Soft Comput (2013) 17:819–834
DOI 10.1007/s00500-012-0955-9

METHODOLOGIES AND APPLICATION

Neighborhood field for cooperative optimization

Zhou Wu · Tommy W. S. Chow

Published online: 6 December 2012

Abstract Inspired by the biological evolution, local cooperation behaviors have been modeled in function optimizations for providing effective search methods. This paper proposes a new meta-heuristic algorithm named Neighborhood Field Optimization algorithm (NFO), which totally utilizes the local cooperation of individuals. This paper also analyzes how the local cooperation helps optimization, which is modeled as the neighborhood field. The proposed NFO is compared with other widely used evolutionary algorithms in intensive simulation under different benchmark functions. The presented results show that NFO is able to solve multimodal problems globally, and thus the cooperation behavior is proven its significance to model a search method.

Keywords Local search · Neighborhood field · Contour gradient optimization · Evolutionary algorithm

1 Introduction

Optimization is the process of finding the most promising solution for a given problem. It requires maximizing or minimizing an objective function efficiently and globally in multi-dimensional search space. Optimization has always been one of the most important topics in science and engineering. In research areas, such as machine learning, artificial intelligence and complex systems, many problems can be regarded as optimization problems. Certain local search techniques, such as gradient-based search (Greiner 1996; Kirkpatrick et al. 1983) and tabu search algorithms (Glover 1990), are successfully applied in conventional applications. But these local algorithms are easily getting trapped in local optima when dealing with much complicated problems.

Some global search algorithms have been proposed to explore the search space stochastically. They usually have random choices of direction to approach good solutions, such as particle swarm optimization (PSO) (Eberhart and Kennedy 1995; Kennedy and Eberhart 1942), genetic algorithm (GA) (Goldberg 1989; Vose 1999), differential evolution (DE) (Storn and Price 1997; Lampinen and Storn 2004) and self-organizing migrating algorithm (SOMA) (Zelinka 2004). Based on the population evolvement, these algorithms are able to escape from local optima till finding the global optimum. They have become increasingly popular when people need to solve highly complicated problems.

The original PSO algorithm emulates the forging behavior in flocks of birds and schools of fish. In the original PSO, each member, called a particle, learns from the best positions found so far. Each particle adapts itself towards the best positions found by the population and by itself. Because of attraction forces of the two poles, each particle can escape out of its surrounding region and explore the whole search space. It is worth noting that PSO has a similar structure with a fully connected network, in which each particle is connected with others for sharing the fitness information (Kennedy and Mendes 2002). In other words, each particle is required to acquire the knowledge of the population with full connections.

Communicated by F. Herrera.

Z. Wu (✉) · T. W. S. Chow
Department of Electronic Engineering,
City University of Hong Kong, Hong Kong, China
e-mail: wuzhsky@gmail.com

T. W. S. Chow
e-mail: eetchow@cityu.edu.hk

The mechanism of GA is inspired by basic principles of natural evolution. In GA, the offspring population is generated from the parent population with three operators, i.e., selection, crossover and mutation. Each individual has the chance of falling into the breeding pool according to its fitness value. Then the crossover and mutation operations are used to reproduce the offspring. For the crossover, individuals in the breeding pool are recombined to explore the search space. For the mutation, the individual randomly exploits its surrounding region to generate new individuals. As GA has combined the global exploration and local exploitation, it has the ability of escaping from local optima. It can be noticed that in selection operation each individual is also required to know the population's fitness, which is equivalent to a fully connected structure.

Differential evolution (DE) is another global search algorithm proposed by Storn and Price. DE generates offspring population by perturbing parents using differential vectors of two random individuals. In DE, the search step is self-adapted along the evolving process because of the balance between the exploitation and the exploration. In respect of the population structure, DE can be seen with a random connected structure. In (Vesterstrom and Thomsen 2004), DE is compared with PSO and some EAs on a large suite of benchmark functions and the results show that DE would outperform PSO and EAs in their test suite.

In all, these global search algorithms are less easily getting trapped in local optima than local search algorithms. But the global search algorithms need much more computation time to find the global optimum than the local search algorithms. It is still necessary that researchers make their efforts in the local search heuristics. First, the local search methods can find the global optimum efficiently when starting from a promising solution with fast convergence speed. But the global search algorithms need more computation time to converge from the same starting point. Second, the local search has emulated the local phenomenon in the nature. In biological communities, individual often communicates with its neighbors within limited ranges of seeing and hearing. These individuals are apt to collect the information in their surrounding regions and to exchange the information with their neighbors. Indeed individuals in real world are mostly affected by the local environment rather than the global environment. The local search methods, which search offspring in the neighborhood region, can emulate this local phenomenon. In contrast, the PSO and GA do not accord with the local phenomenon, because they pre-require the global knowledge of the population.

