37E+00(1.36E+00)**
f_7	1.02E-13(1.13E-13)+	**0.00E+00(0.00E+00)**≈	9.01E-04(3.32E-03)+	**0.00E+00(0.00E+00)**
f_8	6.97E+00(9.35E+00)+	**0.00E+00(0.00E+00)**≈	**0.00E+00(0.00E+00)**≈	**0.00E+00(0.00E+00)**
f_9	4.48E+01(1.01E+01)+	3.95E+01(4.23E+00)+	6.60E+01(1.30E+01)+	**2.48E+01(4.48E+00)**
f_{10}	1.16E+00(7.08E-01)+	1.82E-13(5.75E-13)+	6.94E-04(3.80E-03)+	**0.00E+00(0.00E+00)**
f_{11}	1.85E+03(4.42E+02)+	1.87E+03(2.98E+02)+	2.27E+03(2.78E+02)+	**1.27E+03(2.41E+02)**
f_{12}	2.41E-01(6.21E-02)+	3.63E-01(6.13E-02)+	4.48E-01(1.01E-01)+	**1.22E-01(2.82E-02)**
f_{13}	1.67E-01(1.96E-02)+	2.76E-01(5.86E-02)+	4.78E-01(1.04E-01)+	**6.80E-02(1.31E-02)**
f_{14}	2.64E-01(4.20E-02)+	2.20E-01(2.30E-02)+	2.94E-01(3.21E-02)+	**2.03E-01(2.64E-02)**
f_{15}	3.53E+00(9.88E-01)+	3.72E+00(6.64E-01)+	7.73E+00(1.90E+00)+	**2.60E+00(4.45E-01)**
f_{16}	9.27E+00(5.11E-01)+	9.58E+00(4.84E-01)+	1.05E+01(3.23E-01)+	**8.38E+00(4.13E-01)**
f_{17}	1.68E+03(1.73E+03)+	3.22E+05(7.27E+05)+	8.26E+05(2.10E+06)+	**1.13E+02(5.94E+01)**
f_{18}	1.45E+01(6.02E+00)+	7.32E+00(2.32E+00)+	9.74E+00(3.18E+00)+	**5.95E+00(1.50E+00)**
f_{19}	3.93E+00(6.03E-01)+	2.61E+00(5.07E-01)+	8.37E+00(3.47E+00)+	**2.14E+00(4.61E-01)**
f_{20}	1.32E+01(4.74E+00)+	1.02E+03(2.13E+03)+	1.24E+04(1.35E+04)+	**4.05E+00(9.50E-01)**
f_{21}	2.50E+02(1.41E+02)+	2.70E+01(4.25E+01)+	4.90E+04(2.09E+05)+	**1.01E+01(5.37E+00)**
f_{22}	1.50E+02(1.32E+02)+	2.14E+02(8.88E+01)+	3.54E+02(1.61E+02)+	**2.61E+01(4.46E+00)**
f_{23}	**3.15E+02(3.18E-13)**≈	**3.15E+02(2.21E-13)**≈	**3.15E+02(8.13E-06)**≈	**3.15E+02(2.15E-13)**
f_{24}	2.23E+02(1.29E+00)+	2.23E+02(1.28E+00)+	2.28E+02(1.07E+00)+	**2.22E+02(1.67E-01)**
f_{25}	**2.03E+02(4.12E-01)**≈	**2.03E+02(1.61E+00)**≈	2.05E+02(4.07E+00)+	**2.03E+02(4.91E-02)**
f_{26}	**1.00E+02(6.64E-02)**≈	**1.00E+02(7.37E-02)**≈	**1.00E+02(1.20E-01)**≈	**1.00E+02(1.79E-02)**
f_{27}	3.92E+02(3.06E+01)+	4.69E+02(1.02E+02)+	6.02E+02(1.31E+02)+	**3.90E+02(3.06E+01)**
f_{28}	8.49E+02(4.20E+01)+	8.33E+02(1.22E+01)+	8.74E+02(4.52E+01)+	**7.97E+02(1.63E+01)**
f_{29}	1.06E+03(1.10E+02)+	7.32E+02(2.91E+02)+	1.54E+03(7.54E+02)+	**6.66E+02(1.50E+02)**
f_{30}	8.17E+02(2.33E+02)+	7.42E+02(4.29E+02)+	3.46E+03(2.75E+03)+	**5.14E+02(6.93E+01)**
+/-/≈	27/0/3	23/0/7	27/0/3	- -
R+/R-	378/0	276/0	378/0	- -
p-value	<0.0001	<0.0001	<0.0001	- -
$\alpha = 0.05$	YES	YES	YES	- -
Rank	2.68	2.40	3.70	1.22

Therefore, NAE mechanism could suitably adjust the search capability of each individual, and improve the performance of algorithm effectively.

4.3. Comparisons and discussions

To evaluate the advantages of NDE, we make a comparison of NDE with 21 well-known optimization algorithms on 30 benchmark functions f_1-f_{30} in Table 1 when $D = 30$ and 50.

These algorithms include the classical DE, five state-of-the-art DE variants (CoDE [40], EPSDE [26], JADE [47], jDE [2] and SaDE [31]), nine up-to-date DE variants (CIPDE [49], CoBiDE [41], dynNP-jDE [3], JADE_sort [50], L-SHADE [37], MPEDE [43], SHADE [36], SinDE [12] and TSDE [23]), and six non-DE algorithms (CLPSO [19], CMA-ES [14], DNLPSO [28], EPSO [25], GL-25 [13] and HSOGA [15]). The classical DE adopts mutation operator "DE/rand/1" to generate the offspring. CoDE [40] implements three mutant strategies with different characteristics simultaneously. Four variants, EPSDE [26], JADE [47], jDE [2] and SaDE [31], adjust their control parameters adaptively. TSDE [23] enhances CoDE [40] by dividing the whole evolutionary process into two stages, and dynNP-jDE [3] improves jDE [2] by presenting a simple schema to reduce population size. JADE_sort [50] and SHADE [36] improve JADE [47] by assigning a smaller CR value to the individual

Table 5: Parameters setting

Algorithms	Parameter setting
DE [35]	$NP = 50,\ F = CR = 0.5$
CoDE [40]	$NP = 30,\ [F = 1.0,\ CR = 0.1],\ [F = 1.0,\ CR = 0.9],\ [F = 0.8,\ CR = 0.2]$
jDE [2]	$NP = 100,\ \tau_1 = \tau_2 = 0.1,\ F_l = 0.1,\ F_u = 0.9$
JADE [47]	$NP = 100,\ \mu F_0 = \mu CR_0 = 0.5,\ c = 0.1,\ p = 0.05$
EPSDE [26]	$NP = 50,\ F \in [0.4,\ 0.9]$ and $CR \in [0.1,\ 0.9]$ with stepsize = 0.1
SaDE [31]	$NP = 50,\ K = 4,\ Lp = 50$
CIPDE [49]	$NP = 100,\ c = 0.1,\ \mu_F = 0.7,\ \mu_{CR} = 0.5,\ T = 90$
CoBiDE [41]	$NP = 60,\ pb = 0.4,\ ps = 0.5$
JADE_sort [50]	$NP = 100,\ \mu F_0 = \mu CR_0 = 0.5,\ c = 0.1,\ p = 0.05$
L-SHADE [37]	$N^{init} = 20D,\ H = 5,\ c = 0.1,\ p = 0.1$
SHADE [36]	$NP = 100,\ H = 2,\ c = 0.1,\ p = rand(0.02, 0.2)$
TSDE [23]	$NP = 30,\ [F = 1.0,\ CR = 0.1],\ [F = 1.0,\ CR = 0.9],\ [F = 0.8,\ CR = 0.2]$
dynNP-jDE [3]	$NP^{init} = 200,\ p_{max} = 4$
MPEDE [43]	$NP = 250,\ c = 0.1,\ \lambda_1 = \lambda_2 = \lambda_3 = 0.2,\ ng = 20$
SinDE [12]	$NP = 40,\ freq = 0.25$
CLPSO [19]	$NP = 30, c_1 = c_2 = 1.494, \omega_{max} = 0.9, \omega_{min} = 0.4, m = 5$
CMA-ES [14]	$NP = 4 + \lfloor 3\ln(D) \rfloor, \mu = \lfloor NP/2 \rfloor, \omega_{i=1,\cdots,\mu} = \ln((NP+1)/2) - \ln(i), C_c = C_\sigma = 4/(D+4)$
GL-25 [13]	$NP = 60, \alpha = 1, \omega = 5, n_T = 2$
EPSO [25]	$NP = 30, g_1 = 15, g_2 = 25$
DNLPSO [28]	$NP = 30, c_1 = c_2 = 1.494, \omega_0 = 0.9, \omega_1 = 0.4$
HSOGA [15]	$NP = 200, S = 5, P_c = 0.6, P_m = 0.1$
NDE	$NP^{ini} = 10D,\ NP^{min} = 5,\ gm = 10,\ F^0_{loc} = CR^0_m = 0.5,\ c = 0.1$

with better fitness value, and using the success history information to adaptively set its parameters, respectively. L-SHADE [37] further extends SHADE [36] by incorporating a linear population size reduction. CoBiDE [41] improves DE algorithm by developing a covariance matrix learning and a bimodal distribution parameter setting. SinDE [12] is a sinusoidal DE variant that uses the sinusoidal formulas to adjust automatically the control parameters. Two recent DE variants, MPEDE [43] and CIPDE [49], employ the concept of work specialization, and the collective information of the best candidates in mutation and crossover, respectively. CLPSO [19] updates the particle velocity by using the personal historical best information of all particles. DNLPSO [28] further enhances CLPSO [19] by adopting a learning strategy and dynamically reforming the neighborhood after a certain interval. EPSO [25] combines different PSO algorithms and employs a self-adaptive scheme to identify the top algorithms according to their previous experiences. Two hybrid GAs, GL-25 [13] and SOGA [15], combines the global and local searches, and employs a self-adaptive orthogonal crossover operator, respectively. CMA-ES [14] is a very efficient evolution strategy (ES). Obviously, these algorithms are more competitive or recently published in the literatures. Thus, they are chosen as the compared ones.

In the following experiments, the parameter settings for them are listed in Table 5, where the control parameter settings of each compared algorithm and NDE are the same as those in its original paper and Section 3, respectively.

4.3.1. Comparison with the classical DE and five state-of-the-art DE variants

First, we compare NDE with the classical DE and five state-of-the-art DE variants on 30 benchmark functions f_1-f_{30} in Table 1. These variants include JADE [47], jDE [2], CoDE [40], SaDE [31] and EPSDE [26].

Table 6 reports their experimental results, the statistical results of Wilcoxon rank sum test and Friedman test when $D = 30$ and 50, and the last two rows summarize them.

When $D = 30$, from Table 6, the following detail results can be observed.

1) NDE obtains the best results on unimodal functions f_1-f_3, and CoDE on f_2. This is because the dynamic neighborhood size is helpful to speed up the convergence of NDE by using the information of the promising individuals.

2) NDE obtains the best results on simple multimodal and hybrid functions f_5, f_7-f_{11}, and f_{13}-f_{22}, DE on f_6, CoDE on f_5, f_8 and f_{12}, JADE on f_4, f_7 and f_8, and EPSDE on f_8.

3) NDE obtains the best results on composition functions f_{24}, f_{26} and f_{30}, EPSDE on f_{23}, f_{25}, f_{26}, f_{28} and f_{29}, and DE on f_{26} and f_{27}. From Wilcoxon rank sum test, NDE is much better than DE, CoDE, jDE, JADE, EPSDE and SaDE on 4, 5, 3, 3, 3 and 7 test functions respectively, and slightly worse on 1, 1, 2, 2, 3 and 0 test functions, respectively.

According to the statistical results of two tests in Table 6, a) NDE performs better than DE, CoDE, jDE, JADE, EPSDE and SaDE on 25, 22, 25, 22, 24 and 29 test functions respectively, slightly worse on 2, 3, 2, 3, 4 and 0 test functions respectively, and similar to that on 3, 5, 3, 5, 2 and 1 test functions, respectively; and b) NDE and others get 1.78, 5.45, 3.18, 3.88, 3.53, 4.63 and 5.53 in term of overall performance ranking on all problems, respectively.

To further show the convergence performance, Figure 3 depicts the evolutionary curves of NDE and five DE variants on 12 typical functions f_1-f_4, f_6-f_8, f_{10}, f_{11}, f_{13}, f_{17} and f_{18}. From Figure 3, we see that NDE has faster convergence and better accuracy than others on these functions except for JADE on f_4, CoDE on f_6, and EPSDE on f_8.

