Received August 14, 2019, accepted October 1, 2019, date of publication October 10, 2019, date of current version October 21, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2945831

Neighbor-Induction and Population-Dispersion in Differential Evolution Algorithm

KUN MIAO, (Member, IEEE), AND ZIYANG WANG
School of Civil Engineering, Central South University, Changsha 410075, China

Corresponding author: Ziyang Wang (ziyang_wang@csu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Project 51478480, and in part by the Hunan Provincial Natural Science Foundation of China under Grant 12JJ3040.

ABSTRACT The differential evolution (DE) optimization algorithm predominantly relies on elite individuals and random difference to direct evolution. Although the strategy is clear and easy to implement, identifying a suitable direction for the DE mutation strongly depends on the direction information provided beforehand. To address this, we present a neighbor-induced mutation operator that simulates the neighbor-induced movement of Antarctic krill to guide the evolution direction in a natural manner. Additionally, center dispersion is proposed to disperse the population and redistribute individual positions to escape search stagnation, inspired by the spreading out of krill around newly discovered food. Comprising the new operator and the center dispersion pattern, this paper proposes a neighbor-induced DE algorithm with dispersion pattern (NDEd). The results of the comparative experiments verify the effectiveness of the neighbor-induced mutation operator and the dispersion pattern. Further, experimental results from 28 test functions of CEC2013 demonstrate that NDEd performs better compared to the other classic DE algorithms.

INDEX TERMS Neighbor-induced operator, dispersion strategy, krill herd algorithm, differential evolution algorithm, global optimization.

I. INTRODUCTION

In 1995, Storn and Price proposed the differential evolution (DE) algorithm [1], a practical, robust, and simple global optimization algorithm. Since then, DE and its variants have become some of the most competitive evolutionary computing algorithms. DE algorithms have been successfully applied in various scientific and engineering fields, such as mechanical engineering design [2], signal processing [3], chemical engineering [4], machine intelligence, and pattern recognition [5].

DE algorithms have exhibited outstanding performance when dealing with optimization problems. However, they have problems such as slow convergence speed and susceptibility to local optima [6]. Properly guiding the direction of evolution may be a key to solving these problems. For example, Cai and Wang [7] proposed a DE frame with neighborhood and direction information. However, the frame heavily depends on the selection of direction information. Although they combined an adaptive operator selection (AOS) from the available direction information with their algorithm in their subsequent research [8], selecting the most suitable type of direction information for the specific DE mutation strategy depends on the classification of the direction information given beforehand. Thus, it is difficult to implement automatic selection of the most suitable direction information in practice.

However, nature provides us with many inspirations, just as many bio-inspired algorithms, such as particle swarm optimization (PSO) and ant colony optimization (ACO), derive their inspiration from nature.

Antarctic krill is well-known for its ability to form large aggregations among marine animals. A krill herd has a unique way of transmitting information such as food, movement, or danger from neighbors. Several mathematical models that evaluate the aggregation mechanisms based on experimental observations have been developed [9], [10]. Further, Gandomi *et al.* proposed a krill herd optimization algorithm [11] that simulates the herding behavior of krill individuals. This study was conducted with the objective of establishing a natural rather than artificial means in the DE algorithm to guide the direction of evolution by simulating the information transmission mode between krill individuals in the DE algorithm.

The associate editor coordinating the review of this manuscript and approving it for publication was Nishant Unnikrishnan.

Our simulation on the transmission of krill information involves the induction behavior and the dispersion activity of the krill. Inspired by the former, a novel neighbor-induced operator is integrated into the DE/current-to-pbest/1/bin strategy to adjust the evolution direction and process, and inspired by the latter, new individual positions are produced using a novel type of a dispersion pattern, which refers to the center of gravity of the population and the current individual positions.

This neighbor-induced mutation operator effectively guides the evolution of DE by dynamically determining the range, magnitude, and direction of the influence from neighbors. Thus, the operator not only retains the advantage of DE/current-to-pbest/1/bin, but also enhances local search capability. In addition, with the new dispersion pattern, local optima near the center of gravity is also avoided through the redistribution of the position of individuals, thereby promoting the population diversity.

In summary, the neighbor-induced mutation operator and the dispersion pattern are the main contributions of this paper. To the best of our knowledge, inspired by krill motion, the neighbor-induced mutation operator in the DE algorithm is being presented for the first time.

Furthermore, the neighbor-induced operator and the dispersion pattern are combined with an existing parameter adaptive strategy to form our proposed neighbor-induced differential evolution with dispersion pattern (NDEd) algorithm. In this study, the NDEd algorithm was compared with a variety of algorithms on benchmark test functions, with experimental results showing that it achieves higher search accuracy.

The remainder of this paper is organized as follows. Section II reviews the DE algorithm and its related neighbor strategies and outlines the characteristics of krill movement. Section III introduces the proposed neighbor-induced operator and the dispersion strategy. Section IV presents and analyzes the tests conducted on the effect of the neighbor-induced operator and the dispersion pattern. In addition, the comparative algorithm experiments conducted on benchmark functions and parameter sensitivity are also discussed. Finally, Section V presents concluding remarks.

II. RELATED WORK

A. STANDARD DE ALGORITHM

The three evolutionary operators (mutation, crossover, and selection) adopted by the traditional DE algorithm are the basis of the algorithm. All independent variables in an optimization problem with D dimensions are expressed as an individual in the population. Let the ith individual be X_i^G, where $X_i^G = (x_{i,1}^G, x_{i,2}^G, \cdots x_{i,D}^G)$, $i = 1, 2, \cdots, N$, and N is the population size.

1) MUTATION

DE uses mutation operators to generate mutation vectors for each generation in the population. In general, the conventional DE method is named using the DE/x/y/z format, where x denotes the target vector selection method in the mutation operator, y represents the number of difference vector pairs used to perturb the target vector, and z is the crossover method. Several commonly used mutation operators are presented below:

Rand/1:

$$V_i^G = X_{r1}^G + F \cdot (X_{r2}^G - X_{r3}^G)(1) \tag{1}$$

Current-to-best/1:

$$V_i^G = X_i^G + F \cdot (X_{\text{best}}^G - X_i^G) + F \cdot (X_{r1}^G - X_{r2}^G) \tag{2}$$

Current-to-pbest/1:

$$V_i^G = X_i^G + F \cdot (X_{\text{pbest}}^G - X_i^G) + F \cdot (X_{r1}^G - \hat{X}_{r2}^G) \tag{3}$$

where G denotes the number of generations, V_i^G represents the ith mutation vector at generation G, and X_{best}^G is the best individual in the current generation; X_{r1}^G, X_{r2}^G, and X_{r3}^G are the individuals randomly selected from the population, and they are unique and different from the target vector; X_{r1}^G is randomly selected from the current population and $\hat{X}_{r2}^G$ is the individual randomly selected from the current population; $X_{r1}^G$is different from both $\hat{X}_{r2}^G$ and the target vector; parameter F is called the mutation factor and is used to control the degree of mutation. The top $100p\%(p \in (0, 1])$ of the population with the best fitness value are called elite individuals; X_{pbest}^G is randomly selected from the current set of elite individuals.

2) CROSSOVER

DE uses the crossover operator to generate test vectors by combining the variables of the target vector and the mutation vector after the mutation. There are three classical crossover operators [12]: binomial crossover [13], [14], exponential crossover [15], and rotation-invariant arithmetic crossover [16]. The binary crossover operator, which is the most common, is the one used in this study:

$$u_{i,j}^G = \begin{cases} v_{i,j}^G, & \mathit{if}\,(\mathit{rand}_{i,j}(0,1) \leq CR \text{ } \mathit{or} \text{ } j = j_{rand}); \\ x_{i,j}^G, & \mathit{otherwise}. \end{cases} \tag{4}$$

where $rand_{i,j}(0, 1)$ is a uniform random number in the range [0,1] and j_{rand} is a uniform random integer in the range [1, D].

3) SELECTION

The fitness value of the test vector can be obtained after crossover. Then, the better individual of the target vector X_i^G and the trial vector U_i^Gwill survive in the next generation. The formulation of the selection operator is as follows:

$$X_i^{G+1} = \begin{cases} U_i^G, & \mathit{if}\, f(U_i^G) \leq f(X_i^G); \\ X_i^G, & \mathit{otherwise}. \end{cases} \tag{5}$$

where $f(U_i^G)$and $f(X_i^G)$ are the objective function values of the trial vector and the target vector (individual), respectively. It is worth noting that the trial vector U_i^G is called a successful update when it is better than or equal to the parent X_i^G in this paper.

B. IMPROVEMENTS TO THE DE ALGORITHM

The current studies on DE focus on improving the mutation strategy [17], [18] and adaptive control of parameters [19], [20], as well as mixing DE algorithms with other heuristic algorithms [21]–[24]. In addition, the neighborhood concept is used in the DE algorithm because DE is a population-based stochastic optimization technique and the interaction between individuals is an important factor to promote evolution. Various studies have been conducted on neighborhood strategies to enhance the performance of the algorithm. The neighborhood strategies are usually divided into static neighbor strategies and dynamic neighbor strategies [25] according to whether a neighbor changes along with the search process.

1) STATIC NEIGHBOR STRATEGY

Das *et al.* [12] used a ring topology to define local and global vectors and proposed global and local neighbor strategies, which they combined according to weights to form a mixed mutation operator. Dorronsoro and Bouvry [26] recommended a decentralized population topology and used different decentralized population schemes to propose and analyze several DE variants. Noroozi *et al.* [27], Noman and Iba [28], Dorronsoro and Bouvry [29], and Liao *et al.* [30] introduced a cell topology that defines the range of neighborhoods and selects parents from neighbors. Liao *et al.* [31] used a ring topology to define the neighborhood and grouped the neighbors to construct a direction vector for mutation. Weber *et al.* [32] incorporated a distributed differential evolution structure with multi-scale factor values into the DE algorithm to improve its distributed performance.

2) DYNAMIC NEIGHBOR STRATEGY

Epitropakis *et al.* [33] assigned a selection probability that is inversely proportional to the distance between an individual and the mutant individual to each individual in the selection process to modify the random selection of parents during the mutation. Gong and Cai [34] described a rank-based mutation operator that selects a part as parents based on the ranking of fitness values in the current population. Cai *et al.* References [7] and [8] used neighbors as a differential term and added adaptive direction information to the mutation strategy. Qu *et al.* [35] proposed a Euclidean distance-based neighbor-determined method that generates a mutation vector, and combined it with a variety of niching DE algorithms to maintain multiple optima. Gong *et al.* [36] combined a biogeographic-based migration operator into DE to guide each individual to learn from good neighbors based on their mobility. Cai *et al.* [25] proposed a neighborhood-adaptive DE algorithm in which an index-based neighborhood topology pool is used to define multiple neighbor relationships for each individual, and self-adapting neighborhood relationships can be used in different evolution stages. Liang *et al.* [37] introduced the fitness Euclidean-distance ratio (FER) technique into DE to locate peaks of functions. This method determines the FER of the current individual based on the difference between the fitness values and the reciprocal of the distance.

The means by which information is shared among multiple populations can also be regarded as another way of sharing information among neighbors. Cui and Li [58] divided a parent population into three sub-populations in terms of individual fitness and evolved them via different DE strategies. They also designed an effective adaptive method to adjust the parameters of the three DE strategies and presented a replacement strategy to update a few of the worst individuals in the parent population equivalent to the same number of the best individuals of the offspring population. Wu *et al.* [57] divided the population into three equal-size smaller subpopulations and one much larger subpopulation, with the different smaller subpopulations using different mutation strategies. After every certain number of generations, the current best mutation strategy is then provided to the larger subpopulation. This resulted in better mutation strategies always being applied in the population. Gao *et al.* [59] divided the whole population into many subpopulations based on a clustering partition that can assign individuals to different promising subregions.

In summary, several researchers have been focusing on DE neighbors. However, a method of determining the direction of evolution based on neighbor interactions is still needed.

C. KRILL HERD SIMULATION

Antarctic krill (Euphausia superba) have been studied extensively among marine organisms, particularly their swarm formation mechanisms [10]. Krill are famous for forming large swarms with adaptive aggregation advantages [38]. Several mathematical models that simulate krill swarms in response to specific biological and environmental processes have been developed to examine the swarm formation. For example, Hofmann *et al.* [9] presented a two-dimensional Lagrangian particle model to examine the spatial distribution of Antarctic krill, in which the location of a krill individual is determined by neighbor-induced movement, foraging activity, and random diffusion. The individual movement direction of krill is modeled by the Lagrangian model as follows:

$$\frac{dX_i}{dt} = N_i + F_i + D_i \tag{6}$$

where N_i, F_i, and D_i represent the three types of activities of krill individuals: motion induced by the presence of other individuals, random diffusion, and foraging.

Inspired by these biological simulations, many computational scientists have developed bio-inspired optimization methods. For example, on the basis of the Lagrangian model [9], Gandomi and Alavi developed the krill herd (KH) algorithm [11] to solve optimization tasks by introducing the basic frameworks of the model. In their model or algorithm, density-dependent attraction of krill and finding food are used as objectives to lead the krill to herd for the global minima. In this process, an individual krill moves toward the best solution when it searches for a food source with the highest density. Specifically, the closer the distance to the high density and food, the lower the value of the objective function.

The KH algorithm has been subsequently developed to further enhance its performance in recent years. With the capability of ergodicity and non-repetition, Wang *et al.* replaced the original KH algorithm's random walk with a chaos sequence to further enhance its global search ability [39]. Further, Saremi *et al.* integrated three one-dimensional chaotic maps (circle, sine, and tent) into the KH algorithm [40]. Fattahi *et al.* utilized a fuzzy system to tune a parameter for setting the participation amount of exploration and exploitation considering different conditions [41]. Abualigah *et al.* combined the KH algorithm with the harmony search (HS) algorithm by adding an improvised solution of the HS algorithm via a new probability factor to the KH algorithm to improve its diversification search ability [42].