Some researchers have tried to incorporate the neighborhood exploitation into the global search for acceleration the process of convergence. A local version of PSO has been proposed in (Shi and Eberhart 1998; Eberhart et al. 2001), in which particles only share the information with their neighbors. Each particle moves toward their neighbors' best solutions. A multi-agent GA (MAGA) combined the multi-agent system and GA together to solve the numerical optimization problems (Zhong et al. 2004). In MAGA, each agent lives in a lattice-shape environment and interacts with its neighbors competitively and cooperatively. The local PSO and MAGA have compensated the inadequacy of local heuristic in original PSO and GA. But their local search in their models cannot work separately to deliver promising results. Generally, the local search strategies to date may not have achieved comparable results with the global algorithms like GA, PSO and DE. In our recent work (Wu and Chow 2012), a local algorithm called contour gradient optimization (CGO), was proposed using concepts of contour and gradient in the local region. In (Wu and Chow 2012), the local information is modeled for numerical optimization as neighborhood field model (NFM). The results demonstrated that CGO could outperform several recent PSO versions when solving widely used benchmark functions.

CGO emulated to contour the search space using a sorting approach, in which approximate gradients can provide effective directions of search. But it is problem-dependent to determine the number of contour levels in the sorting. In this paper we propose a new algorithm directly using the cooperation of neighbors, called neighborhood field optimization algorithm (NFO). NFO utilizes exact neighbors to generate directions of search without the need of sorting the population. In NFO, each individual is only sharing information with its neighbors in the search space, and it will be attracted by its superior neighbor and repulsed by its inferior neighbor towards fitter regions. The resultant directions can approximate descending directions of the objective function; NFO can maintain the population with a heterogeneous structure. The comprehensive experimental studies show that NFO has excellent ability to optimize multimodal problems.

The rest of the paper is organized as follows: Section 2 describes the related work about optimization algorithms based on vector field. Section 3 gives the procedure of NFO and analyzes its characteristics. Section 4 reports the simulation results of the proposed algorithm. Finally, Sect. 5 summarizes this paper.

2 Related work

In this section, some related work on modeling the vector field for optimization is introduced. NFM models the local cooperation as field. Based on NFM, CGO algorithm has been proposed for numerical optimization. PSO and DE are

also introduced, which have been used as benchmarks in the experimental studies.

2.1 Neighborhood field model

Neighborhood field model (Wu and Chow 2012) was proposed to emulate the cooperation behavior in a local environment. It is worth noting that agents in real-world networks are likely to cooperate with their neighbors in the local environment rather than with all individuals in the global environment. So NFM states that an agent is influenced by its superior neighbors positively and by its inferior neighbors negatively, which is similar with the potential field model (PFM) (Khatib 1986; Barraquand et al. 1992). The PFM was often used in robotic applications, such as the robots' navigation and obstacle avoidance. In PFM, a robot moves in a field to approach the target without collision. The target position is an attractive pole of the robot, and the obstacles are the repulsive surfaces shown in Fig. 1. The overall force acting upon a robot is composed of the target's attraction force and the obstacle's repulsion force. NFM models that each individual $\boldsymbol{x}_i$ behaves like a robot regarding superior neighbors as targets to follow and inferior neighbors as obstacles to evade. The neighbor field of the individual $\boldsymbol{x}_i$ is driven by a single target and a single obstacle as

$$\mathbf{NF}_i = \Phi(\boldsymbol{xc}_i - \boldsymbol{x}_i) - \Phi(\boldsymbol{xw}_i - \boldsymbol{x}_i), \tag{1}$$

where $\mathbf{NF}_i$ is the overall force driving on the $\boldsymbol{x}_i$, $\boldsymbol{xc}_i$ is the superior neighbor, $\boldsymbol{xw}_i$ is the inferior neighbor and $\Phi(\cdot)$ is the dynamical force function related with the position difference. In the right-hand side of Eq. 1, the first component represents the attractive force of the superior neighbor and the second component represents the repulsive force of the inferior neighbor.

2.2 Contour gradient optimization algorithm

Based on the NFM, CGO algorithm was proposed for numerical optimization in Wu and Chow (2012). To minimize a single objective minimization problem $y = f(\boldsymbol{x})$, a population of N individuals cooperates to search the global optimum. At each generation, these individuals are ranked by their fitness values from the best to the worst. Then they are sorted into m level sets evenly with N/m individuals in each level. The individuals in the same level are regarded to have around the same scale of fitness values.