When $D = 50$, from Table 6, we also see that NDE obtains the best results on f_4, f_7, f_9, f_{11}, f_{13}-f_{18}, f_{20}-f_{22} and f_{26}, JADE on f_1, f_2, f_8, and f_{26}, jDE on f_3, f_{10} and f_{26}, DE on f_6, f_{24} and f_{27}, CoDE on f_5, f_{12} and f_{19}, and EPSDE on f_{23}, f_{25}, f_{26} and f_{28}-f_{30}. According to the statistical results of two tests in Table 6, a) NDE performs better than DE, CoDE, jDE, JADE, EPSDE and SaDE on 25, 25, 25, 24, 23 and 29 test functions respectively, slightly worse on 4, 4, 3, 4, 6 and 0 test functions respectively, similar to that

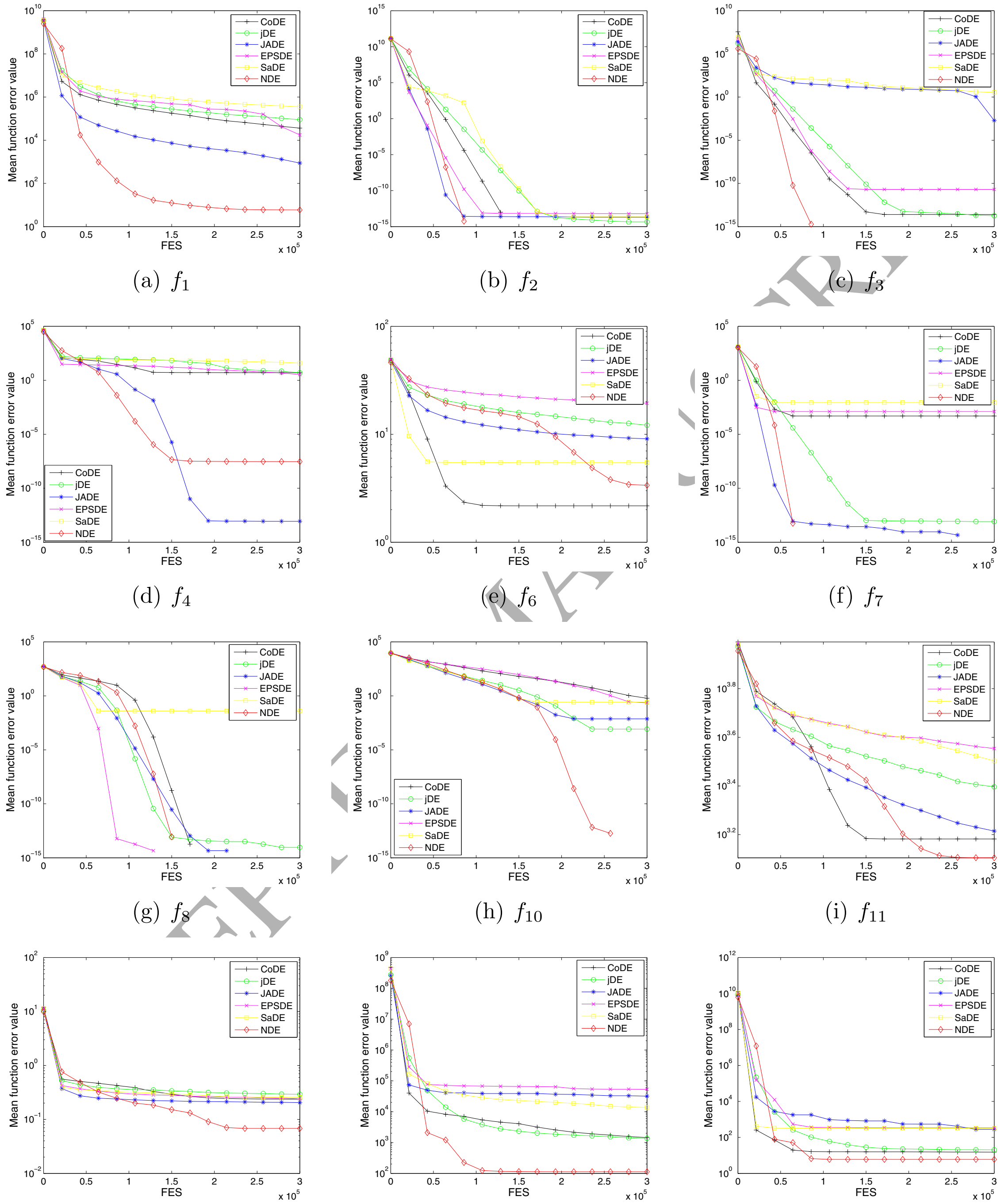
Table 6: Experimental results of NDE, the classical DE and five state-of-the-art DE variants on CEC 2014 functions

D = 30

Func	DE	CoDE	jDE	JADE	EPSDE	SaDE	NDE
	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)
f_1	3.31E+07(8.67E+06)+	3.64E+04(2.32E+04)+	8.99E+04(7.60E+04)+	8.75E+02(1.61E+03)+	1.77E+04(5.26E+04)+	3.57E+05(1.94E+05)+	**5.91E+00(5.58E+00)**
f_2	1.52E-14(1.44E-14)+	**0.00E+00(0.00E+00)**≈	4.55E-15(1.06E-14)+	2.05E-14(1.30E-14)+	6.14E-14(5.42E-14)+	2.39E-14(6.01E-14)+	**0.00E+00(0.00E+00)**
f_3	5.31E-14(1.44E-14)+	2.50E-14(6.98E-14)+	1.82E-14(2.71E-14)+	1.93E-03(6.68E-03)+	2.02E-11(6.42E-11)+	3.51E+00(1.17E+01)+	**0.00E+00(0.00E+00)**
f_4	7.82E+00(2.68E+01)+	5.14E+00(1.78E+01)+	5.00E+00(1.34E+01)+	**8.64E-14(5.22E-14)-**	3.46E+00(2.24E+00)+	3.91E+01(3.76E+01)+	2.94E-08(4.84E-08)
f_5	2.09E+01(7.14E-02)+	**2.01E+01(9.33E-02)**≈	2.03E+01(4.08E-02)+	2.03E+01(3.35E-02)+	2.04E+01(4.79E-02)+	2.05E+01(6.46E-02)+	**2.01E+01(4.71E-02)**
f_6	**1.46E+00(5.44E+00)-**	2.17E+00(2.12E+00)-	1.21E+01(2.50E+00)+	9.07E+00(3.15E+00)+	1.94E+01(3.21E+00)+	5.45E+00(1.48E+00)+	3.37E+00(1.36E+00)
f_7	2.65E-14(4.89E-14)+	4.93E-04(2.46E-03)+	8.19E-14(5.21E-14)+	**0.00E+00(0.00E+00)**≈	1.28E-03(5.09E-03)+	8.55E-03(1.42E-02)+	**0.00E+00(0.00E+00)**
f_8	8.37E+01(7.51E+00)+	**0.00E+00(0.00E+00)**≈	9.09E-15(3.15E-14)+	**0.00E+00(0.00E+00)**≈	**0.00E+00(0.00E+00)**≈	3.98E-02(1.99E-01)+	**0.00E+00(0.00E+00)**
f_9	1.65E+02(9.41E+00)+	3.80E+01(8.84E+00)+	5.08E+01(5.41E+00)+	2.56E+01(3.52E+00)+	4.47E+01(6.39E+00)+	3.94E+01(1.03E+01)+	**2.48E+01(4.48E+00)**
f_{10}	2.81E+03(3.37E+02)+	6.09E-01(5.81E-01)+	8.33E-04(4.16E-03)+	7.49E-03(1.33E-02)+	2.25E-01(1.04E-01)+	2.62E-01(5.25E-01)+	**0.00E+00(0.00E+00)**
f_{11}	6.38E+03(2.92E+02)+	1.52E+03(4.34E+02)+	2.49E+03(2.12E+02)+	1.64E+03(2.51E+02)+	3.58E+03(4.43E+02)+	3.18E+03(7.56E+02)+	**1.27E+03(2.41E+02)**
f_{12}	2.04E+00(2.74E-01)+	**6.23E-02(3.60E-02)-**	4.36E-01(5.12E-02)+	2.62E-01(3.18E-02)+	5.29E-01(1.36E-01)+	8.32E-01(1.10E-01)+	1.22E-01(2.82E-02)
f_{13}	3.37E-01(4.61E-02)+	2.37E-01(4.57E-02)+	2.91E-01(4.19E-02)+	2.05E-01(3.40E-02)+	2.41E-01(4.75E-02)+	2.56E-01(3.67E-02)+	**6.80E-02(1.31E-02)**
f_{14}	2.63E-01(2.68E-02)+	2.30E-01(3.82E-02)+	2.88E-01(2.38E-02)+	2.41E-01(2.81E-02)+	2.87E-01(5.96E-02)+	2.42E-01(2.97E-02)+	**2.03E-01(2.64E-02)**
f_{15}	1.58E+01(7.67E-01)+	2.87E+00(7.18E-01)+	5.61E+00(7.84E-01)+	3.07E+00(3.37E-01)+	5.61E+00(8.01E-01)+	4.20E+00(1.40E+00)+	**2.60E+00(4.45E-01)**
f_{16}	1.21E+01(2.58E-01)+	9.26E+00(8.05E-01)+	1.01E+01(2.96E-01)+	9.43E+00(4.09E-01)+	1.13E+01(3.87E-01)+	1.10E+01(3.35E-01)+	**8.38E+00(4.13E-01)**
f_{17}	8.19E+05(3.04E+05)+	1.45E+03(1.24E+03)+	1.37E+03(1.30E+03)+	3.18E+04(1.53E+05)+	5.34E+04(4.89E+04)+	1.37E+04(1.04E+04)+	**1.13E+02(5.94E+01)**
f_{18}	9.07E+02(1.07E+03)+	1.51E+01(6.93E+00)+	2.03E+01(6.54E+00)+	2.77E+02(7.24E+02)+	3.36E+02(4.48E+02)+	3.22E+02(3.99E+02)+	**5.95E+00(1.50E+00)**
f_{19}	5.73E+00(3.89E-01)+	2.96E+00(1.09E+00)+	4.73E+00(6.84E-01)+	4.32E+00(9.11E-01)+	1.30E+01(1.29E+00)+	6.75E+00(1.20E+01)+	**2.14E+00(4.61E-01)**
f_{20}	6.99E+01(9.57E+00)+	1.40E+01(1.01E+01)+	1.16E+01(3.94E+00)+	3.45E+03(3.18E+03)+	6.43E+01(8.97E+01)+	1.55E+02(1.99E+02)+	**4.05E+00(9.50E-01)**
f_{21}	1.64E+04(6.93E+03)+	2.49E+02(1.41E+02)+	3.24E+02(1.84E+02)+	3.29E+03(1.51E+04)+	6.86E+03(1.17E+04)+	2.74E+03(3.43E+03)+	**1.01E+01(5.37E+00)**
f_{22}	1.53E+02(7.58E+01)+	1.55E+02(7.12E+01)+	1.10E+02(5.05E+01)+	1.61E+02(7.09E+01)+	2.17E+02(9.82E+01)+	1.58E+02(6.93E+01)+	**2.61E+01(4.46E+00)**
f_{23}	3.15E+02(5.78E-14)≈	3.15E+02(1.28E-13)≈	3.15E+02(0.00E+00)≈	3.15E+02(0.00E+00)≈	**3.14E+02(8.44E-13)-**	3.15E+02(2.23E-13)≈	3.15E+02(2.15E-13)
f_{24}	2.23E+02(1.17E+00)+	2.25E+02(2.79E+00)+	2.24E+02(1.25E+00)+	2.25E+02(2.83E+00)+	2.27E+02(5.21E+00)+	2.27E+02(4.21E+00)+	**2.22E+02(1.67E-01)**
f_{25}	2.12E+02(2.11E+00)+	2.04E+02(1.48E+00)+	2.03E+02(6.46E-01)≈	2.03E+02(6.92E-01)≈	**2.01E+02(8.27E-01)-**	2.09E+02(2.58E+00)+	2.03E+02(4.91E-02)
f_{26}	**1.00E+02(3.76E-02)**≈	**1.00E+02(4.57E-02)**≈	**1.00E+02(3.45E-02)**≈	**1.00E+02(3.56E-02)**≈	**1.00E+02(3.82E-02)**≈	1.04E+02(2.00E+01)+	**1.00E+02(1.79E-02)**
f_{27}	**3.06E+02(1.80E+01)-**	3.85E+02(3.49E+01)-	3.85E+02(3.45E+01)-	3.45E+02(4.96E+01)-	8.55E+02(8.92E+01)+	4.15E+02(3.53E+01)+	3.90E+02(3.06E+01)
f_{28}	7.97E+02(3.18E+01)≈	8.26E+02(3.08E+01)+	7.91E+02(2.14E+01)-	7.90E+02(4.57E+01)-	**3.96E+02(1.21E+01)-**	8.86E+02(2.56E+01)+	7.97E+02(1.63E+01)
f_{29}	3.59E+03(1.15E+03)+	7.87E+02(1.50E+02)+	8.39E+02(1.04E+02)+	7.26E+02(9.93E+00)+	**2.14E+02(1.21E+00)-**	1.07E+03(2.23E+02)+	6.66E+02(1.50E+02)
f_{30}	2.16E+03(5.13E+02)+	8.46E+02(3.30E+02)+	1.58E+03(6.45E+02)+	1.64E+03(6.24E+02)+	5.36E+02(1.46E+02)+	1.85E+03(5.05E+02)+	**5.14E+02(6.93E+01)**
+/-/≈	25/2/3	22/3/5	25/2/3	22/3/5	24/4/2	29/0/1	- -
Rank	5.45	3.18	3.88	3.53	4.63	5.53	1.78