III. PROPOSED NEIGHBOR-INDUCED DE WITH DISPERSION STRATEGY

A. NEIGHBOR-INDUCED MUTATION

DE is a population-based evolution technique. Mutation is the internal force that produces new individuals in the evolution. In mutation operators (see (1), (2), or (3)), new vectors are generated by adding a weighted difference vector between two individuals to the current individual. However, these operators do not attach importance to neighborhood effects. In fact, the information from the neighborhood would influence the direction and progress of evolution, and always has an effect on the swarm evolution. Hofmann *et al.* explained that the biological attractive force is usually the dominant factor to form krill swarms, and the density-dependent attraction of krill is also one of the objectives that lead the krill to search targets. With the attractive force, the krill individuals always tend to maintain a high population density and move by the mutual effects [9]. The krill transmit information through mutual induction between them, which may be simulated to compensate for the lack of use of neighbor direction information in the DE algorithm.

Next, we propose an induction operator that simulates the induced motion of the krill, apply this operator to the mutation of the DE algorithm, and then discuss the range of induction.

1) NEIGHBOR-INDUCED OPERATOR

The movement induced by other krill is a key factor to decide the position of a krill in addition to physical diffusion and foraging motion, and the density-dependent attraction from different krill individuals directs the krill to search targets. The krill individuals always tend to move by their mutual effects and maintain a high population density [9]. In their reaction-diffusion model, Azzali *et al.* [43] do not allow the movement speed of krill individuals to change with the environment once the direction is determined. Thus, the step size and the direction may become the main results of the motion induced by other krill. Inspired by this motion, the step size and the direction are regarded as the main factors affecting the evolution processes in DE. We use the neighbor-induced operator ΔX_i to express the induction effect on individual i in the mutation of DE:

$$\Delta X_i = \bar{\delta}_i \cdot d_i \tag{7}$$

$$\bar{\delta}_i = \frac{1}{\text{NP}} \sum_{j=1}^{NP} \|X_i - X_j\| \tag{8}$$

where $\bar{\delta}_i$ is the average distance from other individuals in the population to individual i (X_i and X_j are the positions of the ith and jth individuals); d_i is the direction of induction effects on individual i, which is estimated from the local effect and best effect (see (9)). NP is the total number of individuals in the population.

The higher the dispersion degree of the population, the larger is $\bar{\delta}_i$. In other words, a higher population dispersion degree causes the current individual to be induced to move by the neighbor at a larger step size, which takes charge of the exploration ability of the algorithm. Conversely, a lower dispersion degree produces a smaller $\bar{\delta}_i$; i.e., the current individual requires a smaller step size to move after being induced by neighbors. Thus, the meticulous search takes charge of the exploitation ability of the algorithm. Therefore, the dynamic step size $\bar{\delta}_i$ in the neighbor-induced operator contributes to a good balance between exploration and exploitation.

The induced direction at generation G is represented as follows:

$$d_i = d_i^{\text{local}} + d_i^{\text{target}} \tag{9}$$

where d_i^{local} is the effect from the neighbors, and d_i^{target} is the target direction effect from the global best individual. d_i^{local} indicates the resultant induced directions by different neighbors, although neighbors may act as attraction or repulsion. d_i^{local} and d_i^{target} are expressed as follows:

$$d_i^{\text{local}} = \sum_{j=1}^{N_i} f_{i,j} \cdot \bar{x}_{ij} \tag{10}$$

$$d_i^{\text{target}} = \mu \cdot f_{i,best} \cdot \bar{x}_{i,best} \tag{11}$$

$$\mu = \frac{\text{FES}}{\text{max FES}} \tag{12}$$

where N_i is the number of neighbors of individual i, μ is the coefficient of the effect of the best individual on individual i, $f_{i,j}$ denotes the relative difference of fitness values between individuals i and j, and $\bar{x}_{ij}$ denotes the unit direction vector from individual i to individual j.

$$f_{i,j} = \frac{f_i - f_j}{f^{\text{worst}} - f^{\text{best}}} \tag{13}$$

$$\bar{x}_{i,j} = \frac{X_j - X_i}{\|X_j - X_i\| + \varepsilon} \tag{14}$$

where f_i represents the fitness of the ith individual, f_j is the fitness of the jth ($j = 1, 2, \ldots, N_i, \quad j \neq i$) neighbor, and

f^{best} and f^{worst} are the best and the worst fitness values of the population so far. In (14), ε is a small positive number considered to avoid singularities, *e.g.* $\varepsilon = 0.01$.

The neighbor influences attraction and repulsion. When $f_{i,j} > 0$, neighbor j has a smaller fitness value than individual i, and neighbor j attracts the current individual i to make it move toward it; when $f_{i,j} < 0$, the neighbor repels the current individual i and makes the current individual move in a direction opposite to that of the neighbor.

2) MUTATION WITH NEIGHBOR-INDUCED OPERATORS

The mutation in DE is represented with DE/current-to-pbest/1/bin; thus, (3) is modified with neighbor-induced operator ΔX_i as follows:

$$V_i^G = X_i^G + F_i \cdot ((X_{\text{pbest}}^G - X_i^G) + (X_{r1}^G - \hat{X}_{r2}^G)) + \omega \cdot \Delta X_i^G \tag{15}$$

The top $100p\%$ of the population with the best fitness value are called elite individuals and X_{pbest}^G is randomly selected from the current set of elite individuals; F_i is the mutation factor associated with X_i^G, and is regenerated at each generation by the adaptation process as (21); X_i^G and X_{pbest}^G are selected from the current population; X_{r1}^G is the position of the individual randomly chosen from the population; and $\hat{X}_{r2}^G$ is the position of the individual randomly selected from the current population and external archives. ΔX_i^G is the neighbor-induced operator of individual i at generation G.

The mutation with DE/current-to-pbest/1/bin strategy plays more roles for the need of the exploration in the early stage of evolution, while neighbor induction is reinforced gradually as neighbors bring individuals closer to each other. Thus, ω is designed as a weight to adjust differential mutation with the evolution. Let $c = FES/\max$ FES, where FES represents the current number of fitness evaluations, and maxFES is the maximum number of fitness evaluations. Furthermore, let $\omega = 0.5c$, and ω gradually increases from zero to $0.5c$. Therefore, the weight ω on the influence of neighbors gradually increases with evolution. Thus, ω can adjust the neighbor influence dynamically with the evolutionary process.

3) NEIGHBOR-INDUCED SENSING SCOPE

There is a limit to the range of krill perception. In the impulsive approach [9], the induction from neighbors takes effect in a particular direction only when the individual is inside this range of perception. For the neighbor induction in our simulation, it is necessary to determine the neighbors of individual i.

A circle takes X_i^G as the center and ρ_i as the neighbor-induced radius (*i.e.*, sensing distance), and the individuals in the scope of the circle are all the neighbors of individual i:

$$\rho_i = \frac{1}{(\sigma + p)}\bar{\delta}_i \tag{16}$$

where p is the percentage of the elite individuals in the population, and σ is a parameter that limits the size of the radius. $\bar{\delta}_i$ is the average distance among the neighbors around individual i (see (8)), and it is a major reference length to determine ρ_i. Thus $\bar{\delta}_i$ is a component of neighbor-induced radius ρ_i of individual i.

The neighbor-induced radius is required to be neither too large nor too small. At a given p, if the radius ρ_i is too large, there may be too many elite individuals in the neighborhood, resulting in excessive induction, which may reduce the population diversity to a certain extent; on the other hand, if the radius ρ_i is too small and there are too few elite individuals in the neighborhood, the induction effect is weakened. Therefore, dynamical and self-adaptive change of the induction radius ρ_i with p and $\bar{\delta}_i$ is needed to maintain an appropriate number of elite individuals in the induction area.

In (15), if $\sigma = 0$ and p is very small, the radius ρ_i becomes very large. As a result, the current individual's neighbors cover almost all the individuals of the population, leading to loss of the neighborhood effect. Therefore, σ is introduced to limit the size of the neighbor radius. Our experimental results have verified that the algorithm achieves the best performance at $\sigma = 1.2$. The sensitivity to parameter p is discussed in Section IV.

B. DISPERSION STRATEGY

For complex optimization problems, the DE search always falls into local optima and has difficulty finding a better position with the population gradually gathering. Inspired by krill spreading out when new food is found, we introduce dispersion strategy into the DE algorithm to deal with the plight of entering a local minimum.

Although the absence of food may allow krill to form larger swarms actively, the swarm disperses into single individuals or smaller swarms once food is encountered [9], [44]. At this time, they actively forage for patchy food when the swarm is dispersed [45]. After feeding, larger swarms are formed to search for new food. This result is in a cycle: forage – feeding – forage – feeding. The continuous feeding (meeting food) in the cycle is similar to reaching different local solutions in search of a global optimization solution. Thus, the dispersion pattern of the krill may be simulated to overcome local minimum.

1) CENTER DISPERSION PATTERN

The reaction-diffusion model developed by Azzali *et al.* [43] assumes that krill individuals move toward the center of a swarm. Further, for the attractive force in aggregations, Zhou and Huntley [46] used analogies to Newtonian gravity in the bio-continuum theory of patch dynamics. For dispersion pattern, Cui and Li [48] has designed a shift mechanism (SM) to make the population disperse by shifting some unpromising solutions to a neighborhood of promising solutions, to jump out of stagnation and premature convergence. Inspired by these studies, we designed a center dispersion pattern to acquire new positions ($u_{i,j}$) for the dispersed population based on the center of gravity $x_{gravity,j}^G$ according to

Newtonian gravity as follows:

$$u_{i,j}^G = \begin{cases} x_{\text{gravity},j}^G + rand(0,1)\cdot(x_{r1,j}^G - x_{r2,j}^G) \\ \quad +0.1\cdot rand(0,1), & if\ j \in j_{set}; \\ x_{\text{gravity},j}^G, & otherwise. \end{cases} \tag{17}$$

where $x_{r1,j}^G$, $x_{r2,j}^G$ are two randomly selected individual positions, and $x_{gravity,j}^G (j = 1, 2, \cdots, D)$ is the center of gravity, which is described as

$$x_{\text{gravity},j}^G = \frac{\sum_{i=1}^{NP} \frac{1}{f_i^G}\cdot x_{i,j}^G}{\sum_{i=1}^{NP}\frac{1}{f_i^G}} \tag{18}$$

where f_i^G is the fitness value of individual i. Additionally, in (17), j_{set} is a set reserving the randomly selected dimensions to be modified, and the size of j_{set} is randomly selected from $\{1, 2, \cdots, \mathrm{D}\}$.

2) INTERVAL DISPERSION PATTERN

The above center dispersion pattern can have some effect on escaping stagnation, but it is a diverse choice for individuals to disperse with other patterns. For krill, Hamner *et al.* [45] suggested that krill actively feed on patchy food when the swarm is dispersed, and their distribution is random during the feeding process. For the kind of random position simulation, [48] gives a stochastic distribution pattern of particles in the search space as (19), which is considered as another dispersion pattern in addition to the center dispersion pattern in this paper.

$$x_{i,j}^G = \begin{cases} x_j^{\min} + rand(0,1)\cdot(x_j^{\max} - x_j^{\min}) + 0.1\cdot rand(0,1), \\ \quad if\ j \in j_{set}; \\ x_{i,j}^G, \quad otherwise. \end{cases} \tag{19}$$

The dispersion pattern operates in the interval $(x_j^{\min}, x_j^{\max})$ for dimension j; and this pattern updates $x_{i,j}^G$ in this interval in a random way. Thus, we call it the interval dispersion pattern.

3) INTEGRATION OF DISPERSION PATTERNS

The center dispersion pattern and the interval dispersion pattern are both implemented to promote population diversity from different aspects. The former emphasizes that the locations of the new individuals are scattered based on the center of gravity and the measurement between neighbors, whereas the latter emphasizes that the new locations are randomly dispersed within the whole region. The significance is that the interval dispersion can break the premature convergence in search process, whereas the center dispersion can significantly promote the population to escape local optima near the center of gravity. It is better for both paradigms to work together than to work alone in the creation of diversity.

Premature convergence and stagnation are always present during the search for a minimum. Usually, stagnation can be represented by the average distance of all individuals from the center of the population (the dispersion indicator DP, see (20)) [47]. The DP^G denotes the degree of dispersion of the population at generation G and is used to reflect the population diversity.

$$DP^G = \frac{1}{\mathrm{NP}}\cdot\sqrt{\sum_{i=1}^{\mathrm{NP}}\left\|X_i^G - \frac{1}{\mathrm{NP}}\sum_{j=1}^{\mathrm{NP}}X_j^G\right\|^2} \tag{20}$$

where NP is the number of individuals in the population.

The value of $DP$is small (i.e. $DP < \tau$, where τ is a small positive constant as a threshold for selecting one of the two dispersion patterns) when the degree of population dispersion is low. On the contrary, the value of DP is large (i.e., $DP > \tau$) when the degree of population dispersion is high. For the former, the interval dispersion pattern is used to redistribute the population in the whole search area owing to a high degree of aggregation. For the latter, the center dispersion pattern is considered for using the information of the center of gravity from the current population in a larger degree of dispersion. Hence, the DP is introduced to choose the pattern that would work. The dispersion process including the two patterns is presented as Algorithm 1.

In Algorithm 1, D is the dimension size, FES is the current number of function evaluations, maxFES is the maximal number of function evaluations, p is the proportional coefficient of the elite, and $counter(i)$ is used to record the number of consecutive unsuccessful updates of individual i.

C. THE FRAMEWORK OF NDEd

We combine DE with the neighbor-induced operator and the dispersion operator to develop a new DE variant called the neighbor-induced DE with dispersion (NDEd) algorithm. The pseudocode of NDEd is presented in Algorithm 2, and the flowchart of NDEd is shown in Fig. 1.