In CGO, each individual evolves according to the local information of nearest individuals in the neighboring levels. They are called contour neighbors, which are

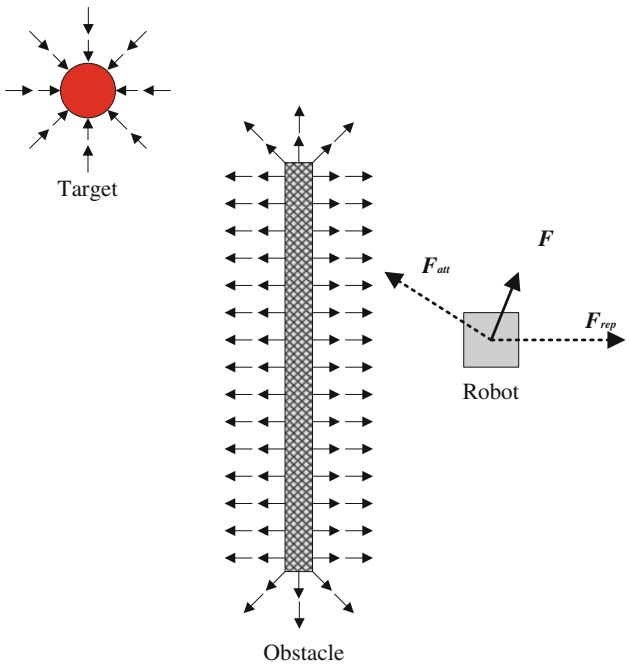


Fig. 1 The potential field model: a robot is driven by the attractive force of target and the repulsive force of the obstacle

calculated as Eq. 2. Base on NFM, CGO proceeds as follows:

1. Initialization: randomize the initial N individuals in the search space.
2. Contouring: at the generation G, rank all individuals by their function value in ascendant order and sort them into m levels. We denote the ith individual $\boldsymbol{x}_{i,G}$'s level number as $L(\boldsymbol{x}_{i,G})$. For each individual$\boldsymbol{x}_{i,G}$, recognize the superior contour neighbor $\boldsymbol{xc}_{i,G}$ in the level $L(\boldsymbol{x}_{i,G}) - 1$ and the inferior contour neighbor $\boldsymbol{xw}_{i,G}$ in the level $L(\boldsymbol{x}_{i,G}) + 1$ as Eq. 2. If $\boldsymbol{x}_{i,G}$ is in the first level, $\boldsymbol{xc}_{i,G}$ is defined as $\boldsymbol{x}_{i,G}$. If $\boldsymbol{x}_{i,G}$ is in the last level, $\boldsymbol{xw}_{i,G}$ is defined as $\boldsymbol{x}_{i,G}$

$$\begin{cases} \boldsymbol{xc}_{i,G} = \arg \min\limits_{L(\boldsymbol{x}_{k,G})=L(\boldsymbol{x}_{i,G})-1} \left\|\boldsymbol{x}_{k,G} - \boldsymbol{x}_{i,G}\right\| \\ \boldsymbol{xw}_{i,G} = \arg \min\limits_{L(\boldsymbol{x}_{k,G})=L(\boldsymbol{x}_{i,G})+1} \left\|\boldsymbol{x}_{k,G} - \boldsymbol{x}_{i,G}\right\| \end{cases}. \tag{2}$$

3. Crossover: $\boldsymbol{x}_{i,G}$ as follows:

$$\boldsymbol{u}_{i,G} = \boldsymbol{x}_{i,G} + \alpha \cdot \boldsymbol{s_c} \cdot \boldsymbol{rand} \cdot \left(\boldsymbol{xc}_{i,G} - \boldsymbol{x}_{i,G}\right) - \alpha \cdot \boldsymbol{s_w} \cdot \boldsymbol{rand} \cdot \left(\boldsymbol{xw}_{i,G} - \boldsymbol{x}_{i,G}\right) \tag{3}$$

where $\boldsymbol{rand}$ is a random vector uniformly distributed in $[0, 1]$ and α is the learning rate. $\boldsymbol{s_c}$ and $\boldsymbol{s_w}$ are two D-dimensional random binary vectors generated at each generation as

$$\begin{cases} \boldsymbol{s_c} = \boldsymbol{rand} \le \mathrm{Cr},\ s_{d_c,c} = 1 \\ \boldsymbol{s_w} = \boldsymbol{rand} \le \mathrm{Cr},\ s_{d_w,w} = 1 \end{cases}, \tag{4}$$

where $\boldsymbol{rand}$ is a random vector uniformly distributed in $[0, 1]$, Cr is a constant at the interval $[0, 1]$ called the crossover probability. d_c and d_w are two random integers in $[1, D]$. The $d_c th$component of $\boldsymbol{s_c}$and the d_wth component of $\boldsymbol{s_w}$ are specified as one so that $\boldsymbol{s_c}$ and $\boldsymbol{s_w}$ are nonzero vectors.