D = 50

Func	DE	CoDE	jDE	JADE	EPSDE	SaDE	NDE
	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)
f_1	1.56E+08(2.77E+07)+	2.23E+05(9.39E+04)+	4.96E+05(1.94E+05)+	**1.55E+04(8.53E+03)-**	1.64E+06(5.84E+06)+	9.61E+05(1.86E+05)+	6.30E+04(2.54E+04)
f_2	5.50E+03(4.36E+03)+	3.25E+01(5.80E+01)+	1.02E-08(1.87E-08)-	**9.21E-14(4.29E-14)-**	8.00E-09(1.35E-08)-	3.36E+03(2.57E+03)+	3.31E-07(4.22E-07)
f_3	1.26E+04(1.84E+03)+	2.86E+01(4.42E+01)+	**3.81E-09(6.74E-09)-**	3.77E+03(2.31E+03)+	7.55E-05(2.40E-04)+	3.57E+03(1.67E+03)+	2.03E-07(3.00E-07)
f_4	9.38E+01(5.83E+00)+	2.99E+01(3.47E+01)+	8.80E+01(1.69E+01)+	2.35E+01(4.28E+01)+	3.71E+01(2.09E+01)+	8.09E+01(3.64E+01)+	**8.19E+00(6.55E-01)**
f_5	2.11E+01(4.09E-02)+	**2.01E+01(8.45E-02)-**	2.04E+01(2.53E-02)+	2.04E+01(3.25E-02)+	2.06E+01(7.18E-02)+	2.07E+01(3.80E-02)+	2.03E+01(4.57E-02)
f_6	**1.62E+00(8.29E+00)-**	8.67E+00(3.51E+00)-	2.85E+01(1.57E+00)+	1.70E+01(5.77E+00)+	4.71E+01(5.80E+00)+	1.84E+01(2.86E+00)+	1.53E+01(2.44E+00)
f_7	1.14E-13(0.00E+00)+	8.88E-04(2.45E-03)+	2.73E-13(1.09E-13)+	9.86E-04(2.76E-03)+	6.01E-03(8.29E-03)+	1.13E-02(1.56E-02)+	**0.00E+00(0.00E+00)**
f_8	2.28E+02(1.14E+01)+	6.37E-01(6.96E-01)+	9.09E-14(4.64E-14)+	**9.09E-15(3.15E-14)-**	1.59E-01(6.21E-01)+	1.07E+00(1.15E+00)+	5.68E-14(5.78E-14)
f_9	3.61E+02(1.28E+01)+	7.74E+01(1.52E+01)+	9.78E+01(1.03E+01)+	5.17E+01(6.15E+00)+	1.46E+02(1.46E+01)+	8.74E+01(1.22E+01)+	**4.15E+01(6.53E+00)**
f_{10}	7.90E+03(3.44E+02)+	5.17E+00(2.39E+00)+	**2.50E-03(5.10E-03)-**	1.20E-02(1.51E-02)-	7.94E+02(6.06E+02)+	6.44E+00(2.40E+01)+	9.92E-02(2.36E-02)
f_{11}	1.28E+04(3.64E+02)+	4.63E+03(8.88E+02)+	5.33E+03(3.67E+02)+	3.91E+03(2.35E+02)+	9.35E+03(1.06E+03)+	6.57E+03(1.61E+03)+	**3.62E+03(4.24E+02)**
f_{12}	3.08E+00(3.03E-01)+	**8.13E-02(4.07E-02)-**	4.82E-01(4.29E-02)+	2.59E-01(2.83E-02)+	9.20E-01(3.07E-01)+	1.03E+00(1.54E-01)+	2.30E-01(3.85E-02)
f_{13}	4.66E-01(5.16E-02)+	3.24E-01(5.15E-02)+	3.83E-01(3.76E-02)+	3.30E-01(4.62E-02)+	3.67E-01(6.09E-02)+	4.37E-01(5.55E-02)+	**1.16E-01(1.67E-02)**
f_{14}	3.13E-01(1.07E-01)+	2.67E-01(3.27E-02)+	3.71E-01(1.33E-01)+	3.11E-01(7.89E-02)+	3.39E-01(8.17E-02)+	3.03E-01(2.98E-02)+	**2.45E-01(3.11E-02)**
f_{15}	3.30E+01(1.71E+00)+	6.46E+00(1.44E+00)+	1.21E+01(1.57E+00)+	7.21E+00(7.48E-01)+	1.85E+01(3.40E+00)+	1.49E+01(3.51E+00)+	**4.72E+00(6.11E-01)**
f_{16}	2.20E+01(2.24E-01)+	1.85E+01(7.62E-01)+	1.83E+01(3.48E-01)+	1.77E+01(3.88E-01)+	2.11E+01(6.85E-01)+	2.01E+01(3.51E-01)+	**1.71E+01(5.61E-01)**
f_{17}	9.24E+06(2.71E+06)+	1.38E+04(8.19E+03)+	1.90E+04(9.87E+03)+	2.22E+03(7.03E+02)+	2.57E+05(1.59E+05)+	7.04E+04(3.68E+04)+	**7.76E+02(1.94E+02)**
f_{18}	1.03E+03(5.91E+02)+	2.95E+02(2.81E+02)+	4.54E+02(5.08E+02)+	1.85E+02(4.98E+01)+	2.69E+03(2.38E+03)+	6.12E+02(4.49E+02)+	**2.40E+01(5.41E+00)**
f_{19}	1.31E+01(4.79E-01)+	**6.47E+00(9.00E-01)-**	1.22E+01(2.83E+00)+	1.27E+01(5.27E+00)+	2.54E+01(1.22E+00)+	1.93E+01(1.21E+01)+	8.40E+00(9.00E-01)
f_{20}	3.19E+03(9.56E+02)+	3.04E+02(2.44E+02)+	5.53E+01(2.44E+01)+	6.94E+03(6.31E+03)+	4.53E+02(1.13E+03)+	1.07E+03(9.53E+02)+	**2.24E+01(5.95E+00)**
f_{21}	3.57E+06(1.02E+06)+	7.55E+03(6.84E+03)+	1.21E+04(1.23E+04)+	1.25E+03(3.88E+02)+	7.92E+04(6.54E+04)+	6.00E+04(4.07E+04)+	**3.51E+02(9.42E+01)**
f_{22}	9.82E+02(1.35E+02)+	6.69E+02(2.02E+02)+	5.39E+02(1.03E+02)+	4.73E+02(1.72E+02)+	7.99E+02(1.69E+02)+	5.33E+02(1.53E+02)+	**2.11E+02(1.34E+02)**
f_{23}	3.44E+02(1.88E-13)≈	3.44E+02(2.32E-13)≈	3.44E+02(2.18E-13)≈	3.44E+02(1.71E-13)≈	**3.37E+02(2.69E-12)-**	3.44E+02(2.32E-13)≈	3.44E+02(2.89E-13)
f_{24}	**2.65E+02(2.64E+00)-**	2.71E+02(1.95E+00)+	2.68E+02(2.43E+00)+	2.74E+02(2.27E+00)+	2.73E+02(7.49E+00)+	2.77E+02(3.38E+00)+	2.67E+02(2.72E+00)
f_{25}	2.38E+02(5.77E+00)+	2.10E+02(5.44E+00)+	2.08E+02(2.06E+00)+	2.17E+02(6.63E+00)+	**2.02E+02(3.63E+00)-**	2.20E+02(7.91E+00)+	2.05E+02(3.01E-01)
f_{26}	1.05E+02(2.35E+01)+	1.08E+02(2.76E+01)+	**1.00E+02(4.34E-02)**≈	**1.00E+02(1.30E-01)**≈	**1.00E+02(5.03E-02)**≈	1.68E+02(4.75E+01)+	**1.00E+02(2.95E-02)**
f_{27}	**3.44E+02(1.70E+01)-**	5.22E+02(5.89E+01)+	4.62E+02(6.89E+01)+	4.55E+02(5.09E+01)+	1.56E+03(7.27E+01)+	7.83E+02(8.39E+01)+	3.50E+02(2.77E+01)
f_{28}	1.06E+03(6.55E+01)-	1.17E+03(4.30E+01)+	1.12E+03(4.92E+01)+	1.15E+03(1.26E+02)+	**3.89E+02(1.33E+01)-**	1.41E+03(1.16E+02)+	1.11E+03(3.07E+01)
f_{29}	2.05E+04(9.51E+03)+	9.16E+02(1.24E+02)+	1.05E+03(1.95E+02)+	8.77E+02(6.56E+01)+	**2.24E+02(3.81E+00)-**	1.35E+03(2.99E+02)+	7.50E+02(5.63E+01)
f_{30}	8.33E+03(3.13E+02)+	9.19E+03(4.29E+02)+	8.57E+03(4.98E+02)+	9.75E+03(7.51E+02)+	**1.16E+03(2.26E+02)-**	1.19E+04(1.64E+03)+	8.16E+03(1.70E+02)
+/-/≈	25/4/1	25/4/1	25/3/2	24/4/2	23/6/1	29/0/1	- -
Rank	5.58	3.48	3.78	3.28	4.58	5.45	1.83



(a) f_1 (b) f_2 (c) f_3
(d) f_4 (e) f_6 (f) f_7
(g) f_8 (h) f_{10} (i) f_{11}
(j) f_{13} (k) f_{17} (l) f_{18}

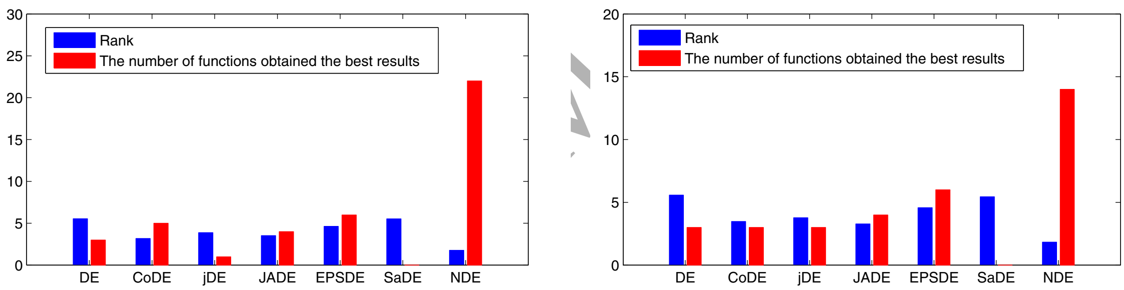
Figure 3: Evolution curves of NDE and five state-of-the-art DE variants with $D = 30$. (a) f_1, (b) f_2, (c) f_3, (d) f_4, (e) f_6, (f) f_7, (g) f_8, (h) f_{10}, (i) f_{11}, (j) f_{13}, (k) f_{17} and (l) f_{18}.

Table 7: Comparison results of NDE with the classical DE and five state-of-the-art DE variants based on the multiproblem Wilcoxon signed-rank test on CEC2014 functions

$D = 30$					$D = 50$				
Algorithm	R+	R-	p-value	$\alpha = 0.05$	Algorithm	R+	R-	p-value	$\alpha = 0.05$
NDE vs DE	353	25	<0.0001	YES	NDE vs DE	395	40	0.0001	YES
NDE vs CoDE	297	28	0.0003	YES	NDE vs CoDE	406	29	<0.0001	YES
NDE vs jDE	347.5	30.5	0.0001	YES	NDE vs jDE	394	12	<0.0001	YES
NDE vs JADE	292	33	0.0005	YES	NDE vs JADE	369	37	0.0002	YES
NDE vs EPSDE	342	64	0.0016	YES	NDE vs EPSDE	347	88	0.0053	YES
NDE vs SaDE	435	0	<0.0001	YES	NDE vs SaDE	435	0	<0.0001	YES

on 1, 1, 2, 2, 1 and 1 test functions, respectively; and b) they get 1.83, 5.58, 3.48, 3.78, 3.28, 4.58 and 5.45 in term of overall performance ranking on all problems, respectively.

For clarity, Figure 4 depicts the bar charts of the statistical results of NDE and other compared algorithms on all functions from CEC 2014 when $D = 30$ and 50, where the blue and red bars represent the overall performance ranking of the Friedman test and the number of function obtained the best results, respectively. From Figure 4, we see that NDE has the best ranking and the most number of the best results on all functions.



(a) $D = 30$ (b) $D = 50$

Figure 4: Statistical results of NDE with the classical DE and five state-of-the-art DE variants on CEC 2014. (a) $D = 30$, (b) $D = 50$.

Furthermore, Table 7 provides the comparison results of NDE with others on all problems based on the multiproblem Wilcoxon signed-rank test when $D = 30$ and 50. From Table 7, we see that NDE obtains higher R+ values than R- values in all cases, and there are significant differences at 0.05 significant level. These might be due to the following two facts. 1) NAE mechanism can identify the neighborhood evolutionary state of each individual and effectively alleviate its evolutionary dilemmas. 2) NM strategy adaptively adjusts its search capability by making full use of the characteristic of each individual to choose a more suitable mutation operator. Therefore, NDE has better performance than DE and five DE variants on these instances.

4.3.2. Comparison with nine up-to-date DE variants

Second, we make a comparison of NDE with nine up-to-date DE variants on 30 benchmark functions f_1-f_{30} in Table 1. These variants include CIPDE [49], CoBiDE [41], SinDE [12], dynNP-jDE [3], MPEDE [43], TSDE [23], JADE_sort [50], SHADE [36] and L-SHADE [37].

Tables 8-9 report their experimental results, the statistical results of Wilcoxon rank sum test and Friedman test when $D = 30$ and 50 respectively, and the last two rows summarize them.

When $D = 30$, from Table 8, the following two results can be observed. 1) L-SHADE obtains the best results on unimodal functions f_1-f_3, NDE and CoBiDE on f_2 and f_3, TSDE and SinDE on f_2. This might be because L-SHADE employs better individuals to guide the search and the population size reduction to adjust the population size. 2) For other functions, NDE obtains the best results on f_4, f_6-f_8, f_{10}, f_{11}, f_{13}-f_{19}, f_{21}-f_{26} and f_{30}, JADE_sort on f_5, f_9 and f_{12}, L-SHADE on f_4, f_{15}, f_{20} and f_{27}, dynNP-jDE on f_{28}, TSDE on f_5, and MPEDE on f_6 and f_{29}.

From the statistical results in Table 8, a) NDE performs better than CIPDE, CoBiDE, JADE_sort, L-SHADE, SHADE, TSDE, dynNP-jDE, MPEDE and SinDE on 23, 20, 23, 18, 25, 21, 24, 25 and 22 test functions respectively, slightly worse on 4, 3, 4, 6, 2, 5, 3, 2 and 3 test functions respectively, and similar to that on 3, 7, 3, 6, 3, 4, 3, 3 and 5 test functions, respectively; and b) NDE and others get 2.72, 7.13, 4.92, 5.15, 3.65, 6.27, 5.75, 6.1, 6.63 and 6.68 in term of overall performance ranking on all problems, respectively.

When $D = 50$, from Table 9, we see that NDE obtains the best results on f_4, f_7 and f_{13}-f_{18}, f_{21}-f_{23}, f_{25}, f_{26} and f_{30}, CIPDE on f_8 and f_{23}, JADE_sort on f_3, f_5, f_9, f_{11}, f_{12} and f_{23}, L-SHADE on f_1 and f_2, f_{20}, f_{23}, f_{25} and f_{26}, SHADE on f_{10}, f_{23} and f_{26}, TSDE on f_{19} and f_{23}, dynNP-jDE and CoBiDE on f_{23} and f_{26}, MPEDE on f_{23}, f_{26} and f_{29}, and SinDE on f_6, f_{23}, f_{24}, f_{27} and f_{28}. From the statistical results in Table 9, a) NDE performs better than CIPDE, CoBiDE, JADE_sort, L-SHADE, SHADE, TSDE, dynNP-jDE, MPEDE and SinDE on 25, 23, 21, 22, 23, 25, 24, 26 and 25 test functions respectively, slightly worse on 4, 4, 8, 4, 3, 4, 4, 2 and 4 test functions respectively, and similar to that on 1, 3, 1, 4, 4, 1, 2, 2 and 1 test functions, respectively; and b) they get 2.55, 6.6, 5.4, 5.3, 4.08, 5.93, 6.25, 5.87, 6.4 and 6.62 in term of overall performance ranking on all problems, respectively.

For clarity, Figure 5 depicts the bar charts of the statistical results of NDE and other compared algorithms on all functions from CEC 2014 when $D = 30$ and 50, where the blue and red bars are same as Figure 4. From Figure 5, we see that NDE has the best rank and the most number of best results for all functions.