As can be seen from Algorithm 2, NDEd features the neighbor-induced operator of the population. For each target individual X_i^G in the population, neighbor-induced operator $\Delta X_{\mathrm{N},i}^G$ is generated for use in the mutation. Moreover, the dispersion process is used to help the search escape from local minima.

In the original DE algorithm, parameters F and CR have a significant influence on performance. Moreover, the DE algorithm has different optimal combinations of F and CR for different mutation strategies and different test functions. In order to reduce the sensitivity of the parameters, we integrated an adaptive parameter strategy [49] into NDEd.

Each individual X_i^G in the population has a corresponding F_i^G and CR_i^G associated with it, and the values of F_i^G and CR_i^G improve as the generation changes; they are generated from the Cauchy and Gaussian distributions, respectively, as defined below.

$$F_i^G = randc(\mu_F^G, 0.1) \tag{21}$$
$$CR_i^G = randn(\mu_{CR}^G, 0.1) \tag{22}$$

Algorithm 1 Dispersion Process

```
input  : $X_i^G$, NP, D, FES, maxFES, $counter(i)$, $p$, $\tau$.
output: $X_i^{G+1}$.
1:  While FES<maxFES:
2:  Calculate DP using (20);
3:  if (DP < $\tau$)        // "Interval Dispersion"
4:      for $i$ from 1 to NP
5:          if $counter(i)$ > D and $X_i^G$ is not the current
            best solution
6:              Generate $X_i^{G+1}$ using (19);
7:              $counter(i) = 0$;
8:              Evaluate $X_i^{G+1}$, FES = FES + 1;
9:          end if
10:     end for
11: else                 // "Center Dispersion"
12:     for $i$ from 1 to NP
13:         if $counter(i)$ > D and $X_i^G$ is not the current
            best solution
14:             Generate $U_i^G$ using (17);
15:             Evaluate $U_i^G$, FES = FES + 1;
16:             if $U_i^G$ is better than $X_i^G$
17:                 $X_i^{G+1} = U_i^G$, $counter(i) = 0$;
18:             else
19:                 $X_i^{G+1} = X_i^G$,
                $counter(i) = counter(i) + 1$;
20:             end if
21:         end if
22:     end for
23: end if
24: end while
```

where *randc*( ) and *randn*( ) denote random sampling from the Cauchy and Gaussian distributions; μ_F^G and μ_{CR}^G are usually taken as 0.5 and $F_i^G \in (0, 1]$, $CR_i^G \in [0, 1]$. To prevent F_i^G from crossing the boundary, F_i^G is reinitialized when $F_i^G \leq 0$ and let $F_i^G = 1$ when $F_i^G \geq 1$; let $CR_i^G = 1$ when $CR_i^G > 1$, and let $CR_i^G = 0$ when $CR_i^G < 0$. Thus, μ_F^G and μ_{CR}^G are updated at each generation by the following formula (all successful F_i^G and CR_i^G are stored in sets S_F and S_{CR} in each generation, respectively):

$$\mu_F^{G+1} = \begin{cases} (1-c_0)\cdot\mu_F^G + c_0\cdot mean_{WL}(S_F), & if\ S_F \neq \emptyset; \\ (1-c_0)\cdot\mu_F^G + c_0\cdot rand(0,1), & otherwise. \end{cases} \tag{23}$$

$$\mu_{CR}^{G+1} = \begin{cases} (1-c_0)\cdot\mu_{CR}^G + c_0\cdot mean_{WA}(S_{CR}), & if\ S_{CR} \neq \emptyset; \\ (1-c_0)\cdot\mu_{CR}^G + c_0\cdot rand(0,1), & otherwise. \end{cases} \tag{24}$$

where parameter c_0 is the learning rate ($c_0 = 0.1$); $mean_{WL}$ and $mean_{WA}$ represent the weighted Lehmer average and the weighted arithmetic mean, respectively, and they are

Algorithm 2 NDEd

```
1:  Set population size NP = 200, $p = 0.1$,
    $\mu_F^0 = \mu_{CR}^0 = 0.5$, $\tau = 0.001$ and A = $\emptyset$; set
    generation G = 0; $counter$(1:NP) = 0
2:  Initialize and evaluate the population;
3:  while FES<maxFES
4:  for $i$ from 1 to NP
5:      $F_i^G$ = randc($\mu_F^G$, 0.1), $CR_i^G$ = randn($\mu_{CR}^G$, 0.1);
6:      Generate $\Delta X_{N,i}^G$ and $V_i^G$ ((7) and (15));
7:      Crossover to generate $U_i^G$ using Equ. (4);
8:      Evaluate $U_i^G$, FES = FES + 1;
9:  end for
10: Update parameter $\mu_F^0$, $\mu_{CR}^0$ ((23) and (24))
11: for $i$ from 1 to NP
12:     if $f(U_i^G) \leq f(X_i^G)$ //successful update
13:             $X_i^{G+1} = U_i^G$, $counter(i) = 0$
14:     else            //unsuccessful update
15:         $X_i^{G+1} = X_i^G$, $X_i^G \to A$,
        $counter(i) = counter(i) + 1$;
16:     end if
17: end for
18: if |A| > NP     //Archive update
19:         Randomly delete |A| -NP individuals from A;
20: end if
21: Dispersion using Algorithm 1;
22: $G = G + 1$;
23: end while
    Output: Best solution
```

calculated as follows [49]:

$$mean_{WL}(S_F) = \frac{\sum_{k=1}^{L}\omega_k\cdot S_{F,k}^2}{\sum_{k=1}^{L}\omega_k\cdot S_{F,k}} \tag{25}$$

$$mean_{WA}(S_{CR}) = \sum_{k=1}^{L}\omega_k S_{CR,k} \tag{26}$$

$$\omega_k = \frac{\Delta f_k}{\sum_{k=1}^{L}\Delta f_k} \tag{27}$$

where $\Delta f_k = \left|f(U_k^G) - f(X_k^G)\right|$, $L = |S_F| = |S_{CR}|$.

IV. EXPERIMENTAL EVALUATION

In this section, we validated the effectiveness of the neighbor-induced operator and the center dispersion added to the DE algorithm. The experiments were based on the CEC2013 [56] benchmark function set. This set includes various optimization functions such as unimodal functions (F1–F5), basic multimodal functions (F6–F20), and composition functions (F21–F28).

The performance of different DE algorithms was measured by the fitness error value, which is defined as $f(x) - f(x^*)$, where $f(x^*)$ is the global optimal solution of the test function and $f(x)$ is the smallest error value obtained after $10^4 \times D$ function evaluations (FES). The comparison algorithms were run 51 times independently on each function. As described

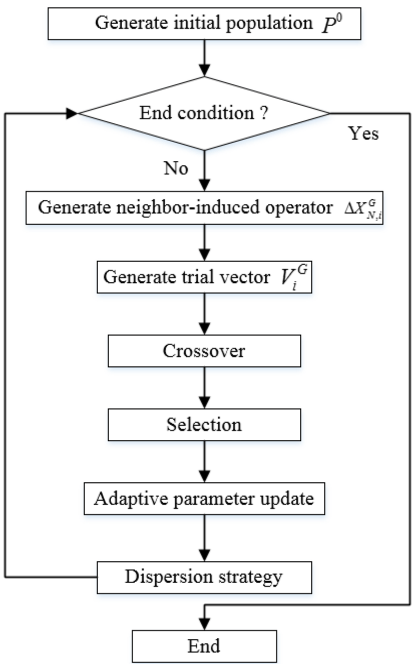

Generate initial population P^0
End condition ?
Yes
No
Generate neighbor-induced operator $\Delta X^G_{N,i}$
Generate trial vector V^G_i
Crossover
Selection
Adaptive parameter update
Dispersion strategy
End

FIGURE 1. **The flowchart of NDEd.**

in [50], solution error values less than 10^{-8} are considered zero. Furthermore, to obtain statistically reasonable conclusions, the error values obtained by different algorithms were compared according to the Wilcoxon signed rank test, with a significance level of 0.05. The results are presented in Table 1. The symbols "−", "=", and "+" respectively indicate that the performance of the compared algorithms is significantly worse than, similar to, or better than that of the considered algorithm. The minimum average error value achieved for each function is highlighted in bold.

TABLE 1. **Results of NDE and DE_pbest for CEC2013 at Dimension = 30 and 50.**

Dimension	D=30		D=50	
F	NDE	DE_pbest	NDE	DE_pbest
F1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F2	**6.57E+06**	7.46E+07	**3.04E+07**	3.01E+08
F3	**2.02E+04**	4.17E+05	**8.35E+02**	1.21E+09
F4	**1.66E+04**	3.25E+04	**1.81E+04**	5.99E+04
F5	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.82E-08
F6	**1.44E+01**	1.55E+01	**4.34E+01**	4.45E+01
F7	**4.80E-01**	1.89E+01	**7.28E-02**	6.23E+01
F8	**2.09E+01**	**2.09E+01**	**2.11E+01**	2.11E+01
F9	**3.51E+01**	3.90E+01	**7.05E+01**	7.24E+01
F10	**1.09E-02**	4.79E+00	**2.20E-02**	5.69E+01
F11	**8.92E+01**	1.19E+02	**2.31E+02**	2.85E+02
F12	**1.42E+02**	1.97E+02	**3.37E+02**	4.07E+02
F13	**1.51E+02**	1.95E+02	**3.34E+02**	4.04E+02
F14	**4.82E+03**	5.01E+03	**1.01E+04**	1.04E+04
F15	**7.06E+03**	7.12E+03	**1.37E+04**	1.39E+04
F16	**2.43E+00**	2.53E+00	**3.30E+00**	3.35E+00
F17	**1.27E+02**	1.56E+02	**2.86E+02**	3.45E+02
F18	**1.91E+02**	2.25E+02	**3.83E+02**	4.60E+02
F19	**1.06E+01**	1.38E+01	**2.47E+01**	3.08E+01
F20	**1.18E+01**	1.24E+01	**2.17E+01**	2.23E+01
F21	2.91E+02	**2.77E+02**	5.07E+02	**4.35E+02**
F22	**4.70E+03**	5.01E+03	**9.58E+03**	1.04E+04
F23	**6.93E+03**	7.22E+03	**1.36E+04**	1.39E+04
F24	**2.00E+02**	2.14E+02	**2.01E+02**	2.86E+02
F25	**2.68E+02**	2.86E+02	**2.90E+02**	3.80E+02
F26	**2.04E+02**	2.06E+02	**2.87E+02**	4.02E+02
F27	**3.10E+02**	1.04E+03	**4.72E+02**	1.95E+03
F28	**3.00E+02**	**3.00E+02**	**4.00E+02**	**4.00E+02**
-/=/+		**23/4/1**		**23/4/1**

A. EFFECT OF THE NEIGHBOR-INDUCED OPERATOR

The neighbor-induced operator considers the influence of each neighbor from the traditional DE/current-to-pbest/1/bin strategy; hence, the individual's evolutionary direction is determined by not only the *pbest* individuals but also the neighbors. In order to better contrast the role of neighbor-induced operators, we designed the following two algorithms:

NDE Algorithm: DE algorithm with the neighbored-induced mutation operator (see (15)). Parameter settings: NP = 200, $p = 0.1$, F = 0.7, Cr = 0.5.

DE_pbest Algorithm: DE algorithm with the DE/current-to-pbest/1/bin mutation strategy. Parameter settings: NP = 200, $p = 0.1$, F = 0.7, Cr = 0.5.

Experiment results of NDE and DE_pbest for CEC2013 are listed at Table 1 (the best results of the algorithms are shown in bold). From, it can be seen that NDE achieves better search performance after the neighbor-induced operator is incorporated into the current-to-pbest/1/bin strategy.

When D = 30, NDE produces results closer to the theoretical value than DE_pbest for 23 out of 28 test functions and achieves the same results with DE_pbest for F1, F5, and F28. Moreover, NDE is significantly better than DE_pbest for F2, F3, F4, F7, F10, F11, and F27. By contrast, NDE produces worse results only for F8 and F21 than DE_pbest.

When D = 50, NDE produces only one worse result (F21) than DE_pbest, and two similar results (F1 and F28). For the other 25 test functions, NDE achieves results closer to the theoretical value than DE_pbest. Among them, NDE is significantly better than DE_pbest for F2, F3, F4, F7, F10, and F27.

The NDE algorithm virtually improve almost all the problems after adding the neighbor-induced operator. Especially, this algorithm exhibits obviously better performance on the unimodal functions (F2, F3, and F4) and simple multimodal functions (F7 and F10), and it is also excellent on the complex multimodal function F27. Thus, the neighbor-induced operator significantly improves the search performance of the DE algorithm.

TABLE 2. Parameter settings.

Algorithm	Parameter settings
NDE	NP = 200, $\mu_F = \mu_{Cr} = 0.5$
NDEd	NP = 200, $\mu_F = \mu_{Cr} = 0.5, \sigma = 1.2$, $\tau = 0.001$, $p = 0.01$

TABLE 3. Comparative results (NDE and NDEd).