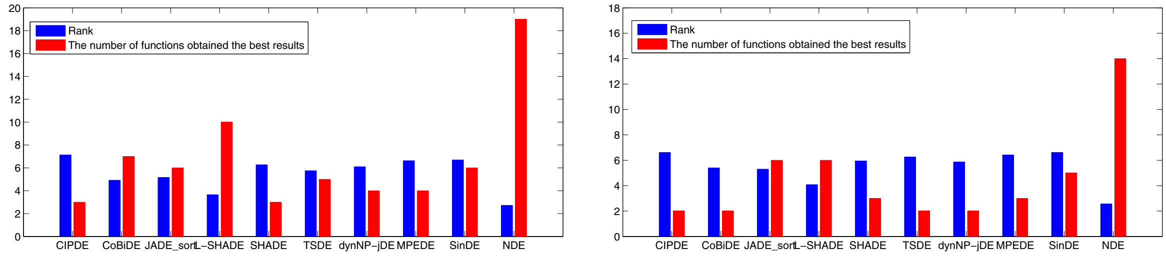
Furthermore, Table 10 provides the comparison results of NDE with others on all problems based on the multiproblem Wilcoxon signed-rank test when $D = 30$ and 50. From

Table 8: Experimental results of NDE and nine up-to-date DE variants on CEC 2014 functions with $D = 30$

Function	Statistic	CIPDE	CoBiDE	JADE_sort	L-SHADE	SHADE	TSDE	dynNP-jDE	MPEDE	SinDE	NDE
f_1	Mean Error	2.86E+03+	1.55E+04+	1.27E+02+	**1.19E-14-**	2.35E+02+	1.52E+04+	3.23E+04+	1.06E-03-	1.33E+06+	5.91E+00
	Std Dev	2.72E+03	1.27E+04	4.98E+02	**5.32E-15**	4.27E+02	1.38E+04	2.19E+04	2.36E-03	1.00E+06	5.58E+00
f_2	Mean Error	2.96E-14+	**0.00E+00≈**	2.05E-14+	**0.00E+00≈**	1.71E-14+	**0.00E+00≈**	9.09E-15+	7.10E-06+	**0.00E+00≈**	**0.00E+00**
	Std Dev	5.68E-15	**0.00E+00**	1.30E-14	**0.00E+00**	1.42E-14	**0.00E+00**	1.35E-14	9.03E-06	**0.00E+00**	**0.00E+00**
f_3	Mean Error	2.53E-01+	**0.00E+00≈**	3.87E-14+	**0.00E+00≈**	3.41E-14+	4.55E-15+	5.23E-14+	7.53E-08+	6.11E-11+	**0.00E+00**
	Std Dev	4.62E-01	**0.00E+00**	2.71E-14	**0.00E+00**	2.84E-14	1.57E-14	1.57E-14	1.27E-07	2.85E-10	**0.00E+00**
f_4	Mean Error	1.66E-13-	8.07E-06+	2.54E+00+	**4.55E-14-**	5.46E-14-	2.54E+00+	1.21E+00+	1.93E-01+	3.07E+01+	2.94E-08
	Std Dev	1.26E-13	3.09E-05	1.27E+01	**2.84E-14**	3.47E-14	1.27E+01	8.96E-01	4.48E-01	2.91E+01	4.84E-08
f_5	Mean Error	2.06E+01+	2.03E+01+	**2.00E+01-**	2.02E+01+	2.02E+01+	**2.00E+01-**	2.03E+01+	2.04E+01+	2.06E+01+	2.01E+01
	Std Dev	3.30E-02	2.70E-01	**2.78E-02**	3.94E-02	3.71E-02	**6.00E-02**	3.06E-02	4.92E-02	4.04E-02	4.71E-02
f_6	Mean Error	4.53E+00+	1.45E+00-	7.23E-01-	9.84E+00+	9.66E+00+	1.58E+00-	2.15E+00-	1.54E+01+	**3.73E-02-**	3.37E+00
	Std Dev	2.06E+00	1.49E+00	6.59E-01	2.26E+00	3.56E+00	1.34E+00	1.46E+00	9.41E-01	**1.80E-01**	1.36E+00
f_7	Mean Error	6.82E-14+	**0.00E+00≈**	2.96E-04+	**0.00E+00≈**	3.55E-03+	2.96E-04+	2.00E-13+	5.32E-11+	**0.00E+00≈**	**0.00E+00**
	Std Dev	5.68E-14	**0.00E+00**	1.48E-03	**0.00E+00**	6.28E-03	1.48E-03	2.21E-13	1.19E-10	**0.00E+00**	**0.00E+00**
f_8	Mean Error	**0.00E+00≈**	**0.00E+00≈**	8.44E+00+	5.00E-14+	5.00E-14+	3.98E-02+	4.55E-15+	8.61E+00+	2.05E-01+	**0.00E+00**
	Std Dev	**0.00E+00**	**0.00E+00**	2.70E+00	5.76E-14	5.76E-14	1.99E-01	2.27E-14	9.02E-01	5.47E-01	**0.00E+00**
f_9	Mean Error	2.07E+01-	3.73E+01+	**1.00E+01-**	1.88E+01-	2.59E+01+	3.72E+01+	3.66E+01+	5.54E+01+	3.10E+01+	2.48E+01
	Std Dev	7.21E+00	6.97E+00	**2.00E+00**	5.89E+00	8.67E+00	1.20E+01	4.81E+00	7.09E+00	7.62E+00	4.48E+00
f_{10}	Mean Error	1.07E+02+	5.57E+01+	2.68E+02+	3.33E-03+	1.08E-02+	2.29E+00+	9.99E-03+	2.02E+02+	7.81E+01+	**0.00E+00**
	Std Dev	3.03E+01	1.46E+01	2.35E+02	1.30E-02	1.49E-02	2.49E+00	1.49E-02	2.76E+01	2.42E+01	**0.00E+00**
f_{11}	Mean Error	2.45E+03+	1.61E+03+	1.57E+03+	1.42E+03+	1.61E+03+	2.00E+03+	1.89E+03+	3.32E+03+	1.94E+03+	**1.27E+03**
	Std Dev	4.88E+02	4.27E+02	3.99E+02	2.21E+02	2.45E+02	4.15E+02	1.95E+02	2.42E+02	5.52E+02	**2.41E+02**
f_{12}	Mean Error	8.74E-01+	2.38E-01+	**7.24E-02-**	2.21E-01+	2.37E-01+	8.14E-02-	3.57E-01+	6.36E-01+	9.98E-01+	1.22E-01
	Std Dev	1.44E-01	3.19E-01	**5.76E-02**	4.58E-02	3.41E-02	3.68E-02	5.08E-02	9.11E-02	1.01E-01	2.82E-02
f_{13}	Mean Error	9.24E-02+	2.42E-01+	1.40E-01+	1.68E-01+	2.21E-01+	2.37E-01+	2.74E-01+	2.24E-01+	2.40E-01+	**6.80E-02**
	Std Dev	2.35E-02	6.87E-02	3.29E-02	2.54E-02	3.91E-02	5.66E-02	4.91E-02	2.56E-02	3.41E-02	**1.31E-02**
f_{14}	Mean Error	2.91E-01+	2.33E-01+	2.79E-01+	2.36E-01+	2.58E-01+	2.37E-01+	2.60E-01+	2.08E-01+	2.40E-01+	**2.03E-01**
	Std Dev	2.76E-02	4.56E-02	4.58E-02	2.13E-02	5.62E-02	3.60E-02	3.53E-02	2.08E-02	2.80E-02	**2.64E-02**
f_{15}	Mean Error	4.38E+00+	3.29E+00+	2.61E+00+	**2.38E+00-**	2.74E+00+	2.95E+00+	4.94E+00+	6.21E+00+	3.99E+00+	2.60E+00
	Std Dev	9.80E-01	7.72E-01	3.48E-01	**2.37E-01**	4.65E-01	7.13E-01	6.10E-01	7.58E-01	8.95E-01	4.45E-01
f_{16}	Mean Error	8.45E+00+	1.00E+01+	9.21E+00+	9.13E+00+	9.52E+00+	9.60E+00+	9.36E+00+	1.06E+01+	1.08E+01+	**8.38E+00**
	Std Dev	7.90E-01	7.19E-01	8.32E-01	3.95E-01	3.56E-01	6.84E-01	3.91E-01	2.27E-01	4.43E-01	**4.13E-01**
f_{17}	Mean Error	1.51E+04+	2.50E+02+	2.95E+02+	2.14E+02+	8.93E+02+	9.98E+02+	8.21E+02+	1.77E+02+	9.28E+04+	**1.13E+02**
	Std Dev	6.94E+04	1.48E+02	1.23E+02	1.11E+02	3.73E+02	8.54E+02	5.43E+02	1.20E+02	6.91E+04	**5.94E+01**
f_{18}	Mean Error	9.74E+01+	1.14E+01+	9.97E+00+	6.00E+00+	5.27E+01+	1.26E+01+	2.26E+01+	9.14E+00+	4.82E+02+	**5.95E+00**
	Std Dev	3.17E+01	4.03E+00	4.52E+00	2.33E+00	2.27E+01	5.24E+00	1.46E+01	3.55E+00	6.17E+02	**1.50E+00**
f_{19}	Mean Error	4.52E+00+	2.73E+00+	3.69E+00+	3.71E+00+	4.68E+00+	2.63E+00+	4.43E+00+	3.57E+00+	3.41E+00+	**2.14E+00**
	Std Dev	5.95E-01	4.09E-01	7.25E-01	5.04E-01	7.63E-01	3.89E-01	3.67E-01	7.83E-01	6.96E-01	**4.61E-01**
f_{20}	Mean Error	8.74E+02+	7.71E+00+	5.62E+00+	**3.24E+00-**	1.83E+01+	9.61E+00+	7.83E+00+	1.14E+01+	9.01E+00+	4.05E+00
	Std Dev	1.26E+03	3.16E+00	3.09E+00	**1.54E+00**	9.42E+00	3.97E+00	2.35E+00	3.34E+00	2.88E+00	9.50E-01
f_{21}	Mean Error	7.91E+03+	1.36E+02+	1.16E+02+	1.04E+02+	2.72E+02+	1.89E+02+	1.50E+02+	8.79E+01+	3.84E+03+	**1.01E+01**
	Std Dev	2.76E+04	9.30E+01	8.18E+01	1.01E+02	9.71E+01	1.25E+02	1.03E+02	9.36E+01	4.61E+03	**5.37E+00**
f_{22}	Mean Error	2.04E+02+	1.19E+02+	5.33E+01+	4.25E+01+	9.37E+01+	1.42E+02+	3.96E+01+	1.45E+02+	5.47E+01+	**2.61E+01**
	Std Dev	1.01E+02	7.56E+01	5.05E+01	3.31E+01	6.42E+01	9.81E+01	1.65E+01	5.71E+01	4.98E+01	**4.46E+00**
f_{23}	Mean Error	**3.15E+02≈**	**3.15E+02≈**	**3.15E+02≈**	**3.15E+02≈**	**3.15E+02≈**	**3.15E+02≈**	**3.15E+02≈**	**3.15E+02≈**	**3.15E+02≈**	**3.15E+02**
	Std Dev	**0.00E+00**	**0.00E+00**	**0.00E+00**	**1.51E-13**	**0.00E+00**	**1.51E-13**	**0.00E+00**	**1.33E-10**	**5.78E-14**	**2.15E-13**
f_{24}	Mean Error	2.25E+02+	2.23E+02+	2.25E+02+	2.24E+02+	2.30E+02+	2.24E+02+	2.24E+02+	2.24E+02+	**2.22E+02≈**	**2.22E+02**
	Std Dev	2.33E+00	9.04E-01	1.20E+00	9.94E-01	6.11E+00	1.49E+00	6.45E-01	4.59E-01	**1.28E+00**	**1.67E-01**
f_{25}	Mean Error	2.08E+02+	**2.03E+02≈**	**2.03E+02≈**	**2.03E+02≈**	**2.03E+02≈**	**2.03E+02≈**	**2.03E+02≈**	**2.03E+02≈**	2.04E+02+	**2.03E+02**
	Std Dev	3.17E+00	**3.64E-01**	**4.96E-01**	**7.53E-02**	**4.94E-01**	**6.12E-01**	**5.17E-01**	**1.45E-01**	4.82E-01	**4.91E-02**
f_{26}	Mean Error	**1.00E+02≈**	**1.00E+02≈**	**1.00E+02≈**	**1.00E+02≈**	**1.00E+02≈**	**1.00E+02≈**	**1.00E+02≈**	**1.00E+02≈**	**1.00E+02≈**	**1.00E+02**
	Std Dev(Rank)	**1.78E-02**	**5.29E-02**	**3.95E-02**	**3.22E-02**	**5.09E-02**	**6.29E-02**	**4.04E-02**	**2.67E-02**	**2.98E-02**	**1.79E-02**
f_{27}	Mean Error	3.21E+02-	3.76E+02-	3.07E+02-	**3.00E+02-**	3.35E+02-	3.77E+02-	3.76E+02-	3.97E+02+	3.04E+02-	3.90E+02
	Std Dev	3.80E+01	4.39E+01	1.43E+01	**1.71E-13**	3.34E+01	4.05E+01	4.20E+01	1.79E+01	1.35E+01	3.06E+01
f_{28}	Mean Error	7.96E+02-	8.09E+02+	8.37E+02+	8.04E+02+	8.28E+02+	8.35E+02+	**7.85E+02-**	8.60E+02+	7.91E+02-	7.97E+02
	Std Dev	2.96E+01	2.32E+01	3.28E+01	2.09E+01	2.80E+01	3.23E+01	**1.79E+01**	2.53E+01	2.34E+01	1.63E+01
f_{29}	Mean Error	7.61E+02+	5.89E+02-	7.16E+02+	7.17E+02+	7.13E+02+	6.50E+02-	7.60E+02+	**4.00E+02-**	1.48E+03+	6.66E+02
	Std Dev	7.01E+01	2.33E+02	1.92E+00	3.37E+00	6.68E+01	1.59E+02	5.07E+01	**2.85E+02**	2.72E+02	1.50E+02
f_{30}	Mean Error	1.48E+03+	6.22E+02+	8.42E+02+	1.09E+03+	1.92E+03+	8.05E+02+	1.22E+03+	5.21E+02+	1.34E+03+	**5.14E+02**
	Std Dev	4.35E+02	1.37E+02	2.48E+02	4.14E+02	1.17E+03	2.90E+02	3.99E+02	1.14E+02	5.02E+02	**6.93E+01**
	+/-/≈	23/4/3	20/3/7	23/4/3	18/6/6	25/2/3	21/5/4	24/3/3	25/2/3	22/3/5	- -
	Rank	7.13	4.92	5.15	3.65	6.27	5.75	6.1	6.63	6.68	2.72

Table 9: Experimental results of NDE and nine up-to-date DE variants on CEC 2014 functions with $D = 50$