Dimension	D=30		D=50	
F	NDE	NDEd	NDE	NDEd
F1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F2	**2.91E+02**	3.27E+02	**4.48E+04**	4.55E+04
F3	**3.36E-01**	1.26E+00	**8.82E+04**	9.08E+04
F4	**3.65E-08**	1.01E+03	**8.42E-04**	1.91E+03
F5	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F6	1.55E+00	**0.00E+00**	**4.34E+01**	**4.34E+01**
F7	**5.11E-01**	7.22E-01	**5.02E+00**	5.63E+00
F8	**2.09E+01**	2.10E+01	**2.11E+01**	2.11E+01
F9	2.87E+01	**1.22E+01**	5.73E+01	**2.86E+01**
F10	**1.82E-02**	2.07E-02	**2.35E-02**	2.80E-02
F11	4.55E-02	**3.74E-03**	1.07E+00	**8.61E-01**
F12	2.89E+01	**1.31E+01**	5.93E+01	**3.10E+01**
F13	4.19E+01	**2.87E+01**	1.06E+02	**9.08E+01**
F14	1.68E+02	**5.02E+01**	3.74E+02	**1.64E+02**
F15	6.70E+03	**3.00E+03**	1.34E+04	**6.56E+03**
F16	2.42E+00	**2.22E+00**	3.25E+00	**2.90E+00**
F17	3.28E+01	**3.20E+01**	5.62E+01	**5.54E+01**
F18	1.27E+02	**3.83E+01**	2.24E+02	**6.50E+01**
F19	2.10E+00	**1.30E+00**	3.86E+00	**2.63E+00**
F20	1.28E+01	**9.97E+00**	2.19E+01	**1.86E+01**
F21	**2.90E+02**	2.97E+02	**8.01E+02**	8.59E+02
F22	2.95E+02	**1.54E+02**	4.37E+02	**2.09E+02**
F23	6.44E+03	**3.04E+03**	1.29E+04	**6.08E+03**
F24	**2.01E+02**	2.01E+02	**2.21E+02**	2.21E+02
F25	2.83E+02	**2.49E+02**	3.70E+02	**3.00E+02**
F26	**2.00E+02**	**2.00E+02**	2.75E+02	**2.63E+02**
F27	4.16E+02	**3.27E+02**	9.17E+02	**7.17E+02**
F28	**3.00E+02**	**3.00E+02**	4.58E+02	**4.00E+02**
-/=/+	**16/8/4**		**17/3/8**	

B. EFFECT OF THE DISPERSION PATTERN

1) EFFECT OF DISPERSION

The dispersion aims to decrease the likelihood of falling into local optimums. Comparative tests of NDE and NDEd were designed to illustrate the dispersion effect using the parameter settings given in Table 2.

NDE Algorithm: Neighbor-induced differential evolution without the dispersion strategy.

NDEd Algorithm: Neighbor-induced differential evolution with the **dispersion** strategy.

The mean values of the solution error of NDE and NDEd are listed in Table 3.

When D = 30, NDEd fails on three functions for the unimodal functions F1–F5 and is the same for two functions, which indicates that adding the dispersion pattern has a negative effect on the unimodal function. This may be because the unimodal function is relatively simple and the performance of NDE itself is sufficient to overcome the local optimum. However, NDEd achieves better results than NDE for 12 out of 15 test functions on the basic multimodal function F6–F20 when D = 30 and D = 50. Furthermore, NDEd performs better or equal on all the composition functions (F21–F28) except F21 in the two cases, despite them having complex mathematical characteristics.

Thus, the dispersion plays a significant role in the multimodal or composition functions for the effect of the dispersion pattern, whereas it is not necessary for simple or unimodal functions.

The shift mechanism is presented in AMECODEs algorithm [48] to allow the population to avoid the situations of stagnation and premature convergence. The mechanism is similar to the center dispersion pattern. The difference between them is that $x^G_{\text{gravity},j}$ in (17) is replaced by $x^G_{\text{pbest},j}$, which is a randomly selected solution from the top elites' solutions. To compare the effect of the center dispersion pattern and the shift mechanism, the comparison experiment between them was done. The test results of CEC2013 of 30- and 50-dimensions problems are listed in the table below.

NDEd Algorithm: Neighbor-induced differential evolution with the **dispersion** pattern.

NDEs Algorithm: Neighbor-induced differential evolution with the **shift** mechanism. In it, the center dispersion pattern is replaced with the shift mechanism in the proposed NDEd algorithm.

We used Wilcoxon's rank-sum test at the 5% significant level to analyze the experimental results. In general, 15 results of NDEd are better than NDEs for the 30-dimensional problems, and 13 results of NDEd for the 50-dimensional problems are better than NDEs. Hence, in general, the center dispersion pattern has better performance than the shift mechanism.

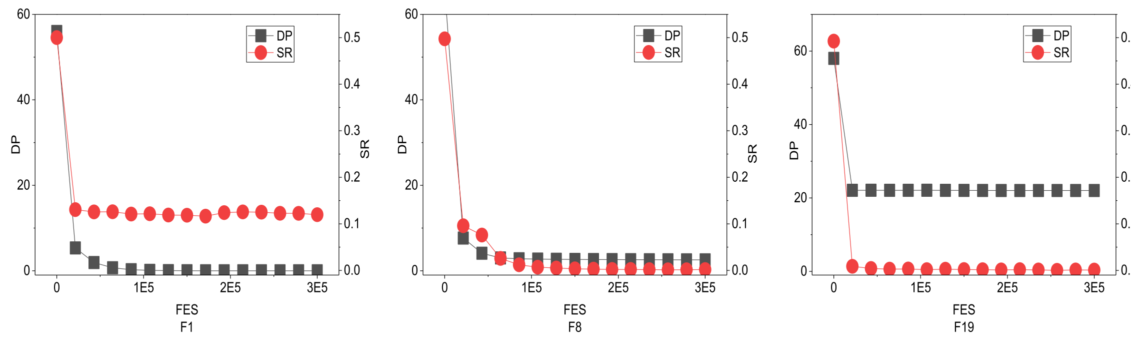
2) DISPERSION EFFECT ANALYSIS

The dispersion indicator (*DP*) represents the average distance of all individuals from the center of the population (see (20)), and the success rate of the population (*SR*) represents the percentage of individuals that are better than the parents within the population of the current generation (see (28)) [47].

$$SR^G = \frac{N_S^G}{\text{NP}} \tag{28}$$

where SR^G denotes the success rate at generation G, and N_S^G is the number of individuals that are better than the parents of the current generation G.

The two indicators are often used to define whether the population is correctly converged [48], [51]. For example, in Fig. 2a, DP gradually decreases to zero but SR remains



(a) Correct convergence (b) Premature convergence (c) Stagnation

FIGURE 2. Determination of whether the convergence is correct by DP and SR.

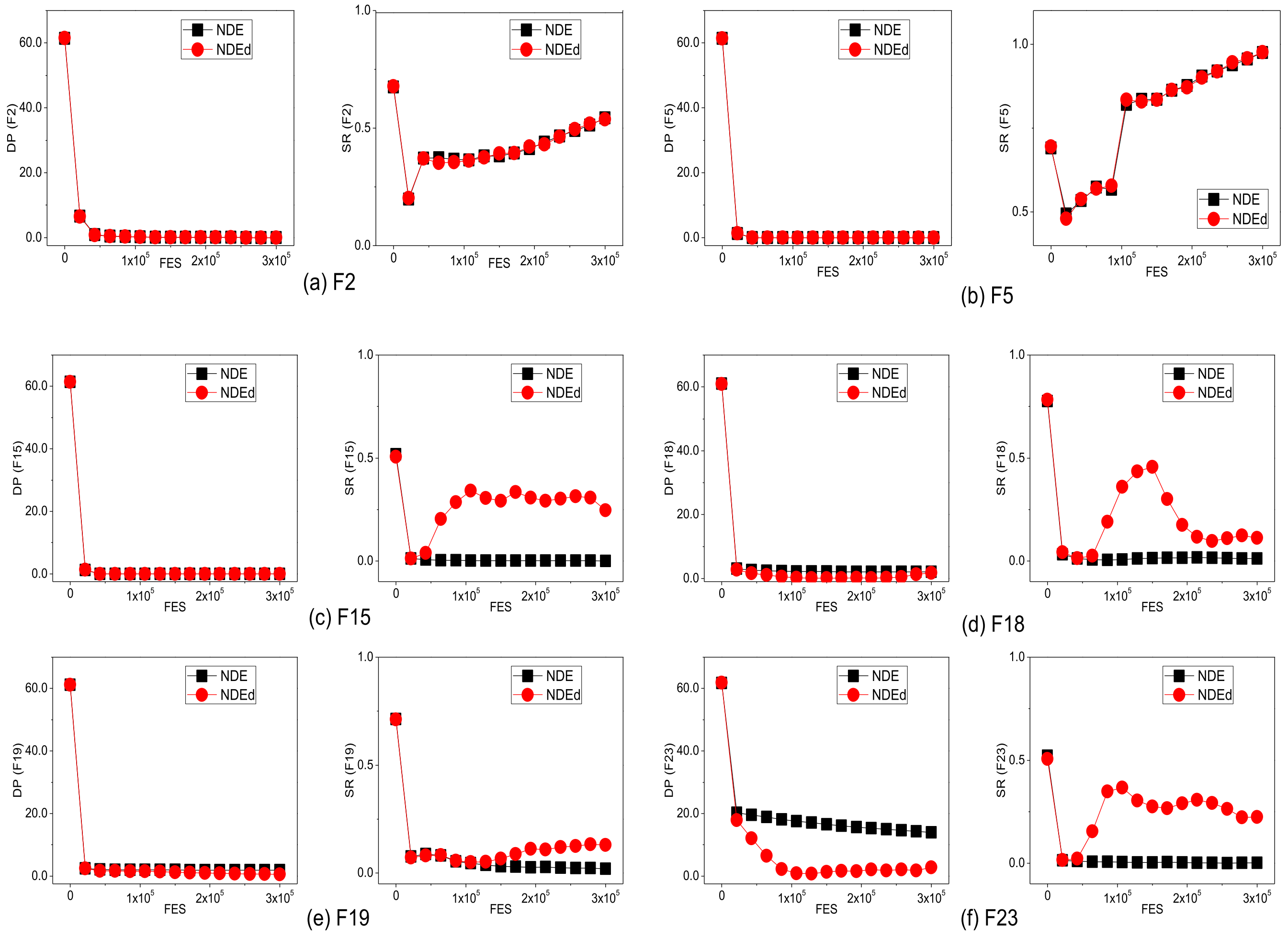


FIGURE 3. DP and SR curves for NDEd and NDE.

at a stable high level with the evolution deepening, which is a correct convergence.

Furthermore, DP and SR can also be used to express premature convergence and stagnation. In Fig. 2b, DP is close to zero but SR is also almost zero in the early stages of population evolution; this case is called premature convergence. In Fig. 2c, SR always approaches zero and DP is always higher than zero after a short evolution, which indicates that the population neither finds a better offspring nor converges further, which is called stagnation. (The results in Fig. 2 comes from tests of functions F1, F8, and F19 (D = 30) in the CEC2013 test function with the basic DE (DE/rand/1/bin, NP = 100, F = 0.5, CR = 0.9)).

Six selected functions (D = 30) from CEC2013 were calculated by NDE and NDEd, respectively. Fig. 3 depicts their behavior for NDE and NDEd by DP and SR with changes in FES. From the figure, we can analyze the ability of NDEd and NDE in overcoming stagnation and premature convergence.

TABLE 4. Comparative results (NDEs and NDEd).

Dimension	D=30		D=50	
F	**NDEs**	**NDEd**	**NDEs**	**NDEd**
F1	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F2	7.85E+01	**3.53E+01**	4.08E+04	**3.73E+04**
F3	**3.58E-05**	9.67E-02	**1.54E+05**	2.56E+05
F4	**4.02E+02**	1.21E+03	**1.07E+03**	8.08E+02
F5	0.00E+00	0.00E+00	2.03E-13	0.00E+00
F6	5.18E-01	**0.00E+00**	4.34E+01	4.34E+01
F7	8.28E-01	**6.53E-01**	5.75E+00	**5.46E+00**
F8	2.09E+01	2.09E+01	2.11E+01	2.11E+01
F9	1.31E+01	**1.25E+01**	3.11E+01	**2.85E+01**
F10	**2.25E-02**	2.37E-02	2.34E-02	**2.30E-02**
F11	1.11E-15	0.00E+00	0.00E+00	0.00E+00
F12	3.87E+01	**1.33E+01**	5.77E+01	**3.03E+01**
F13	7.55E+01	**2.33E+01**	1.47E+02	**8.79E+01**
F14	3.17E+00	**2.24E+01**	**5.07E+01**	1.05E+02
F15	3.51E+03	**3.00E+03**	6.95E+03	**6.63E+03**
F16	**1.16E-01**	2.21E+00	**1.39E-01**	2.98E+00
F17	3.05E+01	3.05E+01	5.09E+01	5.09E+01
F18	6.34E+01	**3.77E+01**	9.96E+01	**6.66E+01**
F19	1.16E+00	**1.06E+00**	2.21E+00	2.23E+00
F20	1.14E+01	**1.02E+01**	2.01E+01	**1.87E+01**
F21	**2.95E+02**	3.06E+02	8.24E+02	**7.92E+02**
F22	**1.14E+02**	1.30E+02	**8.03E+01**	1.29E+02
F23	3.75E+03	**2.98E+03**	7.51E+03	**6.12E+03**
F24	2.01E+02	2.01E+02	2.21E+02	2.21E+02
F25	2.49E+02	2.49E+02	3.09E+02	**3.05E+02**
F26	**2.00E+02**	2.02E+02	**2.63E+02**	2.73E+02
F27	3.49E+02	**3.26E+02**	7.47E+02	**6.83E+02**
F28	3.00E+02	3.00E+02	4.00E+02	4.00E+02
-/=/+	**13/7/8**		**15/7/6**	

For the DP and SR graphs of F2 and F5 in Fig. 3, NDE achieves the correct convergence and NDEd also maintains a similar change curve, indicating that there is no significant performance difference on a simple function for the two algorithms. However, NDEd gets the population out of stagnation more efficiently. Taking F23 as an example, when the NDE's SR approaches zero, the population diversity DP is always close to zero, which indicates that the NDE is in stagnation. By contrast, SR maintains a state higher than zero, although DP gradually approaches zero in NDEd, meaning that NDEd can get the population out of stagnation. The same thing happens in other functions (F15, F18, and F19).

NDEd gets the population out of premature convergence more efficiently. For NDE, DP and SR rapidly move to zero on F15, F18, and F19, suggesting that the population gets into premature convergence. In contrast, for NDEd on F15, F18, and F19, SR remains higher than zero, although DP is close to zero, which indicates that NDEd can make the population escape premature convergence. Hence, we assert that dispersion may effectively promote the convergence of NDEd and get rid of stagnation and premature convergence.