Function	Statistic	CIPDE	CoBiDE	JADE_sort	L-SHADE	SHADE	TSDE	dynNP-jDE	MPEDE	SinDE	NDE
f_1	Mean Error	1.73E+04-	2.98E+05+	2.69E+04-	**4.31E+02-**	1.74E+04-	1.11E+05+	2.81E+05+	1.08E+05+	2.89E+06+	6.30E+04
	Std Dev	8.76E+03	2.05E+05	1.47E+04	**5.83E+02**	1.68E+04	4.96E+04	1.09E+05	9.12E+04	1.17E+06	2.54E+04
f_2	Mean Error	7.57E-12-	1.09E-01+	9.44E-14-	**3.52E-14-**	8.75E-14-	2.50E+02+	5.62E-07+	1.43E+01+	3.17E+03+	3.31E-07
	Std Dev	3.51E-11	2.63E-01	2.56E-14	**1.24E-14**	4.01E-14	7.85E+02	2.31E-06	3.01E+01	3.20E+03	4.22E-07
f_3	Mean Error	1.82E+03+	6.99E-03+	**4.51E-08-**	2.25E+02+	1.98E+02+	1.66E+01+	1.67E-06+	4.71E-04+	4.13E+02+	2.03E-07
	Std Dev	1.58E+03	1.55E-02	**1.48E-07**	8.19E+02	9.88E+02	4.15E+01	4.59E-06	1.05E-03	3.44E+02	3.00E-07
f_4	Mean Error	1.35E+01+	4.27E+01+	1.77E+01+	2.51E+01+	1.98E+01+	1.99E+01+	9.01E+01+	6.61E+01+	9.54E+01+	**8.19E+00**
	Std Dev	2.90E+01	4.07E+01	4.22E+01	4.19E+01	4.00E+01	3.20E+01	1.27E+01	3.38E+01	4.03E+00	**6.55E-01**
f_5	Mean Error	2.08E+01+	2.02E+01-	**2.00E+01-**	2.04E+01+	2.03E+01≈	2.01E+01-	2.04E+01+	2.06E+01+	2.08E+01+	2.03E+01
	Std Dev	8.94E-02	3.29E-01	**1.09E-02**	4.19E-02	2.99E-02	9.72E-02	2.37E-02	3.48E-02	4.97E-02	4.57E-02
f_6	Mean Error	6.39E+00-	5.62E+00-	8.37E+00-	2.40E+01+	2.29E+01+	7.98E+00-	1.15E+01-	3.02E+01+	**1.95E-01-**	1.53E+01
	Std Dev	2.80E+00	3.18E+00	2.34E+00	1.58E+00	5.27E+00	2.97E+00	5.46E+00	2.14E+00	**4.16E-01**	2.44E+00
f_7	Mean Error	3.65E-03+	9.09E-15+	5.02E-03+	3.18E-14+	4.14E-03+	2.66E-03+	8.00E-13+	4.77E-03+	4.93E-14+	**0.00E+00**
	Std Dev	5.42E-03	3.15E-14	8.19E-03	5.21E-14	5.36E-03	4.71E-03	6.42E-13	4.62E-03	5.73E-14	**0.00E+00**
f_8	Mean Error	**0.00E+00-**	3.29E-10+	1.09E+01+	2.23E-13+	1.36E-13+	5.17E-01+	1.00E-13+	1.94E+01+	7.50E+00+	5.68E-14
	Std Dev	**0.00E+00**	1.26E-09	1.38E+01	6.95E-14	4.64E-14	7.11E-01	3.77E-14	1.34E+00	3.60E+00	5.78E-14
f_9	Mean Error	6.36E+01+	9.18E+01+	**2.64E+01-**	3.19E+01-	4.84E+01+	7.20E+01+	7.69E+01+	1.16E+02+	6.50E+01+	4.15E+01
	Std Dev	1.15E+01	1.68E+01	**3.33E+00**	5.05E+00	1.24E+01	2.09E+01	8.97E+00	9.93E+00	8.13E+00	6.53E+00
f_{10}	Mean Error	3.88E+02+	2.71E+02+	9.52E+02+	2.71E-01+	**4.50E-03-**	8.82E+00+	8.49E-03-	4.67E+02+	1.51E+02+	9.92E-02
	Std Dev	8.13E+01	4.83E+01	6.47E+02	1.89E-01	**7.97E-03**	3.33E+00	1.06E-02	5.23E+01	8.23E+01	2.36E-02
f_{11}	Mean Error	5.73E+03+	4.21E+03+	**3.49E+03-**	3.78E+03+	3.73E+03+	4.01E+03+	4.33E+03+	6.74E+03+	4.32E+03+	3.62E+03
	Std Dev	5.23E+02	9.14E+02	**3.71E+02**	3.27E+02	3.33E+02	5.76E+02	3.70E+02	3.12E+02	7.90E+02	4.24E+02
f_{12}	Mean Error	1.15E+00+	1.20E-01-	**7.95E-02-**	3.14E-01+	2.30E-01≈	1.06E-01-	3.64E-01+	7.42E-01+	1.35E+00+	2.30E-01
	Std Dev	1.12E-01	2.54E-01	**3.70E-02**	3.32E-02	3.32E-02	4.18E-02	4.54E-02	7.99E-02	1.40E-01	3.85E-02
f_{13}	Mean Error	1.87E-01+	3.57E-01+	2.45E-01+	2.35E-01+	3.29E-01+	3.34E-01+	3.40E-01+	3.10E-01+	3.43E-01+	**1.16E-01**
	Std Dev	4.07E-02	6.84E-02	4.15E-02	2.83E-02	5.26E-02	7.44E-02	5.41E-02	2.94E-02	3.59E-02	**1.67E-02**
f_{14}	Mean Error	3.56E-01+	2.84E-01+	3.52E-01+	2.84E-01+	3.15E-01+	2.89E-01+	3.05E-01+	2.80E-01+	2.81E-01+	**2.45E-01**
	Std Dev	3.03E-02	2.68E-02	5.39E-02	1.76E-02	8.47E-02	9.31E-02	2.79E-02	1.85E-02	9.84E-02	**3.11E-02**
f_{15}	Mean Error	9.07E+00+	6.05E+00+	6.19E+00+	6.04E+00+	8.12E+00+	6.86E+00+	1.02E+01+	1.33E+01+	7.99E+00+	**4.72E+00**
	Std Dev	2.85E+00	1.22E+00	8.02E-01	5.78E-01	1.35E+00	1.93E+00	9.86E-01	3.95E+00	1.46E+00	**6.11E-01**
f_{16}	Mean Error	1.72E+01+	1.83E+01+	1.74E+01+	1.78E+01+	1.81E+01+	1.82E+01+	1.77E+01+	1.92E+01+	2.00E+01+	**1.71E+01**
	Std Dev	1.16E+00	9.34E-01	7.55E-01	3.75E-01	4.95E-01	7.48E-01	3.96E-01	4.42E-01	4.14E-01	**5.61E-01**
f_{17}	Mean Error	2.68E+03+	1.06E+04+	1.86E+03+	1.41E+03+	2.21E+03+	1.32E+04+	1.23E+04+	9.45E+02+	3.59E+05+	**7.76E+02**
	Std Dev	1.03E+03	6.52E+03	1.09E+03	3.25E+02	4.11E+02	7.37E+03	7.65E+03	3.32E+02	1.98E+05	**1.94E+02**
f_{18}	Mean Error	1.43E+02+	8.44E+01+	1.14E+02+	1.04E+02+	1.72E+02+	1.98E+02+	2.68E+02+	4.33E+01+	3.10E+02+	**2.40E+01**
	Std Dev	3.04E+01	7.10E+01	3.81E+01	1.50E+01	4.87E+01	2.43E+02	4.65E+02	1.33E+01	3.63E+02	**5.41E+00**
f_{19}	Mean Error	1.57E+01+	6.90E+00-	9.51E+00+	9.44E+00+	1.32E+01+	**6.06E+00-**	1.07E+01+	1.01E+01+	9.33E+00+	8.40E+00
	Std Dev	7.57E+00	1.13E+00	2.18E+00	1.84E+00	3.17E+00	**1.11E+00**	9.05E-01	1.26E+00	7.75E-01	9.00E-01
f_{20}	Mean Error	3.49E+03+	3.33E+01+	5.71E+01+	**1.67E+01-**	1.82E+02+	1.55E+02+	3.40E+01+	4.08E+01+	2.14E+02+	2.24E+01
	Std Dev	4.20E+03	1.28E+01	2.67E+01	**6.26E+00**	1.07E+02	1.42E+02	1.01E+01	1.23E+01	1.41E+02	5.95E+00
f_{21}	Mean Error	1.51E+03+	3.35E+03+	6.84E+02+	5.08E+02+	1.24E+03+	3.97E+03+	2.45E+03+	5.88E+02+	2.25E+05+	**3.51E+02**
	Std Dev	4.28E+02	5.07E+03	1.58E+02	1.55E+02	3.69E+02	2.40E+03	1.54E+03	2.09E+02	1.18E+05	**9.42E+01**
f_{22}	Mean Error	6.33E+02+	5.43E+02+	2.84E+02+	2.36E+02+	4.02E+02+	6.43E+02+	4.12E+02+	5.44E+02+	2.49E+02+	**2.11E+02**
	Std Dev	2.45E+02	2.12E+02	1.17E+02	8.49E+01	1.74E+02	1.67E+02	1.31E+02	1.29E+02	1.25E+02	**1.34E+02**
f_{23}	Mean Error	**3.44E+02≈**	**3.44E+02≈**	**3.44E+02≈**	**3.44E+02≈**	**3.44E+02≈**	**3.44E+02≈**	**3.44E+02≈**	**3.44E+02≈**	**3.44E+02≈**	**3.44E+02**
	Std Dev	**5.80E-14**	**3.22E-13**	**3.09E-13**	**2.32E-13**	**3.01E-13**	**2.32E-13**	**2.64E-13**	**4.29E-11**	**2.89E-13**	**2.89E-13**
f_{24}	Mean Error	2.71E+02+	2.67E+02≈	2.75E+02+	2.75E+02+	2.79E+02+	2.71E+02+	2.66E+02-	2.71E+02+	**2.64E+02-**	2.67E+02
	Std Dev	1.46E+01	3.53E+00	1.77E+00	6.99E-01	2.98E+00	1.80E+00	2.08E+00	1.54E+00	**3.97E+00**	2.72E+00
f_{25}	Mean Error	2.21E+02+	2.07E+02+	2.18E+02+	**2.05E+02≈**	2.09E+02+	2.08E+02+	2.07E+02+	2.06E+02+	2.08E+02+	**2.05E+02**
	Std Dev	8.23E+00	1.07E+00	7.59E+00	**3.50E-01**	5.87E+00	4.20E+00	1.39E+00	9.67E-01	1.21E+00	**3.01E-01**
f_{26}	Mean Error	1.14E+02+	**1.00E+02≈**	1.16E+02+	**1.00E+02≈**	**1.00E+02≈**	1.12E+02+	**1.00E+02≈**	**1.00E+02≈**	1.04E+02+	**1.00E+02**
	Std Dev	3.34E+01	**6.21E-02**	3.73E+01	**1.86E-02**	**8.48E-02**	3.31E+01	**4.29E-02**	**2.61E-02**	1.82E+01	**2.95E-02**
f_{27}	Mean Error	4.51E+02+	4.06E+02+	4.91E+02+	3.74E+02+	7.13E+02+	5.51E+02+	4.35E+02+	3.47E+02-	**3.34E+02-**	3.50E+02
	Std Dev	5.05E+01	6.58E+01	7.46E+01	1.42E+02	1.42E+02	7.78E+01	8.15E+01	3.87E+01	**2.24E+01**	2.77E+01
f_{28}	Mean Error	1.14E+03+	1.14E+03+	1.20E+03+	1.11E+03≈	1.19E+03+	1.19E+03+	1.09E+03-	1.27E+03+	**1.06E+03-**	1.11E+03
	Std Dev	5.86E+01	6.01E+01	5.76E+01	2.71E+01	5.96E+01	6.96E+01	3.52E+01	5.23E+01	**6.01E+01**	3.07E+01
f_{29}	Mean Error	9.30E+02+	1.06E+03+	8.67E+02+	8.13E+02+	8.74E+02+	9.09E+02+	1.03E+03+	**6.59E+02-**	1.99E+03+	7.50E+02
	Std Dev	5.51E+01	2.07E+02	5.87E+01	4.96E+01	6.04E+01	1.09E+02	1.97E+02	**1.41E+02**	3.49E+02	5.63E+01
f_{30}	Mean Error	1.03E+04+	8.72E+03+	9.11E+03+	9.01E+03+	1.03E+04+	8.97E+03+	8.46E+03+	9.31E+03+	8.20E+03+	**8.16E+03**
	Std Dev	7.74E+02	5.09E+02	7.31E+02	7.38E+02	1.05E+03	4.88E+02	3.05E+02	7.39E+02	2.99E+02	**1.70E+02**
+/-/≈		25/4/1	23/4/3	21/8/1	22/4/4	23/3/4	25/4/1	24/4/2	26/2/2	25/4/1	- -
Rank		6.6	5.4	5.3	4.08	5.93	6.25	5.87	6.4	6.62	2.55



(a) $D = 30$ (b) $D = 50$

Figure 5: Statistical results of NDE and nine up-to-date DE variants on CEC 2014. (a) $D = 30$, (b) $D = 50$.

Table 10: Comparison results of NDE with nine up-to-date DE variants based on the multiproblem Wilcoxon signed-rank test on CEC2014 functions

$D = 30$					$D = 50$				
Algorithm	R+	R-	p-value	$\alpha = 0.05$	Algorithm	R+	R-	p-value	$\alpha = 0.05$
NDE vs CIPDE	333	45	0.0006	YES	NDE vs CIPDE	390	45	0.0002	YES
NDE vs CoBiDE	234	42	0.0037	YES	NDE vs CoBiDE	342	36	0.0002	YES
NDE vs JADE_sort	314	64	0.0028	YES	NDE vs JADE_sort	339.5	95.5	0.0086	YES
NDE vs L-SHADE	229	71	0.0249	YES	NDE vs L-SHADE	295	56	0.0025	YES
NDE vs SHADE	353	25	<0.0001	YES	NDE vs SHADE	318	33	0.0003	YES
NDE vs TSDE	292	59	0.0032	YES	NDE vs TSDE	407	28	<0.0001	YES
NDE vs dynNP-jDE	328	50	0.0009	YES	NDE vs dynNP-jDE	358	48	0.0004	YES
NDE vs MPEDE	338	40	0.0004	YES	NDE vs MPEDE	376	30	<0.0001	YES
NDE vs SinDE	283	42	0.0012	YES	NDE vs SinDE	381.5	53.5	0.0004	YES

Table 10, we see that NDE obtains higher R+ values than R- values in all cases, and there are significant differences at 0.05 significant level. The reason for these might be that the exploration and exploitation can be effectively balanced by the following two facts. 1) A more suitable mutation operator is chosen to each individual by employing its fitness value. 2) The neighborhood evolutionary dilemmas are alleviated by designing a dynamic neighborhood model and two exchanging operations. Therefore, NDE has better performance than nine up-to-date DE variants on these instances.