TABLE 5. Parameter settings for the five compared DE variants.

Algorithms	Parameters settings
JADE	NP=100, c=0.1, p=0.05, $\mu_F = \mu_{Cr}$ =0.5
SaDE	NP=50, μ_{Cr} =0.5
CoDE	NP=30, F=1.0, Cr=0.1 or F=1.0, Cr=0.9 or F=0.8, Cr=0.2
SHADE	NP=100, p=0.2, $\mu_F = \mu_{Cr}$ =0.5
PALMDE	NP=10D, F_j =0.5, k=8, p=0.1, a=1.6, T_0 =70
NDEd	NP=200,p=0.1, $\mu_F = \mu_{Cr}$ =0.5, σ=1.2, τ =0.001

C. COMPARISON OF NDED WITH STATE-OF-THE-ART DE VARIANTS

In order to verify the performance of the NDEd, we compared it with the following five state-of-the-art DE variants: JADE [52], CoDE [53], SaDE [54], SHADE [49], and PALMDE [55]. JADE has adaptive control parameters, SaDE can adaptively adjust the mutation strategy and adaptive parameter update mechanism, CoDE is a composite DE with multiple mutation strategies and control parameters, SHADE is a superior DE variant with adaptive parameters and improved external archiving, and PALMDE improves JADE's external archiving update strategy by adopting a new parameter adaptation strategy and a linear population reduction strategy. The parameter settings of JADE, CoDE, SaDE, SHADE, and PALMDE were the same as in their original papers. Table 5 lists the parameter settings of the five DE algorithms.

The maximum number of FES was set to 10,000 $\times$ D in the various cases. The mean and standard deviation of the error values on the 30, 50, and 100-dimensional CEC2013 [56] benchmark functions are listed in Tables 6, 7, and 8, respectively.

1) EXPERIMENTS ON 30-DIMENSIONAL FUNCTIONS

The experimental results for 30-dimensional CEC2013 functions are listed in Table 6.

For the unimodal functions F1–F5, NDEd shows significantly better performance than all other DE variants except PALMDE when D = 30. Compared with JADE and SaDE, NDEd wins three functions (F2, F3, and F4) and reaches the same result for the two other functions (F1 and F5). Compared with CoDE, NDEd fails only on F4. Nonetheless, NDEd shows better or similar performance on the other four functions.

For most of the basic multimodal functions F6–F20, NDEd achieves the best performance. NDEd outperforms JADE, SaDE, PALMDE, CoDE, and SHADE in 10 out of 15 cases

TABLE 6. Results from different DE variants (Dimensions = 30).

DE variants	JADE	SaDE	PALMDE	CoDE	SHADE	NDEd
No.	mean	mean	mean	mean	mean	mean
1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	6.30E+03	4.25E+05	**0.00E+00**	7.75E+04	9.39E+03	3.53E+01
	5.23E+03	1.84E+05	2.01E-08	3.70E+04	6.81E+03	2.47E+02
3	7.80E+05	1.81E+07	1.65E+00	1.42E+06	2.62E+05	**9.67E-02**
	2.74E+06	2.93E+07	1.11E+01	2.82E+06	1.01E+06	6.59E-01
4	6.35E+03	3.02E+03	**0.00E+00**	1.12E-01	1.74E-04	1.21E+03
	1.43E+04	1.25E+03	0.00E+00	2.77E-01	3.08E-04	4.35E+03
5	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
6	1.55E+00	2.73E+01	5.18E-01	3.96E+00	1.04E+00	**0.00E+00**
	6.28E+00	2.66E+01	3.70E+00	9.05E+00	5.13E+00	0.00E+00
7	3.24E+00	3.06E+01	7.35E-01	9.53E+00	4.53E+00	**6.53E-01**
	3.16E+00	1.27E+01	1.05E+00	6.57E+00	4.50E+00	5.89E-01
8	2.09E+01	2.09E+01	2.09E+01	2.08E+01	**2.07E+01**	2.09E+01
	7.56E-02	5.03E-02	1.44E-01	1.11E-01	2.03E-01	5.45E-02
9	2.66E+01	1.72E+01	2.84E+01	1.44E+01	2.73E+01	**1.25E+01**
	1.48E+00	2.50E+00	6.11E+00	3.09E+00	1.80E+00	3.31E+00
10	4.00E-02	3.03E-01	**7.87E-03**	3.47E-02	8.19E-02	2.37E-02
	2.18E-02	1.74E-01	7.34E-03	1.98E-02	4.79E-02	1.40E-02
11	**0.00E+00**	1.17E-01	4.85E+01	3.90E-02	0.00E+00	**0.00E+00**
	0.00E+00	3.80E-01	6.57E+00	1.95E-01	0.00E+00	0.00E+00
12	2.42E+01	4.85E+01	7.24E+01	3.73E+01	2.20E+01	**1.33E+01**
	4.95E+00	1.10E+01	1.68E+01	1.13E+01	3.61E+00	7.32E+00
13	4.82E+01	9.61E+01	8.46E+01	7.60E+01	4.95E+01	**2.33E+01**
	1.39E+01	2.38E+01	1.23E+01	2.28E+01	1.29E+01	1.66E+01
14	3.43E-02	7.37E-01	2.60E+03	3.09E+00	**2.61E-02**	2.24E+01
	2.56E-02	1.08E+00	3.93E+02	3.15E+00	2.63E-02	6.52E+00
15	3.31E+03	4.56E+03	5.86E+03	3.30E+03	3.17E+03	**3.00E+03**
	3.13E+02	9.91E+02	5.32E+02	5.16E+02	3.13E+02	6.33E+02
16	2.04E+00	2.19E+00	1.78E+00	**3.66E-01**	9.36E-01	2.21E+00
	6.09E-01	2.83E-01	9.25E-01	2.05E-01	1.54E-01	2.97E-01
17	**3.04E+01**	3.05E+01	8.47E+01	**3.04E+01**	**3.04E+01**	**3.05E+01**
	1.14E-14	5.97E-02	8.73E+00	2.39E-02	3.52E-14	3.57E-03
18	7.81E+01	1.30E+02	1.53E+02	6.31E+01	7.29E+01	**3.77E+01**
	6.31E+00	3.93E+01	9.34E+00	1.05E+01	5.57E+00	4.47E+00
19	1.44E+00	4.05E+00	7.58E+00	1.62E+00	1.36E+00	**1.06E+00**
	1.10E-01	1.14E+00	1.32E+00	3.64E-01	1.04E-01	1.81E-01
20	1.04E+01	1.10E+01	1.10E+01	1.06E+01	1.03E+01	**1.02E+01**
	5.32E-01	5.53E-01	2.62E-01	6.82E-01	5.32E-01	5.29E-01
21	**3.05E+02**	3.07E+02	2.97E+02	3.09E+02	3.14E+02	**3.06E+02**
	7.12E+01	6.96E+01	3.17E+01	8.98E+01	7.29E+01	5.31E+01
22	9.54E+01	1.30E+02	2.91E+03	**1.12E+02**	9.18E+01	1.30E+02
	2.94E+01	8.48E+01	4.13E+02	2.50E+01	3.16E+01	5.45E+00
23	3.35E+03	4.69E+03	5.96E+03	3.70E+03	3.51E+03	**2.98E+03**
	3.88E+02	1.16E+03	3.60E+02	6.20E+02	4.93E+02	7.48E+02
24	2.15E+02	2.26E+02	**2.01E+02**	2.23E+02	2.04E+02	**2.01E+02**
	1.20E+01	6.62E+00	1.65E+00	9.29E+00	4.76E+00	1.85E+00
25	2.73E+02	2.65E+02	**2.21E+02**	2.55E+02	2.64E+02	2.49E+02
	1.09E+01	1.27E+01	2.46E+01	7.34E+00	1.78E+01	8.85E+00
26	2.10E+02	2.08E+02	**2.00E+02**	2.10E+02	2.09E+02	2.02E+02
	3.39E+01	3.08E+01	0.00E+00	3.35E+01	3.14E+01	1.41E+01
27	7.17E+02	5.96E+02	**3.13E+02**	6.12E+02	3.97E+02	3.26E+02
	2.09E+02	7.04E+01	1.22E+01	1.05E+02	1.08E+02	3.13E+01
28	3.20E+02	**3.00E+02**	**3.00E+02**	**3.00E+02**	**3.00E+02**	**3.00E+02**
	1.43E+02	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00

TABLE 7. Results from different DE variants (Dimensions = 50).

DE variants	JADE	SaDE	PALMDE	CoDE	SHADE	NDEd
No.	mean	mean	mean	mean	mean	mean
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	2.64E+04	7.75E+05	**1.11E+04**	2.38E+05	2.81E+04	3.73E+04
	1.40E+04	2.20E+05	8.20E+03	7.72E+04	1.37E+04	2.28E+04
3	1.30E+06	9.59E+07	**3.48E+04**	1.12E+07	6.85E+05	2.56E+05
	2.11E+06	1.74E+08	1.39E+05	1.34E+07	1.10E+06	7.59E+05
4	1.09E+04	4.58E+03	**3.62E-06**	1.82E-01	2.72E-03	8.08E+02
	2.12E+04	1.50E+03	7.29E-06	3.01E-01	6.28E-03	4.11E+03
5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
6	4.36E+01	5.68E+01	**4.34E+01**	4.39E+01	4.36E+01	**4.34E+01**
	7.95E-01	2.22E+01	0.00E+00	1.54E+00	7.88E-01	0.00E+00
7	2.26E+01	5.09E+01	**3.09E+00**	4.09E+01	2.21E+01	5.46E+00
	1.15E+01	9.63E+00	2.14E+00	1.10E+01	9.83E+00	2.22E+00
8	2.11E+01	2.11E+01	2.11E+01	2.10E+01	**2.09E+01**	2.11E+01
	8.93E-02	3.86E-02	4.19E-02	9.82E-02	1.72E-01	3.50E-02
9	5.44E+01	3.94E+01	4.33E+01	3.17E+01	5.54E+01	**2.85E+01**
	2.00E+00	4.13E+00	1.66E+01	5.94E+00	2.71E+00	5.06E+00
10	2.76E-02	2.93E-01	**2.03E-02**	5.23E-02	8.97E-02	2.30E-02
	1.97E-02	1.57E-01	1.64E-02	2.63E-02	4.60E-02	1.77E-02
11	**0.00E+00**	2.01E+00	1.76E+02	5.46E-01	0.00E+00	**0.00E+00**
	0.00E+00	1.89E+00	1.71E+01	8.28E-01	0.00E+00	0.00E+00
12	5.62E+01	1.30E+02	2.07E+02	8.81E+01	5.90E+01	**3.03E+01**
	1.01E+01	2.28E+01	1.27E+01	1.97E+01	1.01E+01	8.39E+00
13	1.26E+02	2.42E+02	2.46E+02	1.96E+02	1.36E+02	**8.79E+01**
	2.97E+01	4.32E+01	1.52E+01	3.57E+01	2.78E+01	2.74E+01
14	4.14E-02	8.79E+00	8.10E+03	3.08E+01	**3.67E-02**	1.05E+02
	2.01E-02	1.68E+01	8.42E+02	2.27E+01	1.63E-02	1.67E+01
15	6.95E+03	9.19E+03	1.26E+04	6.99E+03	6.79E+03	**6.63E+03**
	4.68E+02	2.07E+03	3.38E+02	8.19E+02	4.34E+02	9.69E+02
16	2.13E+00	3.04E+00	3.26E+00	**9.05E-01**	1.26E+00	2.98E+00
	7.98E-01	3.49E-01	2.80E-01	3.62E-01	1.93E-01	4.05E-01
17	**5.08E+0**1	5.13E+01	2.40E+02	5.23E+01	**5.08E+01**	5.09E+01
	0.00E+00	3.41E-01	1.92E+01	6.25E-01	0.00E+00	1.84E-02
18	1.44E+02	1.59E+02	3.14E+02	1.25E+02	1.35E+02	**6.66E+01**
	1.15E+01	6.21E+01	1.42E+01	2.13E+01	1.02E+01	5.50E+00
19	2.71E+00	1.12E+01	1.97E+01	3.26E+00	2.76E+00	**2.23E+00**
	2.55E-01	2.67E+00	1.25E+00	6.81E-01	3.09E-01	2.82E-01
20	1.97E+01	2.00E+01	2.06E+01	1.96E+01	1.95E+01	**1.87E+01**
	6.85E-01	8.15E-01	3.42E-01	8.51E-01	6.94E-01	7.65E-01
21	8.30E+02	9.05E+02	1.02E+03	**7.56E+02**	8.62E+02	7.92E+02
	4.03E+02	3.30E+02	2.05E+02	4.39E+02	3.65E+02	4.30E+02
22	2.22E+01	6.01E+01	8.94E+03	3.73E+01	**1.25E+01**	1.29E+02
	4.07E+01	1.83E+02	6.97E+02	1.41E+01	5.64E+00	1.79E+01
23	7.22E+03	8.77E+03	1.25E+04	7.16E+03	7.60E+03	**6.12E+03**
	6.95E+02	2.18E+03	4.74E+02	9.07E+02	6.33E+02	1.02E+03
24	2.47E+02	2.77E+02	**2.11E+02**	2.63E+02	2.33E+02	2.21E+02
	1.93E+01	9.36E+00	4.92E+00	1.21E+01	9.94E+00	6.37E+00
25	3.54E+02	3.47E+02	**2.92E+02**	3.17E+02	3.45E+02	3.05E+02
	1.91E+01	1.18E+01	7.07E+00	1.22E+01	3.01E+01	1.19E+01
26	3.14E+02	2.32E+02	**2.38E+02**	2.88E+02	2.53E+02	2.73E+02
	1.08E+02	7.05E+01	5.40E+01	8.82E+01	7.22E+01	5.99E+01
27	1.40E+03	1.18E+03	**4.57E+02**	1.05E+03	8.75E+02	6.83E+02
	3.10E+02	1.14E+02	6.51E+01	1.09E+02	2.34E+02	1.44E+02
28	4.00E+02	4.72E+02	**4.00E+02**	4.59E+02	4.59E+02	**4.00E+02**
	0.00E+00	5.12E+02	0.00E+00	4.22E+02	4.20E+02	0.00E+00

TABLE 8. Results from different DE variants (dimensions = 100).

DE variants	JADE	SaDE	PALMDE	CoDE	SHADE	NDEd
No.	mean	mean	mean	mean	mean	mean
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	1.50E+05	2.59E+06	3.28E+05	9.26E+05	1.58E+05	**9.83E+04**
	5.02E+04	4.96E+05	7.39E+04	2.46E+05	3.88E+04	2.94E+04
3	2.09E+08	1.52E+09	**3.03E+06**	1.95E+08	6.04E+07	1.56E+07
	2.03E+08	9.30E+08	2.52E+06	2.04E+08	4.79E+07	1.56E+07
4	1.12E+04	9.18E+03	8.51E-02	4.35E+00	6.28E-03	**1.09E-03**
	2.85E+04	3.52E+03	4.62E-02	1.06E+01	6.19E-03	1.59E-03
5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
6	1.09E+02	1.43E+02	1.20E+02	1.49E+02	**7.06E+01**	7.21E+01
	6.87E+01	5.55E+01	5.69E+01	5.05E+01	6.02E+01	6.21E+01
7	7.24E+01	1.09E+02	**9.85E+00**	7.62E+01	7.02E+01	2.75E+01
	1.58E+01	1.96E+01	3.57E+00	1.16E+01	1.38E+01	7.00E+00
8	2.13E+01	2.13E+01	2.13E+01	**2.12E+01**	2.12E+01	2.13E+01
	4.25E-02	3.07E-02	2.37E-02	6.01E-02	8.66E-02	3.28E-02
9	1.33E+02	1.09E+02	1.47E+02	9.28E+01	1.37E+02	**8.90E+01**
	3.67E+00	6.88E+00	2.41E+00	1.02E+01	4.04E+00	8.21E+00
10	2.81E-02	1.69E-01	**1.97E-02**	8.78E-02	5.26E-02	2.26E-02
	1.91E-02	9.76E-02	1.64E-02	3.92E-02	2.57E-02	2.31E-02
11	0.00E+00	3.47E+01	5.64E+02	2.16E+01	1.76E-01	**0.00E+00**
	0.00E+00	1.26E+01	2.97E+01	7.23E+00	1.24E+00	0.00E+00
12	2.00E+02	4.59E+02	5.06E+02	3.05E+02	2.02E+02	**1.43E+02**
	2.55E+01	6.15E+01	1.72E+02	5.52E+01	2.22E+01	2.70E+01
13	4.36E+02	7.29E+02	6.82E+02	5.98E+02	4.65E+02	**2.88E+02**
	5.51E+01	7.89E+01	2.31E+01	7.12E+01	4.81E+01	4.27E+01
14	8.06E-02	8.02E+01	2.43E+04	1.71E+02	**5.93E-02**	5.40E+02
	1.66E-02	8.44E+01	9.57E+02	8.55E+01	1.70E-02	6.73E+01
15	1.50E+04	1.47E+04	2.81E+04	1.47E+04	1.46E+04	**1.27E+04**
	5.99E+02	2.44E+03	4.35E+02	1.21E+03	6.82E+02	1.69E+03
16	1.94E+00	3.67E+00	3.90E+00	**1.81E+00**	1.84E+00	3.31E+00
	3.99E-01	2.13E-01	2.31E-01	5.66E-01	1.82E-01	9.25E-01
17	**1.02E+02**	1.11E+02	7.09E+02	1.15E+02	**1.02E+02**	**1.02E+02**
	0.00E+00	4.43E+00	2.44E+01	2.97E+00	1.16E-13	5.47E-02
18	3.70E+02	4.26E+02	7.87E+02	3.51E+02	3.55E+02	**1.98E+02**
	2.30E+01	5.70E+01	1.83E+01	4.41E+01	2.40E+01	3.30E+01
19	8.99E+00	3.88E+01	5.25E+01	9.10E+00	1.12E+01	**7.67E+00**
	1.11E+00	9.94E+00	2.94E+00	1.58E+00	1.54E+00	1.05E+00
20	5.00E+01	4.99E+01	5.00E+01	4.99E+01	4.99E+01	5.00E+01
	1.30E-07	2.15E-01	3.04E-09	1.81E-01	2.95E-01	0.00E+00
21	3.78E+02	3.90E+02	4.00E+02	**3.53E+02**	3.76E+02	3.88E+02
	4.15E+01	3.00E+01	0.00E+00	5.04E+01	3.45E+01	3.25E+01
22	6.06E+01	**1.17E+02**	2.59E+04	1.90E+02	5.29E+01	6.50E+02
	5.02E+01	8.35E+01	1.13E+03	9.43E+01	4.78E+01	1.16E+02
23	1.68E+04	1.72E+04	2.83E+04	1.64E+04	1.74E+04	**1.33E+04**
	1.10E+03	2.38E+03	6.71E+02	1.34E+03	1.02E+03	1.63E+03
24	3.40E+02	4.28E+02	**2.51E+02**	3.62E+02	3.23E+02	2.84E+02
	2.06E+01	1.55E+01	9.17E+00	1.79E+01	1.35E+01	1.09E+01
25	5.75E+02	5.76E+02	**4.27E+02**	4.89E+02	5.19E+02	4.75E+02
	4.89E+01	1.68E+01	1.38E+01	2.50E+01	7.11E+01	2.73E+01
26	5.90E+02	5.29E+02	**3.63E+02**	5.04E+02	4.61E+02	4.09E+02
	6.11E+01	1.81E+01	7.85E+00	2.62E+01	5.98E+01	2.94E+01
27	2.69E+03	2.70E+03	8.71E+02	2.34E+03	1.73E+03	**1.35E+03**
	7.76E+02	1.94E+02	9.97E+01	2.33E+02	3.03E+02	2.00E+02
28	3.40E+03	7.11E+03	**2.56E+03**	3.28E+03	3.24E+03	3.03E+03
	1.21E+03	2.09E+03	2.30E+01	9.93E+02	1.13E+03	8.74E+02

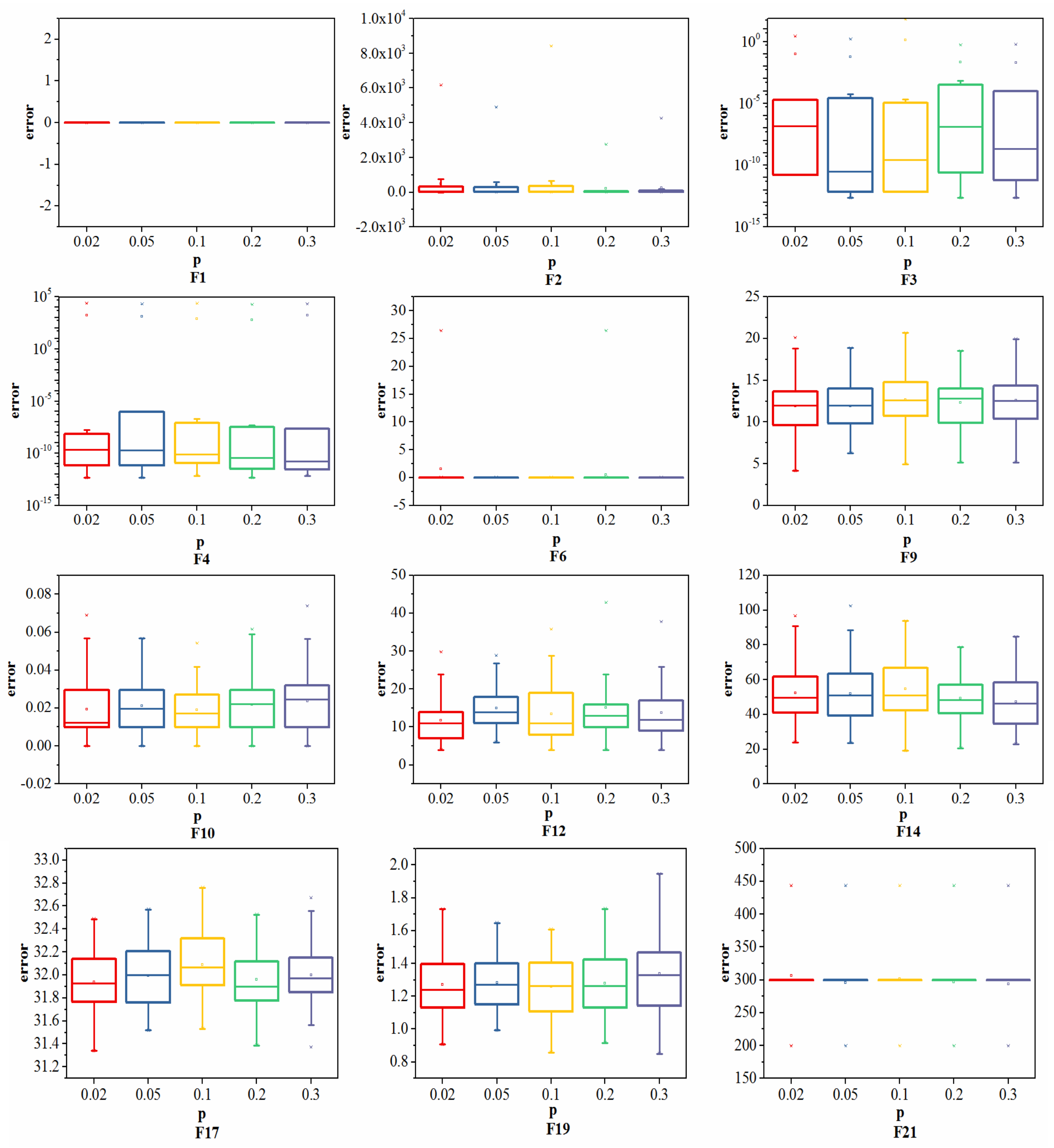


FIGURE 4. Boxplots of NDEd with different *p* values on representative functions (D = 30).

(F6, F7, F9, F11, F12, F13, F15, and F18–F20). Moreover, NDEd has much smaller error values than the competitors on F12, F13, and F18. Compositional functions F21–F28 with complex mathematical characteristics are very difficult to optimize. However, NDEd defeats JADE, SaDE, PALMDE, CoDE, and SHADE in seven (F21 and F23–F28), six (F21 and F23–F27), three (F22–F24), seven (F21 and F23–F28), and seven (F21 and F23–F28) functions, respectively. Furthermore, NDEd only loses to JADE, SaDE, CoDE, and SHADE in F22, and PALMDE in F25– F27, respectively. Thus, NDEd performs best for the 30-dimensional functions.

2) EXPERIMENTS ON 50- AND 100-DIMENSIONAL FUNCTIONS

To further study the performance of NDEd on higher-dimensional functions, NDEd was also compared with the 50- and 100-dimensional functions of the CEC2013. The experimental results are listed in Table 7 and Table 8.

The experimental results for the 50-dimensional function are listed in Table 7. With pairwise comparison, the winning percentage of NDEd is significantly greater than that of other algorithms. NDEd defeats JADE, SaDE, PALMDE, CoDE,

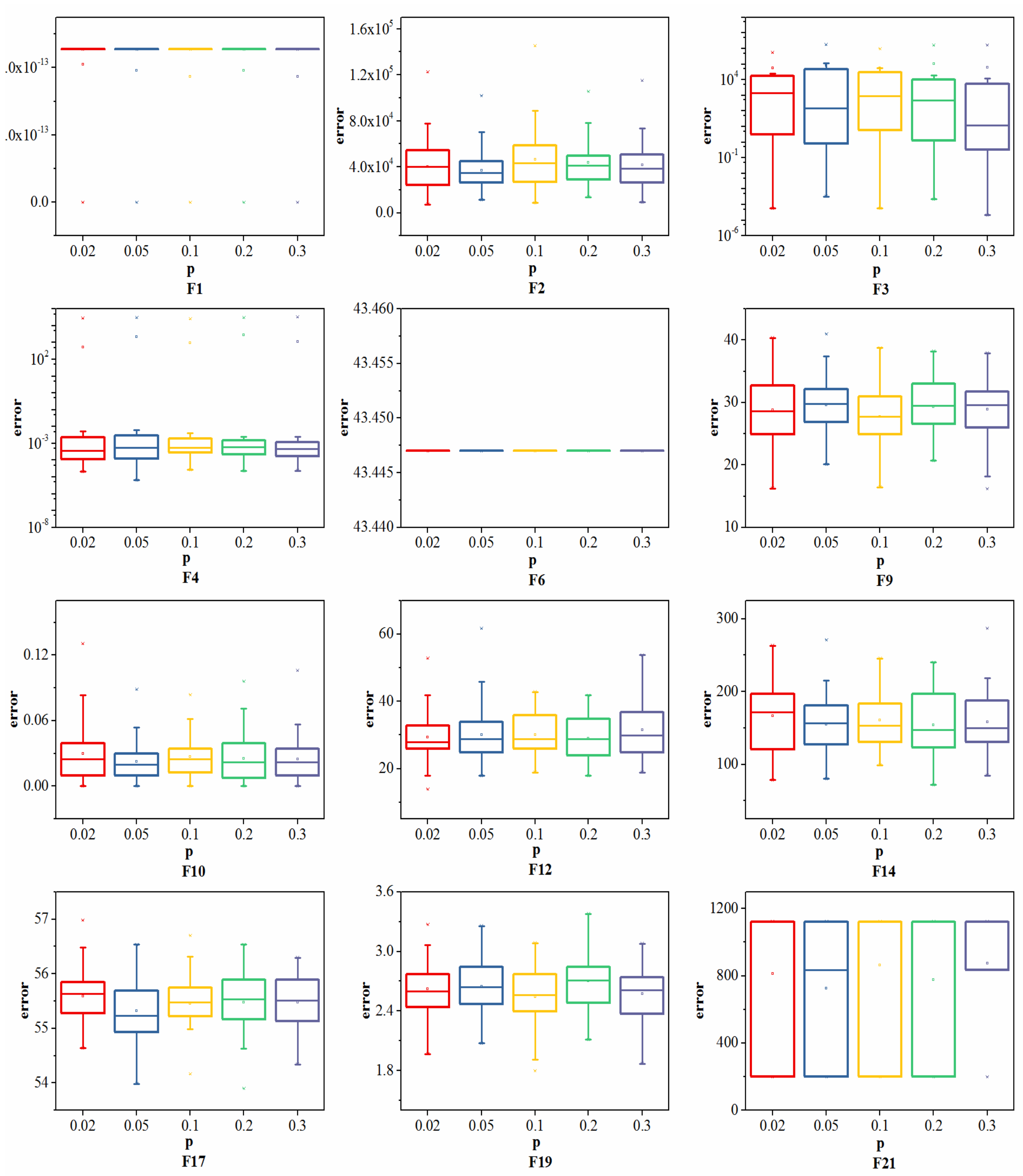


FIGURE 5. Boxplots of NDEd with different *p* values on representative functions (D = 50).

and SHADE on 18, 22, 14, 20 and 17 out of 28 functions while it fails to win on 5, 3, 9, 6, and 8 out of 28 functions, respectively.