4.3.3. Comparison with six non-DE algorithms

Next, NDE is compared with six non-DE algorithms on 30 benchmark functions f_1-f_{30} in Table 1. These algorithms include CLPSO [19], GL-25 [13], DNLPSO [28], EPSO [25], HSOGA [15] and CMA-ES [14].

Table 11 reports their experimental results, the statistical results of Wilcoxon rank sum test and Friedman test when $D = 30$ and 50, and the last two rows summarize them.

When $D = 30$, from Table 11, the following detail results can be observed.

1) CMA-ES obtains the best results on unimodal functions f_1-f_3, and NDE onf_2 and f_3. This might be because the evolution path added in CMA-ES is helpful to improve

Table 11: Experimental results of NDE and six non-DE algorithms on CEC 2014 functions

Func	$D=30$ CLPSO Mean Error(Std Dev)	CMA-ES Mean Error(Std Dev)	GL-25 Mean Error(Std Dev)	NDLPSO Mean Error(Std Dev)	EPSO Mean Error(Std Dev)	HSOGA Mean Error(Std Dev)	NDE Mean Error(Std Dev)
f_1	9.15E+06(2.15E+06)+	**0.00E+00(1.74E-14)**-	1.08E+06(1.32E+06)+	2.18E+06(2.47E+06)+	2.16E+05(1.52E+05)+	8.87E+06(2.28E+06)+	5.91E+00(5.58E+00)
f_2	1.31E+02(3.14E+02)+	**0.00E+00(3.62E-14)**≈	1.16E+03(2.03E+03)+	7.28E+05(2.94E+06)+	2.04E+02(6.24E+02)+	1.17E+04(6.67E+03)+	**0.00E+00(0.00E+00)**
f_3	1.92E+02(2.17E+02)+	**0.00E+00(6.36E-14)**≈	3.15E-01(6.96E-01)+	6.28E+03(1.95E+04)+	6.05E+01(9.48E+01)+	3.75E+00(6.06E+00)+	**0.00E+00(0.00E+00)**
f_4	7.17E+01(1.68E+01)+	**0.00E+00(7.43E-14)**-	9.75E+01(1.34E+01)+	2.76E+01(2.78E+01)+	2.42E+01(3.54E+01)+	7.21E+01(1.40E+01)+	2.94E-08(4.84E-08)
f_5	2.04E+01(6.04E-02)+	**2.00E+01(2.66E-06)**-	2.10E+01(5.37E-02)+	2.08E+01(3.32E-01)+	2.05E+01(5.74E-02)+	2.03E+01(3.56E-01)+	2.01E+01(4.71E-02)
f_6	1.32E+01(1.01E+00)+	4.79E+01(7.85E+00)+	5.81E+00(4.11E+00)+	7.23E+00(6.00E+00)+	9.93E+00(2.04E+00)+	6.96E+00(2.71E+00)+	**3.37E+00(1.36E+00)**
f_7	1.09E-05(3.96E-05)+	1.77E-03(3.73E-03)+	9.00E-12(1.66E-11)+	2.69E-02(1.17E-01)+	2.86E-04(4.75E-04)+	2.27E-02(1.63E-02)+	**0.00E+00(0.00E+00)**
f_8	**0.00E+00(0.00E+00)**≈	4.08E+02(8.19E+01)+	2.24E+01(5.89E+00)+	4.37E+01(2.98E+01)+	3.90E-02(1.95E-01)+	5.96E-04(7.50E-04)+	**0.00E+00(0.00E+00)**
f_9	4.93E+01(8.35E+00)+	5.75E+02(1.39E+02)+	5.75E+01(5.92E+01)+	6.43E+01(3.39E+01)+	5.09E+01(1.05E+01)+	4.79E+01(1.37E+01)+	**2.48E+01(4.48E+00)**
f_{10}	3.20E+00(1.16E+00)+	5.12E+03(8.01E+02)+	7.25E+02(3.16E+02)+	2.37E+03(1.70E+03)+	1.34E+01(2.83E+01)	1.45E+00(1.18E+00)+	**0.00E+00(0.00E+00)**
f_{11}	2.18E+03(2.74E+02)+	5.17E+03(9.79E+02)+	5.92E+03(1.51E+03)+	4.02E+03(1.71E+03)+	2.08E+03(4.02E+02)+	2.36E+03(6.50E+02)+	**1.27E+03(2.41E+02)**
f_{12}	3.84E-01(7.28E-02)+	4.25E-01(6.40E-01)+	2.44E+00(3.67E-01)+	1.52E+00(9.95E-01)+	4.62E-01(9.08E-02)+	2.18E-01(2.92E-01)+	**1.22E-01(2.82E-02)**
f_{13}	3.25E-01(4.79E-02)+	2.63E-01(8.79E-02)+	2.63E-01(3.76E-02)+	3.17E-01(1.10E-01)+	3.06E-01(5.86E-02)+	2.70E-01(5.50E-02)+	**6.80E-02(1.31E-02)**
f_{14}	2.52E-01(2.56E-02)+	3.66E-01(7.55E-02)+	2.59E-01(3.35E-02)+	6.67E-01(2.06E+00)+	2.43E-01(4.27E-02)+	2.87E-01(3.05E-02)+	**2.03E-01(2.64E-02)**
f_{15}	7.37E+00(9.44E-01)+	3.83E+00(1.07E+00)+	1.30E+01(4.77E+00)+	5.61E+00(3.07E+00)+	5.61E+00(2.21E+00)+	5.98E+00(2.36E+00)+	**2.60E+00(4.45E-01)**
f_{16}	1.03E+01(3.59E-01)+	1.43E+01(4.16E-01)+	1.18E+01(2.90E-01)+	1.16E+01(9.10E-01)+	1.04E+01(5.34E-01)+	9.87E+00(7.15E-01)+	**8.38E+00(4.13E-01)**
f_{17}	7.71E+05(2.74E+05)+	1.84E+03(4.63E+02)+	1.99E+05(1.06E+05)+	2.81E+05(6.20E+05)+	6.12E+04(4.73E+04)+	8.81E+04(5.78E+04)+	**1.13E+02(5.94E+01)**
f_{18}	1.11E+02(5.12E+01)+	1.56E+02(3.78E+01)+	2.37E+02(3.21E+02)+	1.28E+04(1.77E+04)+	2.84E+02(2.78E+02)+	1.43E+03(1.36E+03)+	**5.95E+00(1.50E+00)**
f_{19}	7.47E+00(4.82E-01)+	1.02E+01(1.53E+00)+	4.71E+00(6.23E-01)+	1.03E+01(5.08E+00)+	7.33E+00(1.12E+00)+	8.85E+00(1.55E+00)+	**2.14E+00(4.61E-01)**
f_{20}	3.23E+03(1.62E+03)+	2.84E+02(1.22E+02)+	1.94E+02(1.23E+02)+	4.71E+03(1.64E+04)+	1.01E+03(1.11E+03)+	1.16E+02(7.29E+01)+	**4.05E+00(9.50E-01)**
f_{21}	8.47E+04(5.47E+04)+	9.82E+02(3.16E+02)+	5.59E+04(2.45E+04)+	9.56E+04(9.09E+04)+	3.53E+04(3.00E+04)+	1.63E+04(1.59E+04)+	**1.01E+01(5.37E+00)**
f_{22}	2.02E+02(7.25E+01)+	2.27E+02(1.40E+02)+	1.52E+02(6.09E+01)+	3.25E+02(2.09E+02)+	2.75E+02(1.05E+02)+	2.69E+02(1.16E+02)+	**2.61E+01(4.46E+00)**
f_{23}	3.15E+02(3.72E-06)≈	3.15E+02(3.93E-12)≈	3.15E+02(1.33E-09)≈	3.14E+02(5.26E-01)-	3.15E+02(2.21E-12)≈	**2.07E+02(2.71E+01)**-	3.15E+02(2.15E-13)
f_{24}	2.23E+02(4.72E+00)+	2.41E+02(4.69E+01)+	2.22E+02(5.99E-01)≈	2.34E+02(8.84E+00)+	2.26E+02(1.26E+00)+	**2.00E+02(1.31E-01)**-	2.22E+02(1.67E-01)
f_{25}	2.08E+02(8.30E-01)+	2.04E+02(2.44E+00)+	2.07E+02(1.95E+00)+	2.01E+02(1.45E+00)-	2.08E+02(1.77E+00)+	**2.00E+02(1.18E-03)**-	2.03E+02(4.91E-02)
f_{26}	**1.00E+02(9.15E-02)**≈	1.13E+02(4.60E+01)+	**1.00E+02(4.64E-02)**≈	1.02E+02(1.40E+01)+	1.06E+02(2.37E+01)+	**1.00E+02(4.21E-02)**≈	**1.00E+02(1.79E-02)**
f_{27}	4.13E+02(5.77E+00)+	4.07E+02(1.41E+02)+	**3.02E+02(8.78E-01)**-	4.76E+02(1.17E+02)+	4.12E+02(3.62E+01)+	4.16E+02(3.52E+01)+	3.90E+02(3.06E+01)
f_{28}	9.20E+02(5.66E+01)+	3.69E+03(2.68E+03)+	8.80E+02(3.04E+01)+	4.18E+02(5.22E+01)-	9.53E+02(7.67E+01)+	**2.26E+02(6.96E+01)**-	7.97E+02(1.63E+01)
f_{29}	1.01E+03(9.84E+01)+	8.14E+02(9.54E+01)+	1.01E+03(1.03E+02)+	2.12E+02(1.25E+01)-	9.80E+02(1.28E+02)+	**2.11E+02(2.81E+00)**-	6.66E+02(1.50E+02)
f_{30}	3.69E+03(9.60E+02)+	2.29E+03(6.05E+02)+	1.33E+03(2.74E+02)+	9.19E+02(2.96E+02)+	2.38E+03(5.55E+02)+	**3.70E+02(1.57E+02)**-	5.14E+02(6.93E+01)
+/-/≈	27/0/3	24/3/3	27/1/2	26/4/0	29/0/1	23/6/1	- -
Rank	4.53	4.32	4.33	5.35	4.4	3.42	1.65