The experimental results of the 100-dimensional function are listed in Table 8. Compared with JADE, SaDE, PALMDE, CoDE, and SHADE in pairs, NDEd achieves better results in 18, 22, 16, 20, and 19 out of 28 functions and worse in 4, 3, 8, 6, and 6 out of 28 functions, respectively. The performance of NDEd for 100-dimensional functions is better than that for 50-dimensional functions.

Table 9 summarizes the overall statistical comparisons with these algorithms on the 30-, 50-, and100-dimensional functions. Table 10 gives the overall performance ranking of all algorithms on the 30-, 50-, and100-dimensional functions with the Friedman test. As the table shows, NDEd gives the minimum ranking (2.35). Hence, NDEd is more competitive than other algorithms and achieves the overall better performance than any one of these famous algorithms in the three different dimensions, although NDEd does not obtain the best solution on each function.

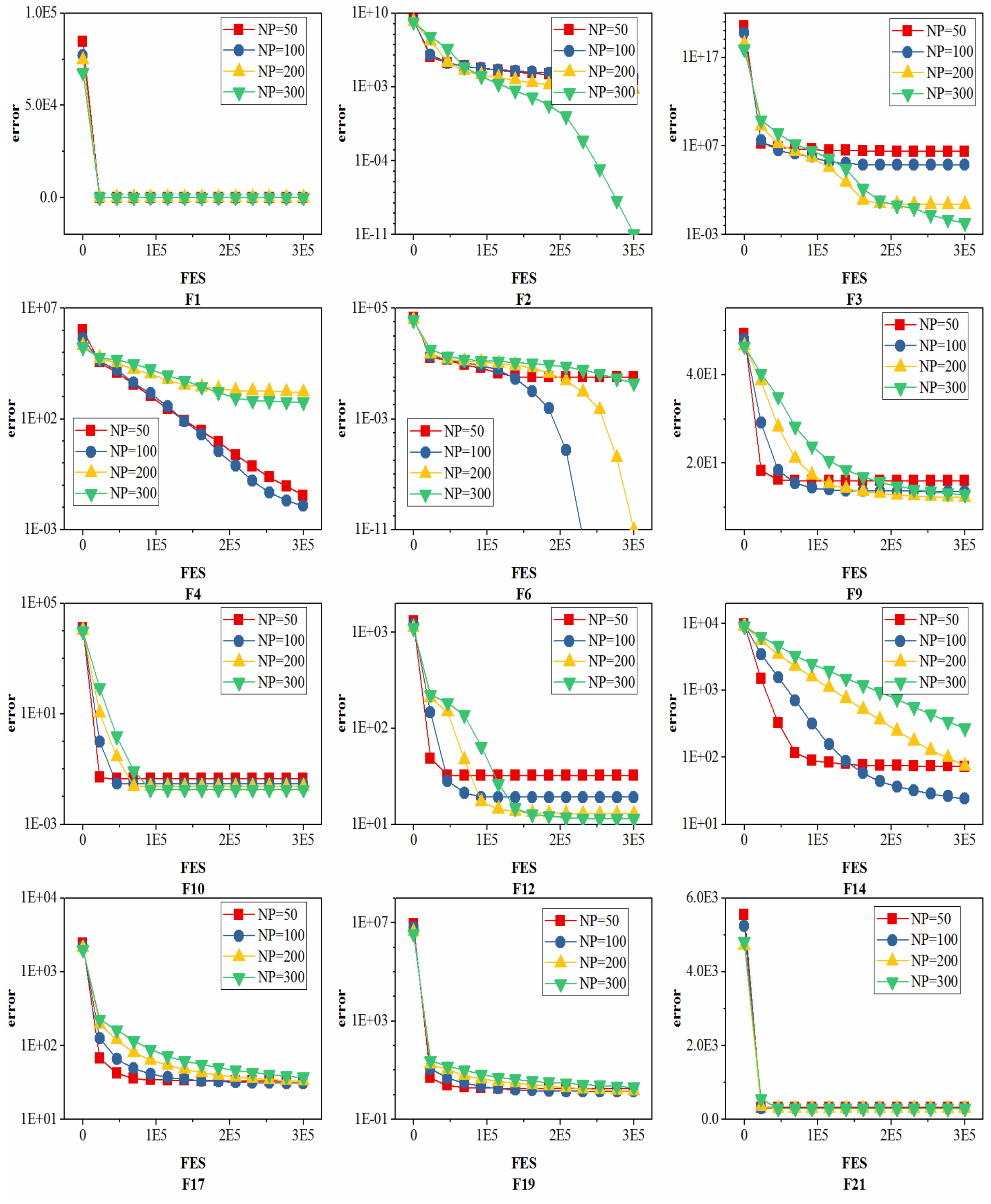


FIGURE 6. Convergence curves of the mean error on various selected functions (D = 30).

D. PARAMETER SENSITIVITY

1) PROPORTION OF ELITE INDIVIDUALS

Parameter p is the proportion of elite individuals in the population ($p \in (0, 1)$) and controls the number of elite individuals used by neighbors in evolution in NDEd. The population is guided by more elites as p increases according to (15), while each individual's neighbor radius and the number of neighbors is reduced. Therefore, parameter p significantly influences the diversity of the population in the evolution process, which has a great influence on the performance of

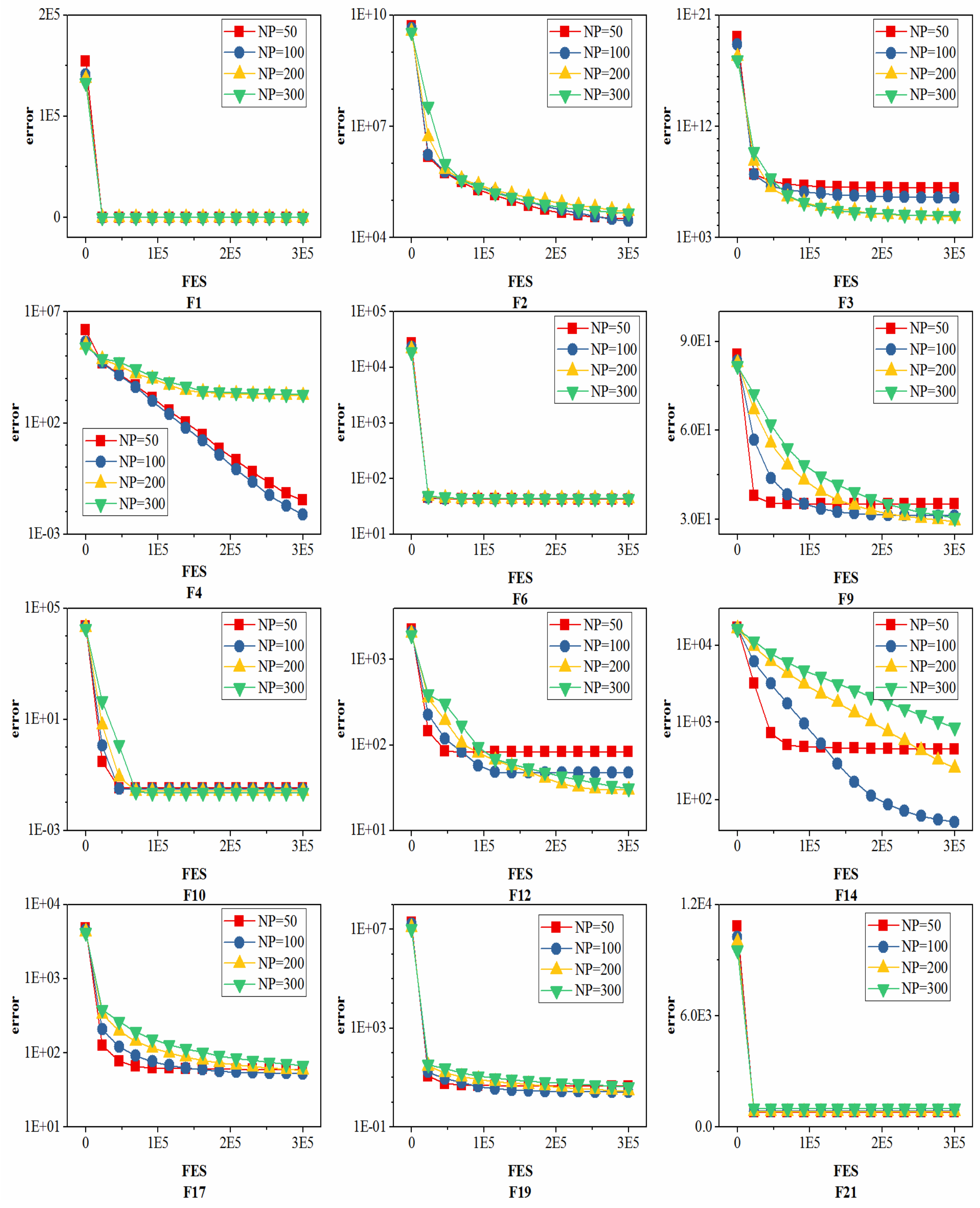


FIGURE 7. Convergence curves of the mean error on various selected functions (D = 50).

NDEd. In order to find a general value for p, we tested 12 representative functions of CEC2013 by NDEd with various p values ($p = 0.02, 0.05, 0.1, 0.2$, and 0.3) when D = 30 and D = 50. The other parameters remained the same as those listed in Table 2. The box plots of the experimental results at D = 30 and D = 50 are shown in Figs. 4 and 5, respectively.

As shown in Fig. 4, p has a similar effect on the quality of the NDEd solution in most cases when D = 30, *i.e.*, functions is not sensitive to p when D = 30. As shown in Fig. 5, when D = 50, except for F21, p does not have a significant influence on the performance of NDEd. In F21, the result is more stable when $p = 0.3$. Overall, in most cases, NDEd

TABLE 9. Statistical comparison of different algorithms with NDEd.

vs. NDEd		D=30	D=50	D=100
JADE	+(better)	5	5	4
	=(similar)	4	5	6
	-(worse)	**19**	**18**	**18**
SaDE	+(better)	2	3	3
	=(similar)	6	3	3
	-(worse)	**20**	**22**	**22**
PALMDE	+(better)	8	9	8
	=(similar)	5	5	4
	-(worse)	**15**	**14**	**16**
CoDE	+(better)	6	6	6
	=(similar)	3	2	2
	-(worse)	**19**	**20**	**20**
SHADE	+(better)	6	8	6
	=(similar)	4	3	3
	-(worse)	**18**	**17**	**19**

TABLE 10. Overall performance ranking of the compared DE variants on the 30-, 50-, and 100-dimensional CEC2013 benchmarks set by the Friedman's test.

Algorithm	Ranking (D = 30, 50 and 100)
JADE	3.54
SaDE	4.74
PALMDE	3.84
CoDE	3.60
SHADE	2.93
NDEd	**2.35**

produces stable results at $p = 0.1$, which may be used as a general parameter value for the NDEd algorithm.

2) POPULATION SIZE

To determine the effect of the population size NP on NDEd, we tested 12 representative functions of CEC2013 using the NDEd algorithm with different NP values (specifically, NP = 50, 100, 200, and 300) when D = 30 and D = 50. Other parameters remained the same as those listed in Table 2. The convergence curves of the function's average error are shown in Figs. 6 and 7.

As shown in Fig. 6 (D = 30), NP has no significant effect on the performance of NDEd in most cases (F1, F9, F10, F12, F17, F19, and F21). However, NP = 50 and NP = 100 have slightly faster convergence speed. In terms of the capability to find the global optimal solution, NP = 200 is the strongest.

In the case of D = 50 (Fig. 7), the performance of NDEd is insensitive to NP for some functions (*e.g.*, F1, F2, F6, F10, F17, F19, and F21). In general, NP is not sensitive for most functions, and NP = 100–200 is more suitable for most cases considering the overall performance of all test functions.

V. CONCLUSION

In this paper, we proposed a novel neighbor-induced mutation operator to improve the global optimization ability of the DE algorithm. The use of this operator enables the individuals to evolve with the guidance of not only the elite individuals and the stochastic difference but also the induced information from neighbors. This makes the population less likely to be misled by the local optimal solutions to non-promising areas.

To re-distribute a population, the center dispersion pattern presented in this paper can be seen as a complement to the traditional population distribution methods when the search falls into premature convergence.