Func	$D=50$ CLPSO Mean Error(Std Dev)	CMA-ES Mean Error(Std Dev)	GL-25 Mean Error(Std Dev)	NDLPSO Mean Error(Std Dev)	EPSO Mean Error(Std Dev)	HSOGA Mean Error(Std Dev)	NDE Mean Error(Std Dev)
f_1	1.60E+07(3.36E+06)+	**4.43E-14(1.55E-14)**-	2.11E+06(9.69E+05)+	1.25E+07(2.45E+07)+	2.16E+05(1.52E+05)+	9.27E+06(2.74E+06)+	6.30E+04(2.54E+04)
f_2	6.85E+01(1.38E+02)+	**8.75E-14(2.00E-14)**-	2.60E+03(1.30E+03)+	8.29E+08(3.60E+09)+	2.04E+02(6.24E+02)+	1.42E+06(1.15E+06)+	3.31E-07(4.22E-07)
f_3	2.19E+03(7.74E+02)+	**1.75E-13(7.32E-14)**-	3.02E+02(3.20E+02)+	1.18E+04(1.26E+04)+	6.05E+01(9.48E+01)+	5.60E+03(1.61E+03)+	2.03E-07(3.00E-07)
f_4	9.52E+01(1.11E+01)+	1.57E+01(3.68E+01)+	9.56E+01(1.35E+00)+	2.24E+02(8.08E+02)+	2.42E+01(3.54E+01)+	1.84E+02(3.44E+01)+	**8.19E+00(6.55E-01)**
f_5	2.05E+01(4.75E-02)+	**2.00E+01(1.75E-06)**-	2.11E+01(3.20E-02)+	2.10E+01(3.99E-01)+	2.05E+01(5.74E-02)+	2.01E+01(2.57E-01)-	2.03E+01(4.57E-02)
f_6	2.89E+01(2.69E+00)+	7.50E+01(9.95E+00)+	**2.95E+00(1.99E+00)**-	1.93E+01(8.19E+00)+	9.93E+00(2.04E+00)-	1.89E+01(4.49E+00)+	1.53E+01(2.44E+00)
f_7	1.45E-04(1.69E-04)+	6.90E-04(2.41E-03)+	1.14E-09(1.67E-09)+	3.32E+00(1.35E+01)+	2.86E-04(4.75E-04)+	8.29E-01(2.53E-01)+	**0.00E+00(0.00E+00)**
f_8	1.46E-13(5.21E-14)+	6.96E+02(1.14E+02)+	5.13E+01(1.56E+01)+	1.10E+02(6.60E+01)+	3.90E-02(1.95E-01)+	1.29E-01(3.23E-01)+	**5.68E-14(5.78E-14)**
f_9	1.23E+02(1.63E+01)+	1.22E+03(1.97E+02)+	1.26E+02(1.14E+02)+	1.44E+02(8.22E+01)+	5.09E+01(1.05E+01)+	9.44E+01(1.88E+01)+	**4.15E+01(6.53E+00)**
f_{10}	7.06E+00(2.50E+00)+	8.26E+03(7.11E+02)+	2.58E+03(9.78E+02)+	5.14E+03(3.01E+03)+	1.34E+01(2.83E+01)+	2.89E+00(1.29E+00)+	**9.92E-02(2.36E-02)**
f_{11}	4.96E+03(4.16E+02)+	8.42E+03(1.12E+03)+	1.27E+04(3.05E+02)+	7.94E+03(3.34E+03)+	**2.08E+03(4.02E+02)**-	5.26E+03(7.01E+02)+	3.62E+03(4.24E+02)
f_{12}	4.13E-01(7.59E-02)+	2.21E-01(1.52E-01)-	3.39E+00(2.81E-01)+	2.45E+00(1.38E+00)+	4.62E-01(9.08E-02)+	**1.45E-01(4.26E-02)**-	2.30E-01(3.85E-02)
f_{13}	4.17E-01(3.42E-02)+	3.69E-01(7.60E-02)+	4.21E-01(5.16E-02)+	4.96E-01(1.39E-01)+	3.06E-01(5.86E-02)+	4.07E-01(3.20E-02)+	**1.16E-01(1.67E-02)**
f_{14}	2.93E-01(3.39E-02)+	5.06E-01(3.06E-01)+	3.06E-01(3.82E-02)+	2.94E+00(1.04E+01)+	**2.43E-01(4.27E-02)**-	3.34E-01(2.09E-02)+	2.45E-01(3.11E-02)
f_{15}	1.73E+01(1.63E+00)+	6.02E+00(1.48E+00)+	2.09E+01(1.10E+01)+	8.59E+01(3.61E+02)+	5.61E+00(2.21E+00)+	2.27E+01(8.24E+00)+	**4.72E+00(6.11E-01)**
f_{16}	1.90E+01(4.24E-01)+	2.37E+01(5.48E-01)+	2.15E+01(2.72E-01)+	2.09E+01(9.79E-01)+	**1.04E+01(5.34E-01)**-	1.86E+01(9.32E-01)+	1.71E+01(5.61E-01)
f_{17}	2.67E+06(9.10E+05)+	2.74E+03(6.51E+02)+	5.14E+05(2.31E+05)+	5.62E+05(4.44E+05)+	6.12E+04(4.73E+04)+	1.72E+06(7.31E+05)+	**7.76E+02(1.94E+02)**
f_{18}	1.92E+02(6.99E+01)+	2.49E+02(5.86E+01)+	6.10E+02(3.72E+02)+	1.28E+04(2.03E+04)+	2.84E+02(2.78E+02)+	4.30E+02(4.29E+02)+	**2.40E+01(5.41E+00)**
f_{19}	1.70E+01(2.17E+00)+	1.92E+01(2.78E+00)+	3.49E+01(1.28E+00)+	2.12E+01(1.18E+01)+	**7.33E+00(1.12E+00)**-	3.49E+01(2.12E+01)+	8.40E+00(9.00E-01)
f_{20}	7.06E+03(2.74E+03)+	4.61E+02(1.08E+02)+	3.97E+02(1.68E+02)+	3.91E+03(6.18E+03)+	1.01E+03(1.11E+03)+	5.79E+02(2.85E+02)+	**2.24E+01(5.95E+00)**
f_{21}	1.55E+06(7.42E+05)+	1.71E+03(3.79E+02)+	3.19E+05(1.08E+05)+	3.81E+05(5.90E+05)+	3.53E+04(3.00E+04)+	3.52E+05(1.72E+05)+	**3.51E+02(9.42E+01)**
f_{22}	6.43E+02(1.02E+02)+	4.51E+02(2.83E+02)+	5.44E+02(3.39E+02)+	8.29E+02(3.56E+02)+	2.75E+02(1.05E+02)+	6.56E+02(2.72E+02)+	**2.11E+02(1.34E+02)**
f_{23}	3.44E+02(1.61E-06)≈	3.44E+02(2.77E-05)≈	3.44E+02(2.90E-09)≈	3.43E+02(2.90E+01)-	3.15E+02(2.21E-12)-	**2.00E+02(1.37E-01)**-	3.44E+02(2.89E-13)
f_{24}	2.58E+02(2.50E+00)-	3.20E+02(2.04E+02)+	2.58E+02(4.18E+00)-	2.80E+02(1.69E+01)+	2.26E+02(1.26E+00)-	**2.01E+02(2.58E-01)**-	2.67E+02(2.72E+00)
f_{25}	2.16E+02(9.75E-01)+	2.05E+02(1.78E+00)≈	2.19E+02(3.41E+00)+	2.04E+02(1.01E+01)-	2.08E+02(1.77E+00)+	**2.00E+02(2.20E-03)**-	2.05E+02(3.01E-01)
f_{26}	1.01E+02(7.37E-02)+	**1.00E+02(8.76E-02)**≈	1.12E+02(3.28E+01)+	1.30E+02(8.04E+01)+	1.06E+02(2.37E+01)+	**1.00E+02(5.88E-02)**≈	**1.00E+02(2.95E-02)**
f_{27}	7.76E+02(3.03E+02)+	4.92E+02(5.85E+01)+	**3.28E+02(2.09E+01)**-	8.65E+02(3.15E+02)+	4.12E+02(3.62E+01)+	7.23E+02(7.32E+01)+	3.50E+02(2.77E+01)
f_{28}	1.47E+03(9.75E+01)+	5.63E+03(4.27E+03)+	1.27E+03(5.86E+01)+	4.28E+02(1.07E+02)-	9.53E+02(7.67E+01)-	**2.17E+02(5.74E+01)**-	1.11E+03(3.07E+01)
f_{29}	1.71E+03(2.46E+02)+	8.62E+02(7.27E+01)+	1.40E+03(1.24E+02)+	2.34E+02(3.45E+01)-	9.80E+02(1.28E+02)+	**2.23E+02(2.57E+00)**-	7.50E+02(5.63E+01)
f_{30}	1.04E+04(1.09E+03)+	8.95E+03(6.01E+02)+	9.84E+03(2.46E+02)+	1.45E+03(3.94E+02)-	2.38E+03(5.55E+02)-	**1.26E+03(3.47E+02)**-	8.16E+03(1.70E+02)
+/-/≈	28/1/1	22/5/3	26/3/1	25/5/0	21/9/0	21/8/1	- -
Rank	4.68	4.03	4.82	5.57	3.02	3.75	2.13

the quality of evaluation.

2) NDE obtains the best results on simple multimodal and hybrid functions f_6-f_{22}, CMA-ES on f_4 and f_5, and CLPSO on f_8.

3) HSOGA gets the best results on composition functions f_{23}-f_{26} and f_{28}-f_{30}, and GL-25 on f_{27}. This might be because the self-adaptive orthogonal crossover operator in HSOGA can effectively maintain the population diversity and enhance the exploitation of promising regions by using a representative set of points as the potential offspring and a local search scheme.

According to the statistical results in Table 11, a) NDE performs better than CLPSO, CMA-ES, GL-25, NDLPSO, EPSO and HSOGA on 27, 24, 27, 26, 29 and 23 test functions respectively, slightly worse on 0, 3, 1, 4, 0 and 6 test functions respectively, similar to that on 3, 3, 2, 0, 1 and 1 test functions, respectively; and b) they get 1.65, 4.53, 4.32, 4.33, 5.35, 4.4 and 3.42 in term of overall performance ranking on all problems, respectively.

When $D = 50$, from Table 11, we see that NDE obtains the best results on f_4, f_7-f_{10}, f_{13}, f_{15}, f_{17}, f_{18}, f_{20}-f_{22} and f_{26}, CMA-ES on f_1-f_3, f_5 and f_{26}, EPSO on f_{11}, f_{14}, f_{16} and f_{19}, HSOGA on f_{12}, f_{23}-f_{26} and f_{28}-f_{30}, and GL-25 on f_6 and f_{27}. According to the statistical results in Table 11, a) NDE performs better than CLPSO, CMA-ES, GL-25, NDLPSO, EPSO and HSOGA on 28, 22, 26, 25, 21 and 21 test functions respectively, slightly worse on 1, 5, 3, 5, 9 and 8 test functions respectively, similar to that on 1, 3, 1, 0, 0 and 1 test functions, respectively; and b) they get 2.13, 4.68, 4.03, 4.82, 5.57, 3.02 and 3.75 in term of overall performance ranking on all problems, respectively.

For clarity, Figure 6 depicts s the bar charts of the statistical results of NDE and these six compared algorithms on all functions from CEC 2014 with $D = 30$ and 50, where the blue and red bars are same as Figure 4. From Figure 6, we see that NDE has the best rank and the most number of best results for all functions.

Furthermore, Table 12 provides the comparison results of NDE with others on all problems based on the multiproblem Wilcoxon signed-rank test when $D = 30$ and 50. From Table 12, we see that NDE gets higher R+ values than R- values in all cases, and there are significant differences at 0.05 significant level except for EPSO when $D = 50$. These might be because NM strategy suitably chooses a more promising mutation operator for each individual based on its fitness value, and NAE mechanism alleviates the evolutionary dilemmas. Therefore, NDE has better performance than six non-DE algorithms on these instances.

In summary, it should be noted that it is just the proposed strategy and mechanism that make NDE superior to other algorithms on these functions, especially for multimodal

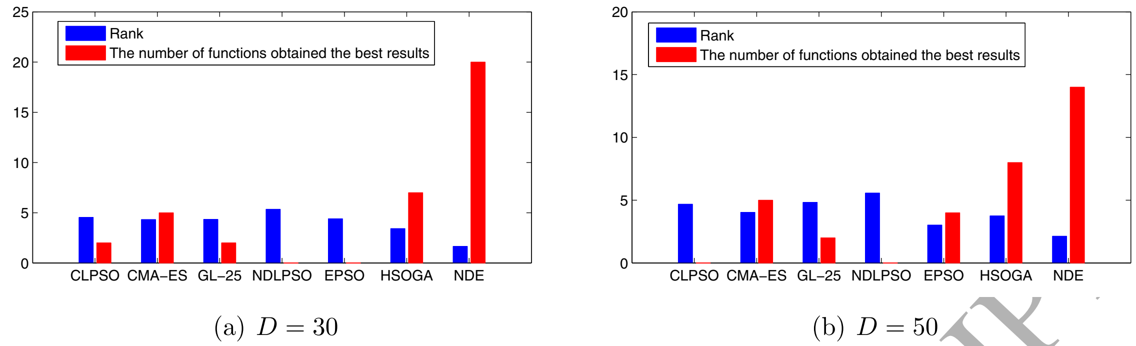


Figure 6: Statistical results of NDE and six non-DE algorithms on CEC 2014. (a) $D = 30$, (b) $D = 50$.

Table 12: Comparison results of NDE with six non-DE variants based on the multiproblem Wilcoxon signed-rank test on CEC2014 functions

$D = 30$					$D = 50$				
Algorithm	R+	R-	p-value	$\alpha = 0.05$	Algorithm	R+	R-	p-value	$\alpha = 0.05$
NDE vs CLPSO	378	0	<0.0001	YES	NDE vs CLPSO	424	11	<0.0001	YES
NDE vs CMA-ES	365	13	<0.0001	YES	NDE vs CMA-ES	337	41	0.0004	YES
NDE vs GL-25	363	15	<0.0001	YES	NDE vs GL-25	407	28	<0.0001	YES
NDE vs NDLPSO	412.5	52.5	0.0002	YES	NDE vs NDLPSO	396	69	0.0008	YES
NDE vs EPSO	435	0	<0.0001	YES	NDE vs EPSO	326	139	0.0558	NO
NDE vs HSOGA	329	106	0.0164	YES	NDE vs HSOGA	324	111	0.0219	YES

and hybrid functions. In fact, the worse or better individuals employ an explorative or exploitative mutation operator to adjust their search regions in NM strategy. Meanwhile, NAE mechanism alleviates the neighborhood evolutionary dilemmas of each individual to improve the search performance. Thus, NDE effectively maintains a suitable balance between exploration and exploitation, and is a more promising algorithm.

4.3.4. The reliability of NDE

Another important factor to evaluate the performance of an algorithm is reliability, *i.e.*, the experimental results of the algorithm vary slightly as the number of runs increases. To measure the reliability of NDE, it is further independently run with 1000 times on 30 benchmark functions f_1-f_{30} in Table 1 when $D = 30$ and 50.

Table 13 reports its experimental results obtained by 1000 independent runs on all problems, and also lists those by 30 independent runs for the convenience of comparison. From Table 13, we see that there is only a slight variation in the experimental results of NDE on each function for different running times whether $D = 30$ or 50. In particular, the difference between the experimental results of 30 and 1000 independent runs is the same or no more than one order of magnitude for each function. In fact, the numerical results obtained by 1000 independent runs are same and slightly worse than those by 30

Table 13: Experimental results of NDE obtained by 30 and 1000 independent runs

Function	$D = 30$ Mean Error(Std Dev) 30 runs	$D = 30$ Mean Error(Std Dev) 1000 runs	$D = 50$ Mean Error(Std Dev) 30 runs	$D = 50$ Mean Error(Std Dev) 1000 runs
f_1	5.91E+00(5.58E+00)	2.18E+01(3.07E+01)	6.30E+04(2.54E+04)	5.94E+04(2.25E+04)
f_2	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)	3.31E-07(4.22E-07)	6.78E-07(8.62E-07)
f_3	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)	2.03E-07(3.00E-07)	7.69E-07(1.20E-06)
f_4	2.94E-08(4.84E-08)	8.54E-08(2.07E-07)	8.19E+00(6.55E-01)	2.27E+01(3.19E+01)
f_5	2.01E+01(4.71E-02)	2.01E+01(4.86E-02)	2.03E+01(4.57E-02)	2.03E+01(5.57E-02)
f_6	3.37E+00(1.36E+00)	3.65E+00(1.36E+00)	1.53E+01(2.44E+00)	1.56E+01(2.64E+00)
f_7	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)
f_8	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)	5.68E-14(5.78E-14)	7.84E-14(5.26E-14)
f_9	2.48E+01(4.48E+00)	2.51E+01(5.43E+00)	4.15E+01(6.53E+00)	3.94E+01(8.27E+00)
f_{10}	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)	9.92E-02(2.36E-02)	1.06E-01(2.79E-02)
f_{11}	1.27E+03(2.41E+02)	1.32E+03(2.93E+02)	3.62E+03(4.24E+02)	3.70E+03(4.90E+02)
f_{12}	1.22E-01(2.82E-02)	1.47E-01(3.99E-02)	2.30E-01(3.85E-02)	2.31E-01(5.60E-02)
f_{13}	6.80E-02(1.31E-02)	7.76E-02(1.74E-02)	1.16E-01(1.67E-02)	1.24E-01(2.11E-02)
f_{14}	2.03E-01(2.64E-02)	2.11E-01(3.08E-02)	2.45E-01(3.11E-02)	2.57E-01(3.37E-02)
f_{15}	2.60E+00(4.45E-01)	2.70E+00(4.89E-01)	4.72E+00(6.11E-01)	4.99E+00(7.25E-01)
f_{16}	8.38E+00(4.13E-01)	8.42E+00(5.42E-01)	1.71E+01(5.61E-01)	1.73E+01(5.94E-01)
f_{17}	1.13E+02(5.94E+01)	1.14E+02(5.95E+01)	7.76E+02(1.94E+02)	7.61E+02(2.20E+02)
f_{18}	5.95E+00(1.50E+00)	6.62E+00(1.86E+00)	2.40E+01(5.41E+00)	2.58E+01(7.17E+00)
f_{19}	2.14E+00(4.61E-01)	2.29E+00(5.03E-01)	8.40E+00(9.00E-01)	8.68E+00(8.79E-01)
f_{20}	4.05E+00(9.50E-01)	4.86E+00(1.28E+00)	2.24E+01(5.95E+00)	2.47E+01(6.26E+00)
f_{21}	1.01E+01(5.37E+00)	1.32E+01(1.05E+01)	3.51E+02(9.42E+01)	3.72E+02(1.13E+02)
f_{22}	2.61E+01(4.46E+00)	3.84E+01(3.06E+01)	2.11E+02(1.34E+02)	2.42E+02(1.68E+02)
f_{23}	3.15E+02(2.15E-13)	3.15E+02(2.04E-12)	3.44E+02(2.89E-13)	3.44E+02(3.98E-13)
f_{24}	2.22E+02(1.67E-01)	2.22E+02(4.19E+00)	2.67E+02(2.72E+00)	2.67E+02(2.06E+00)
f_{25}	2.03E+02(4.91E-02)	2.03E+02(5.98E-02)	2.05E+02(3.01E-01)	2.05E+02(3.29E-01)
f_{26}	1.00E+02(1.79E-02)	1.00E+02(2.16E-02)	1.00E+02(2.95E-02)	1.00E+02(3.23E-02)
f_{27}	3.90E+02(3.06E+01)	3.94E+02(2.48E+01)	3.50E+02(2.77E+01)	3.59E+02(2.83E+01)
f_{28}	7.97E+02(1.63E+01)	8.05E+02(1.85E+01)	1.11E+03(3.07E+01)	1.11E+03(2.87E+01)
f_{29}	6.66E+02(1.50E+02)	6.74E+02(1.39E+02)	7.50E+02(5.63E+01)	7.67E+02(4.08E+01)
f_{30}	5.14E+02(6.93E+01)	5.33E+02(9.78E+01)	8.16E+03(1.70E+02)	8.42E+03(3.22E+02)

independent runs on 10 and 20 test functions with $D = 30$, respectively. Meanwhile, they are same, slightly worse and better than those by 30 independent runs on 7, 20 and 3 test functions with $D = 50$, respectively. This might be due to the computational errors and some worse cases with very small probabilities in a large number of numerical experiments. Thus, NDE is robust and reliable.