The test results show that the proposed algorithm with the new operator and pattern produces superior solutions than the traditional DE algorithm. Further, the performance of the algorithm is superior to that of the classic and advanced DE algorithm variants. Moreover, the sensitivity test of the key parameters demonstrates that the proposed algorithm has low parameter dependence.

In summary, through the simulation of the movement of krill herd in the optimization process, inspiration derived from nature can definitely provide us with numerous advantages.

As future work, multiple elites corresponding to different ways of choosing neighbors and maintaining a better balance between the neighbor and elite guidance can be investigated. The proposed algorithm can therefore be used to verify the effectiveness of engineering optimization problems in real world scenarios.

REFERENCES

[1] A. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, Dec. 1997. doi: 10.1023/A:1008202821328.

[2] R. Joshi and A. C. Sanderson, "Minimal representation multi-sensor fusion using differential evolution," in *Differential Evolution: A Practical Approach to Global Optimization* (Natural Computing Series). Berlin, Germany: Springer, 1999.

[3] S. Das and A. Konar, "Two-dimensional IIR filter design with modern search heuristics: A comparative study," *Int. J. Comput. Intell. Appl.*, vol. 6, no. 3, pp. 329–355, 2006.

[4] F. Wang and H.J. Jang, "Parameter estimation of a bioreaction model by hybrid differential evolution," in *Proc. Congr. Evol. Comput.*, La Jolla, CA, USA, Jul. 2000, pp. 410–417.

[5] S. Das, A. Abraham, and A. Konar, "Automatic clustering using an improved differential evolution algorithm," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 38, no. 1, pp. 218–237, Jan. 2008. doi: 10.1109/tsmca.2007.909595.

[6] J. Lampinen and I. Zelinka, "On stagnation of the differential evolution algorithm," in *Proc. 6th Int. Conf. Soft Comput.*, Brno, Czech Republic, 2000, pp. 76–83.

[7] Y. Cai and J. Wang, "Differential evolution with neighborhood and direction information for numerical optimization," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 2202–2215, Dec. 2013. doi: 10.1109/TCYB.2013.2245501.

[8] Y. Cai, J. Wang, Y. Chen, T. Wang, H. Tian, and W. Luo, "Adaptive direction information in differential evolution for numerical optimization," *Soft Comput.*, vol. 20, no. 2, pp. 465–494, 2016.

[9] E. E. Hofmann, A. E. Haskell, J. M. Klinck, and C. M. Lascara, "Lagrangian modelling studies of Antarctic krill (Euphausia superba) swarm formation," *ICES J. Mar. Sci.*, vol. 61, no. 4, pp. 617–631, Jan. 2004. doi: 10.1016/j.icesjms.2004.03.028.

[10] G. Flierl, D. Grünbaum, S. Levins, and D. Olson, "From individuals to aggregations: The interplay between behavior and physics," *J. Theor. Biol.*, vol. 196, no. 4, pp. 397–454, Feb. 1999. doi: 10.1006/jtbi.1998.0842.

[11] A. H. Gandomi and A. H. Alavi, "Krill herd: A new bio-inspired optimization algorithm," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 17, pp. 4831–4845, May 2012.

[12] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 526–553, Jun. 2009.

[13] M. Weber and F. Neri, "Contiguous binomial crossover in differential evolution," in *Swarm and Evolutionary Computation*, vol. 7269, L. Rutkowski, Ed. Berlin, Germany: Springer, 2012, pp. 145–153.

[14] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[15] S. Zhao and P. N. Suganthan, "Empirical investigations into the exponential crossover of differential evolutions," *Swarm Evol. Comput.*, vol. 9, pp. 27–36, Apr. 2013. doi: 10.1016/j.swevo.2012.09.004.

[16] A. Iorio and X. Li, "Rotationally invariant crossover operators," in *Simulated Evolution and Learning* (Lecture Notes in Computer Science), vol. 4247, T. D. Wang, ED. Berlin, Germany: Springer, 2006, pp. 310–317.

[17] H.-Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *J. Global Optim.*, vol. 27, no. 1, pp. 105–129, 2003. doi: 10.1023/A:1024653025686.

[18] P. Kaelo and M. M. Ali, "Differential evolution algorithms using hybrid mutation," *Comput. Optim. Appl.*, vol. 37, no. 2, pp. 231–246, Jun. 2007.

[19] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Comput.*, vol. 9, no. 6, pp. 448–462, 2005.

[20] M. G. H. Omran, A. Salman, and A. P. Engelbrecht, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proc. IEEE Congr. Evol. Comput.* Berlin, Germany: Springer-Verlag, Sep. 2005, pp. 192–199.

[21] S. Sayah and A. Hamouda, "A hybrid differential evolution algorithm based on particle swarm optimization for nonconvex economic dispatch problems," *Appl. Soft Comput.*, vol. 13, no. 4, pp. 1608–1619, 2013.

[22] Y.-J. Zheng, X.-L. Xu, H.-F. Ling, and S.-Y. Chen, "A hybrid fireworks optimization method with differential evolution operators," *Neurocomputing*, vol. 148, pp. 75–82, Jan. 2015. doi: 10.1016/j.neucom.2012.08.075.

[23] A. Trivedi, D. Srinivasan, S. Biswas, and T. Reindl, "A genetic algorithm—Differential evolution based hybrid framework: Case study on unit commitment scheduling problem," *Inf. Sci.*, vol. 354, pp. 275–300,Aug. 2016. doi: 10.1016/j.ins.2016.03.023.

[24] Z. Xu, A. Unveren, and A. Acan, "Probability collectives hybridised with differential evolution for global optimisation," *Int. J. Bio-Inspired Comput.*, vol. 8, no. 3, pp. 133–153, May 2016.

[25] Y. Cai, M. Zhao, J. Liao, T. Wang, H. Tian, and Y. Chen, "Neighborhood guided differential evolution," *Soft Comput.*, vol. 21, no. 16, pp. 4769–4812, Aug. 2016.

[26] B. Dorronsoro and P. Bouvry, "Improving classical and decentralized differential evolution with new mutation operator and population topologies," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 67–98, Jun. 2011.

[27] V. Noroozi, A. B. Hashemi, and M. R. Meybodi, "CellularDE: A cellular based differential evolution for dynamic optimization problems," in *Adaptive and Natural Computing Algorithms* (Lecture Notes in Computer Science), vol. 6593, A. Dobnikar, U. Lotrič, and B.Šter, Eds. Berlin, Germany: Springer, 2011.

[28] N. Noman and H. Iba, "Cellular differential evolution algorithm," in *Advances in Artificial Intelligence Lecture Notes in Computer Science*, vol. 6464. Berlin, Germany: Springer, 2010, pp. 293–302.

[29] B. Dorronsoro and P. Bouvry, "Differential evolution algorithms with cellular populations," in *Proc. Int. Conf. Parallel Problem Solving Nature*, 2010, pp. 320–330.

[30] J. Liao, Y. Cai, T. Wang, H. Tian, and Y. Chen, "Cellular direction information based differential evolution for numerical optimization: An empirical study," *Soft Comput.*, vol. 20, no. 7, pp. 2801–2827, Jul. 2016.

[31] J. Liao, Y. Cai, Y. Chen, T. Wang, and H. Tian, "Differential evolution enhanced with composite population information based mutation operators," *J. Digit. Inf. Manage.*, vol. 13, no. 6, pp. 1051–1056, Aug. 2015.

[32] M. Weber, F. Neri, and V. Tirronen, "A study on scale factor in distributed differential evolution," *Inf. Sci.*, vol. 181, no. 12, pp. 2488–2511, Jun. 2011.

[33] M. Epitropakis, D. Tasoulis, N. Pavlidis, V. Plagianakos, and M. Vrahatis, "Enhancing differential evolution utilizing proximity-based mutation operators," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 99–119, Feb. 2011.

[34] W. Gong and Z. Cai, "Differential evolution with ranking-based mutation operators," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 2066–2081, Dec. 2013.

[35] B.-Y. Qu, P. N. Suganthan, and J.-J. Liang, "Differential evolution with neighborhood mutation for multimodal optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 5, pp. 601–614, Oct. 2012.

[36] W. Gong, Z. Cai, and C. X. Ling, "DE/BBO: A hybrid differential evolution with biogeography-based optimization for global numerical optimization," *Soft Comput.*, vol. 15, no. 4, pp. 645–665, 2010.

[37] J. Liang, B. Qu, X. Mao, B. Niu, and D. Wang, "Differential evolution based on fitness Euclidean-distance ratio for multimodal optimization," *Neurocomputing*, vol. 137, pp. 252–260, Aug. 2014.

[38] D. G. M. Miller and I. Hampton, "Krill aggregation characteristics: Spatial distribution patterns from hydroacoustic observations," *Polar Biol.*, vol. 10, no. 2, pp. 125–134, Nov. 1989. doi: 10.1007/BF00239157.

[39] G. Wang, A. H. Gandomi, and A. H. Alavi, "A chaotic particle-swarm krill herd algorithm for global numerical optimization," *Kybernetes*, vol. 42, no. 6, pp. 962–978, 2013. doi: 10.1108/K-11-2012-0108.

[40] S. Saremi, S. Mirjalili, and S. Mirjalili, "Chaotic krill herd optimization algorithm," *Procedia Technol.*, vol. 12, pp. 180–185, Jan. 2014. doi: /10.1016/j.protcy.2013.12.473.

[41] E. Fattahi, M. Bidar, and H. R. Kanan, "Fuzzy Krill herd optimization algorithm," in *Proc. 1st Int. Conf. Netw. Soft Comput. (ICNSC)*, Aug. 2014, pp. 423–426.

[42] L. M. Abualigah, A. T. Khader, E. S. Hanandeh, and A. H. Gandomi, "A novel hybridization strategy for krill herd algorithm applied to clustering techniques," *Appl. Soft Comput.*, vol. 60, pp. 423–435, Nov. 2017. doi: 10.1016/j.asoc.2017.06.059.

[43] M. Azzali, J. Kalinowski, G. Lanciani, and G. Cosimi, "Characteristic properties and dynamic aspects of krill swarms from the ross sea," in *Ross Sea Ecology*, F. M. Faranda, L. Guglielmo, A. Ianora, Eds. Berlin, Germany: Springer, 2000, pp. 413–431.

[44] G. O. Eddie and R. Fao, "Southern ocean Fisheries survey programme. The harvesting of krill," Food Agricult. Org. United Nations, Rome, Italy, Tech. Rep. GLO/SO/77/2,1-76, 1977.

[45] W. M. Hamner, P. P. Hamner, S. W. Strand, and R. W. Gilmer, "Behavior of antarctic krill, euphausia superba: Chemoreception, feeding, schooling, and molting," *Science*, vol. 220, no. 4595, pp. 433–435, Apr. 1983. doi: 10.1126/science.220.4595.433.

[46] M. Zhou and M. E. Huntley, "The principle of biological attraction, demonstrated by the bio-continuum theory of zooplankton patch dynamics," *J. Mar. Res.*, vol. 54, no. 5, pp. 1017–1037, Sep. 1996. doi: 10.1357/0022240963213619.

[47] L. Tang, Y. Dong, and J. Liu, "Differential evolution with an individual-dependent mechanism," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 560–574, Aug. 2015.

[48] L. Cui, G. Li, Z. Zhu, Q. Lin, K.-C. Wong, J. Chen, N. Lu, and J. Lu, "Adaptive multiple-elites-guided composite differential evolution algorithm with a shift mechanism," *Inf. Sci.*, vol. 422, pp. 122–143, Jan. 2018.

[49] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, Jun. 2013, pp. 71–78.

[50] L. M. Zheng, S. X. Zhang, K. S. Tang, and S. Y. Zheng, "Differential evolution powered by collective information," *Inf. Sci.*, vol. 399, pp. 13–29, Aug. 2017.

[51] L. Cui, Q. Huang, G. Li, S. Yang, Z. Ming, Z. Wen, N. Lu, and J. Lu, "Differential evolution algorithm with tracking mechanism and backtracking mechanism," *IEEE Access*, vol. 6, pp. 44252–44267, 2018. doi: 10.1109/ACCESS.2018.2864324.

[52] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009. doi: 10.1109/TEVC.2009.2014613.

[53] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 55–66, Feb. 2011.

[54] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.

[55] Z. Meng, J.-S. Pan, and L. Kong, "Parameters with adaptive learning mechanism (PALM) for the enhancement of differential evolution," *Knowl.-Based Syst.*, vol. 141, pp. 92–112, Feb. 2017.

[56] J. Liang, Y. B. Qu, P. N. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization technical report," Nanyang Technol. Univ., Singapore, Tech. Rep. 201212, 2013.

[57] G. Wu, R. Mallipeddi, P. N. Suganthanc, R. Wang, and H. Chen, "Differential evolution with multi-population based ensemble of mutation strategies," *Inf. Sci.*, vol. 329, pp. 329–345, 2016.
[58] L. Cui and G. Li, "Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations," *Comput. Oper. Res.* vol. 67, pp. 155–173, Mar. 2016.
[59] W. Gao, G. G. Yen, and S. Liu, "A cluster-based differential evolution with self-adaptive strategy for multimodal optimization," *IEEE Trans. Cybern.*, vol. 44, no. 8, pp. 1314–1327, Aug. 2014.

KUN MIAO received the Ph.D. degree from Central South University, China, in 2011, where he is currently an Associate Professor with the School of Civil Engineering. He led some projects such as the National Natural Science Foundation and the Hunan Provincial Natural Science Foundation of China projects. His research interests include intelligence computing, optimization algorithms for large-scale problems, automatic determination of route locations for three-dimensional highways or railways, computer-aided design, design of geographic information systems, and sensor relocation.

ZIYANG WANG was born in Zhengzhou, China, in 1991. He is currently pursuing the master's degree in road and railway engineering with Central South University, China. His research interests include swarm intelligence computing, optimization algorithms for large-scale problems, and the automatic determination of route locations for three-dimensional highways or railways.