4.4. Algorithm efficiency

To show the efficiency of NDE, we compare it with the classical DE, EPSDE and SaDE on 5 typical functions including unimodal functions f_1-f_3, and simple multimodal functions f_6 and f_9 in Table 1 when $D = 30$. The classical DE employs the DE/rand/1 and binomial crossover operation, the scaling factor and crossover rate are set to 0.5. In this experiment, the average CPU time of 30 independent runs is recorded to evaluate their efficiencies. Table 14 reports the average CPU times of 30 independent runs expended by them.

From Table 14, we see that NDE is slower than DE and EPSDE, and similar to SaDE. Unlike the classical DE and EPSDE, NDE requires to sort the neighbors of each individual at each generation and to calculate the diversity of all neighborhoods based on fitness values. Then it takes a longer time than the classical DE and EPSDE. Overall, numerical results show that NDE is a promising algorithm.

Table 14: Average CPU time expended by NDE, DE, EPSDE and SaDE.

Function	unimodal			multimodal	
	f_1	f_2	f_3	f_6	f_9
DE	19.00 s	18.44 s	19.27 s	54.64 s	18.50 s
EPSDE	24.39 s	22.36 s	25.71 s	60.31 s	23.10 s
SaDE	56.00 s	54.37 s	57.44 s	88.69 s	54.80 s
NDE	59.10 s	57.08 s	59.38 s	96.69 s	57.28 s

Table 15: Numerical and statistic results of NDE and five DE variants on PEFM

Function	Best(Result)	Worst(Result)	Average value	Standard deviation	p-value	$\alpha = 0.05$
CoDE	**0.00E+00**	3.91E-12	3.92E-12	1.24E-11	0.0482	YES
jDE	3.06E+00	1.25E+01	7.37E+00	3.01E+00	<0.0001	YES
JADE	3.17E-02	1.82E+00	6.70E-01	5.43E-01	0.0024	YES
EPSDE	3.76E+00	1.29E+01	1.00E+01	2.50E+00	<0.0001	YES
SaDE	**0.00E+00**	6.61E+00	9.12E-01	2.11E+00	0.0019	YES
NDE	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	- -	- -

4.5. Application

As an application, we consider the Parameter Estimation for Frequency-Modulated Sound Waves (PEFM) [9]. It has an important role in several modern music systems, aims to generate a sound similar to target sound and can be modeled as the following optimization problem

$$\min \; f(\vec{X}) = \sum_{t=0}^{100} (y(t) - y_0(t))^2, \tag{23}$$

where $\vec{X} = (a_1, \omega_1, a_2, \omega_2, a_3, \omega_3)$,

$$y(t) = a_1 \sin(\omega_1 t\theta + a_2 \sin(\omega_2 t\theta + a_3 \sin(\omega_3 t\theta))),$$

and

$$y_0(t) = \sin(5t\theta + 1.5 \sin(4.8t\theta + 2 \sin(4.9t\theta))).$$

Clearly, this problem is highly complex and multimodal, and its minimum value is 0.

To show the effectiveness of NDE, we compare it with five state-of-the-art DE variants CoDE, jDE, JADE, EPSDE and SaDE on this problem. Let $FES_{max} = 60000$, Table 15 reports their numerical results by 30 independent runs, and the statistic results of Wilcoxon rank sum test at 0.05 significant level. From Table 15, we see that NDE gets the best performance among them, and the significant differences between NDE and others can be observed in all cases. Thus, NDE is more effective for this problem.

5. Conclusion

To make full use of the characteristics of individuals and the evolutionary states of the neighborhood, this paper proposes a novel differential evolution with NAE mechanism. A NM strategy is first designed to adjust suitably the search ability of each individual by developing two NM operators with different search characteristics and choosing a suitable one for each individual according to its fitness value. Then a NAE mechanism is presented to identify and mitigate the evolutionary dilemmas of the neighborhood by tracking its fitness value and diversity and designing a dynamic neighborhood model and two exchanging operations, respectively. Meanwhile, a simple reduction method is employed to adjust the population size dynamically. Compared with the DE variants based on neighborhood and evolutionary state, the proposed algorithm not only chooses a more suitable mutation operator for each individual, but also relieves adaptively the neighborhood evolutionary dilemmas of each individual. Thus, NDE not only suitably adjusts the search performance of each individual, but also effectively maintains a proper balance between exploration and exploitation. Finally, the proposed algorithm is compared with 21 typical algorithms by numerical experiments on 30 benchmark functions from CEC2014, and applied to the Parameter Estimation for Frequency-Modulated Sound Waves. Experimental results show that the proposed algorithm is reliable and has better performance.

Further research can be focused on extending the NAE mechanism to other algorithms, designing adaptive hybrid neighborhood topology to further enhance the performance of DE, and applying NDE to practical problems.

Acknowledgments

We are very grateful to the Editor in Chief, Associate Editor and five anonymous referees for their valuable comments and suggestions on earlier versions of this paper. The research was supported in part by the National Natural Science Foundation of China No. 61273311 and 61502290, and by the Fundamental Research Funds For the Central Universities No. 2017TS002.

References

[1] M.Z. Ali, N.H. Awad, P.N. Suganthan, R.G. Reynolds, An adaptive multipopulation differential evolution with dynamic population reduction, IEEE Trans. Cyber. 99 (2016) 1-12.

[2] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (6) (2006) 646-657.

[3] J. Brest, M.S. Maucec, Population size reduction for the differential evolution algorithm, Appl. Intell. 29 (3) (2008) 228-247.

[4] Y. Cai, J. Wang, Differential evolution with neighborhood and direction information numerical optimization, IEEE Trans. Cyber. 43 (6) (2013) 2202-2215.

[5] Y.Q. Cai, M. Zhao, J.L. Liao, T. Wang, H. Tian, Y.H. Chen, Neighborhood guided differential evolution, Soft Comput. 21 (16) (2016) 1-44.

[6] M. Črepinšek, S.H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: a survey, ACM Comput. Surveys (CSUR) 45 (3) (2013) 1-33.

[7] R. Das, B. Akay, R.K. Singla, K. Singh, Application of artificial bee colony algorithm for inverse modelling of a solar collector, Inverse Probl. Sci. Eng., 2017, pp. 1-22.

[8] R. Das, K. Singh, B. Akay, T.K Gogoi, Application of artificial bee colony algorithm for maximizing heat transfer in a perforated fin, In Proc. Inst. Mech. Eng. 232 (1) (2016) 1989-1996.

[9] S. Das, P.N. Suganthan, Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization roblems, Technical report, Jadavpur University, Kolkata, and Nanyang Technological University, Singapore, 2010.

[10] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2011) 4-31.

[11] J. Derrac, S. Garca, 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.

[12] A. Draa, S. Bouzoubia, I. Boukhalfa, A sinusoidal differential evolution algorithm for numerical optimization, Appl. Soft Comput. 27 (2015) 99-126.

[13] C. Garcia-Martinez, M. Lozano, F. Herrera, D. Molina, A.M. Sanchez, Global and local real-coded genetic algorithm based on parent-centric crossover operators, Eur. J. Oper. Res. 185 (3) (2008) 1088-1113.

[14] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evol. Comput. 9 (2) (2001) 159-195.

[15] Z.Y. Jiang, Z.X. Cai, Y. Wang, Hybrid self-adaptive orthogonal genetic algorithm for solving global optimization problems, J. Softw. 21 (6) (2010) 1296-1307.

[16] R. Karimpour, G. Ruhe, Evolutionary robust optimization for software product line scoping: an explorative study, Computer Languages, Systems and Structures 47 (2) (2017) 189-210.

[17] T. Karmaker, R. Das, Estimation of riverbank soil erodibility parameters using genetic algorithm, Sādhanā 42 (11) (2017) 1-11.

[18] Y. Lee, J.J. Filliben, R.J. Micheals, P.J. Phillips, Sensitivity analysis for biometric system: a methododology based on orthogonal experiment designs, Computer Vision and Image Understanding 117 (2013) 532-550.

[19] J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (3) (2006) 281-295.

[20] J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization, Technical report, Nanyang Technological University, China: Zhenzhou University, Singapore, 2013.

[21] 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.

[22] S.H. Liu, M. Mernik, D. Hrnčič, M. Črepinšek, A parameter control method of evolutionary algorithms using exploration and exploitation measures with a practical application for fitting sovova's mass transfer model, Appl. Soft Comput. 13 (2013) 3792-3805.

[23] Z.Z. Liu, Y. Wang, S.X. Yang, Z.X. Cai, Differential evolution with a two-stage optimization mechanism for numerical optimization, In Proc. IEEE Congr. Evol. Comput., 2016.

[24] N. Lynn, M.Z. Ali, P.N. Suganthan, Population topologies for particle swarm optimization and differential evolution, Swarm Evol. Comput. 39 (2018) 24-35.

[25] N. Lynn, P.N. Suganthan, Ensemble particle swarm optimizer, Appl. Soft Comput. 55 (2017) 533-548.

[26] R. Mallipeddi, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2011) 1679-1696.

[27] A.W. Mohamed, An improved differential evolution algorithm with triangular mutation for global numerical optimization, Comput. Ind. Eng. 85 (2015) 359-375.

[28] M. Nasir, S. Das, D. Maity, S. Sengupta, U. Halder, P.N. Suganthan, A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization, Inf. Sci. 209 (2012) 16-36.

[29] National Institute of Standards and Technology, Engineering statitics handbook, 2003.

[30] M. Omran, A.P. Engelbrecht, A. Salman, Differential evolution methods for unsupervised image classification, In Proc. 7th Congr. Evol. Comput., Piscataway, NJ: IEEE Press, 2005, pp. 966-973.

[31] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2) (2009) 398-417.

[32] H. Samma, C.P. Lim, J.M. Saleh, A new reinforcement learning-based memetic particle swarm optimizer, Appl. Soft Comput. 43 (2016) 276-297.

[33] P. Sarmah, T.K. Gogoi, R. Das, Estimation of operating parameters of a SOFC integrated combined power cycle using differential evolution based inverse method, Appl. Therm. Eng. 119 (2017) 98-107.

[34] K. Singh, R. Das, Simultaneous optimization of performance parameters and energy consumption in induceddraft cooling towers, Chem. Eng. Res. Des. 123 (2017) 1-13.

[35] R. Storn, K. Price, Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341-359.

[36] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, In Proc. IEEE Congr. Evol. Comput., 2013, pp. 71-78.

[37] R. Tanabe, A. Fukunaga, Improving the search performance of SHADE using linear population size reduction, In Proc. IEEE Congr. Evol. Comput., 2014, pp. 1658-1665.

[38] M.N. Tian, X.B. Gao, An improved differential evolution with information intercrossing and sharing mechaisnism for numerical optimiation, Swarm Evol. Comput., doi:10.1016/j.swevo.2017.12.010.

[39] N. Veček, M. Mernik, B. Filipič, M. Črepinšek, Parameter tuning with chess rating system (CRS-tuning) for meta-heuristic algorithms, Inf. Sci. 372 (2016) 446-469.

[40] 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.

[41] Y. Wang, H.X. Li, T.W. Huang, L. Li, Differential evolution based on covariance matrix learning and bimodal distribution parameter setting, Appl. Soft Comput. 18 (2014) 232-247.

[42] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bull., 1945, pp. 80-83.

[43] G.H. Wu, R. Mallipeddi, P.N. Suganthan, R. Wang, H.K. Chen, Differential evolution with multi-population based ensemble of mutation strategies, Inf. Sci. 329 (2016) 329-345.

[44] G.H. Wu, X. Shen, H.F. Li, H.K. Chen, A.P. Lin, Ensemble of differential evolution variants, Inf. Sci. 423 (2018) 172-186.

[45] Q.H. Wu, Y. Wang, Z.P. Lü, A tabu search based hybrid evolutionary algorithm for the max-cut problem, Appl. Soft Comput. 34 (2015) 827-837.

[46] M. Yang, C.H. Li, Z.H. Cai, J. Guan, Differential evolution with auto-ehhanced population diversity, IEEE Trans. Cyber. 45 (2) (2015) 302-315.

[47] J.Q. Zhang, A. C. Sanderson, JADE: Adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (5) (2009) 945-958.

[48] S.X. Zhang, S.Y. Zheng, L.M. Zheng, An efficient multiple variants coordination framework for differential evolution, IEEE Trans. Cyber. 47 (9) (2017) 2780-2793.

[49] L.M. Zheng, S.X. Zhang, K.S. Tang, S.Y. Zheng, Differential evolution powered by collective information, Inf. Sci. 399 (2017) 13-29.

[50] Y.Z. Zhou, W.C. Yi, L. Gao, X.Y. Li, Adaptive differential evolution with sorting crossover rate for continuous optimization problems, IEEE Trans. Cyber. 47 (9) (2017) 1-12.