4. Selection: in the next generation, the ith individual will be updated as the better one between $\boldsymbol{x}_{i,G}$ and $\boldsymbol{u}_{i,G}$ as

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G}, & \text{if } f(\boldsymbol{u}_{i,G}) \le f(\boldsymbol{x}_{i,G}) \\ \boldsymbol{x}_{i,G}, & \text{otherwise} \end{cases} \tag{5}$$

5. If the stopping criteria are not satisfied, go to step 2.

2.3 Particle swarm optimization

Particle swarm optimization (Eberhart and Kennedy 1995; Kennedy and Eberhart 1942) is a population-based algorithm inspired by the foraging behavior of swarms of animals. In PSO, the population is called a swarm and an individual is called a particle. PSO consists of a swarm of particles searching a D-dimensional real-valued search space. Every particle has a position vector $\boldsymbol{x}_i = [x_{1,i}, x_{2,i}, \ldots, x_{D,i}]^T$ and a velocity vector $\boldsymbol{v}_i = [v_{1,i}, v_{2,i}, \ldots, v_{D,i}]^T$. Each particle has intelligent memory of its own best position $\boldsymbol{x_p}$ and the global best position $\boldsymbol{x_g}$. After sharing memory with each other, the information about good areas in the search space can spread through the swarm. When a certain particle finds the best solution, other particles are informed and move toward the best solution with an adaptive velocity. The new position and velocity of the ith particle are updated by

$$\begin{aligned} \boldsymbol{v_i} = {} & \omega \cdot \boldsymbol{v_i} + c_1 \cdot \boldsymbol{rand} \cdot (\boldsymbol{x_p} - \boldsymbol{x_i}) \\ & + c_2 \cdot \boldsymbol{rand} \cdot (\boldsymbol{x_g} - \boldsymbol{x_i}), \end{aligned} \tag{6}$$

$$\boldsymbol{x_i} = \boldsymbol{x_i} + \boldsymbol{v_i}, \tag{7}$$

where ω is called the inertia weight, c_1, c_2 are positive learning rates that determine the significance of $\boldsymbol{x}_p$ and $\boldsymbol{x}_g$ and $\boldsymbol{rand}$ is a random vector uniformly distributed in the interval of [0, 1]. The particle is driven from the previous position to the new position with the updated velocity. There is another local PSO version different from the original PSO in the mutation step (Kennedy and Mendes 2002; Vesterstrom and Thomsen 2004). In the local PSO, each particle does not learn from the best position in the population, but learns from its neighbors as,

$$\begin{aligned} \boldsymbol{v_i} = {} & \omega \cdot \boldsymbol{v_i} + c_1 \cdot \boldsymbol{rand} \cdot (\boldsymbol{x_p} - \boldsymbol{x_i}) \\ & + c_2 \cdot \boldsymbol{rand} \cdot (\boldsymbol{x}_{lbest} - \boldsymbol{x_i}), \end{aligned} \tag{8}$$

where $\boldsymbol{x}_{lbest}$ is the best position found by its neighbors. The other parameters in Eq. 8 are the same with the original PSO.

2.4 Differential evolution

Differential evolution algorithm was introduced in (Storn and Price 1997; Lampinen and Storn 2004). It resembles the structure of the traditional EAs, but differs from them in generating new candidate solutions and selecting the offspring. DE is a population-based stochastic optimization algorithm, which takes the difference of two randomly chosen parameter vectors to perturb an existing vector. DE mutates and recombines the population to produce a population of trial vectors. At the generation G, each individual is encoded as $\boldsymbol{x}_{i,G} = [x_{1,i,G}, x_{2,i,G}, \ldots, x_{D,i,G}]^T$ (called target vectors), and its corresponding trial vector after perturbation is denoted as $\boldsymbol{u}_{i,G} = [u_{1,i,G}, u_{2,i,G}, \ldots, u_{D,i,G}]^T$. The mutation and crossover operations are shown as the following:

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{r0,G} + F \cdot (\boldsymbol{x}_{r1,G} - \boldsymbol{x}_{r2,G}), \tag{9}$$

$$u_{j,i,G} = \begin{cases} v_{j,i,G}, & \text{if } rand(0,1) \le \mathrm{Cr} \text{ or } j = j_{rand} \\ x_{j,i,G}, & \text{otherwise} \end{cases}, \tag{10}$$

where $r0$, $r1$ and $r2$ are three distinct random indices; $\boldsymbol{x}_{r0}$, $\boldsymbol{x}_{r1}$, $\boldsymbol{x}_{r2}$ are three random individuals in the population. The mutant vector $\boldsymbol{v_i}$ is generated in the mutation, $F \in [0, 1]$ is the mutation rate. The trail vector $\boldsymbol{u_i}$ is obtained by recombining $\boldsymbol{x_i}$ and $\boldsymbol{v_i}$ with a crossover probability Cr as Eq. 10. $rand(0, 1)$ is a uniformly distributed random number in the scale of [0, 1] which is independently generated for each j at each generation. j_{rand} is a randomly chosen integer in $[1, D]$ to accept the new mutant vector so that the trial vector is different from the target vector. After the mutation and crossover, the fitter solution between $\boldsymbol{x_i}$ and $\boldsymbol{u_i}$ will be selected into the next generation as

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G}, & \text{if } f(\boldsymbol{u}_{i,G}) \le f(\boldsymbol{x}_{i,G}) \\ \boldsymbol{x}_{i,G}, & \text{otherwise} \end{cases}, \tag{11}$$

where $f(\cdot)$ is the object function to be minimized.

3 Neighborhood field optimization

Many single objective optimization algorithms aim to find the global optimum efficiently in a given period. This challenge requires that optimization algorithms are able to escape local optima. Some global search algorithms, such as GA and PSO, are not easily getting trapped by the local optima, but fail to solve some difficult problems efficiently.

One main reason is their lack of local search, which may help to accelerate the search process. CGO has been proposed to utilize the local search in the optimization and has delivered promising results. CGO needs to divide the population in m levels, but the value of m should be fine-tuned for different problems. This paper newly proposes NFO algorithm without any additional parameter by utilizing the cooperation of nearest neighbors directly.

3.1 Neighborhood field optimization algorithm

Neighborhood *f*ield *o*ptimization *a*lgorithm is a population-based algorithm, in which each individual is updated under the concept of "learning from the neighbors" mentioned in NFM. The detailed procedure of NFO algorithm for minimization problems is illustrated as follows:

1. Initialization: randomize the initial N individuals, which are sampled uniformly in the search space.
2. Localization: for each individual $\boldsymbol{x}_{i,G}$ at the generation G, find the superior neighbor $\boldsymbol{xc}_{i,G}$ and the inferior neighbor $\boldsymbol{xw}_{i,G}$ (in the search space) as

$$\begin{cases} \boldsymbol{xc}_{i,G} = \arg \min\limits_{f(\boldsymbol{x}_{k,G}) < f(\boldsymbol{x}_{i,G})} \left\| \boldsymbol{x}_{k,G} - \boldsymbol{x}_{i,G} \right\| \\ \boldsymbol{xw}_{i,G} = \arg \min\limits_{f(\boldsymbol{x}_{k,G}) > f(\boldsymbol{x}_{i,G})} \left\| \boldsymbol{x}_{k,G} - \boldsymbol{x}_{i,G} \right\| \end{cases}, \tag{12}$$

   where $\boldsymbol{xc}_{i,G}$ is the superior neighbor with the function value smaller than $f(\boldsymbol{x}_{i,G})$ and $\boldsymbol{xw}_{i,G}$ is the inferior neighbor with a larger function value (for minimization problems).$\|\cdot\|$ is the distance evaluation (Euclidean distance is used). If $\boldsymbol{x}_{i,G}$ is in the best individual in the population, $\boldsymbol{xc}_{i,G}$ is defined as $\boldsymbol{x}_{i,G}$. If $\boldsymbol{x}_{i,G}$ is in the worst individual in the population, $\boldsymbol{xw}_{i,G}$ is defined as $\boldsymbol{x}_{i,G}$
3. Mutation: perturb each individual as

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{i,G} + \alpha \cdot \boldsymbol{rand} \cdot \left(\boldsymbol{xc}_{i,G} - \boldsymbol{x}_{i,G}\right) + \alpha \cdot rand \cdot \left(\boldsymbol{xc}_{i,G} - \boldsymbol{xw}_{i,G}\right), \tag{13}$$

   where $\boldsymbol{xc}_{i,G}$ is the superior neighbor, $\boldsymbol{xw}_{i,G}$ is the inferior neighbor, $\boldsymbol{rand}$ is a random vector in [0, 1], α is the learning rate. and $\boldsymbol{v}_{i,G}$ is the obtained mutant vector.
4. Crossover: recombine the mutation vector with the target vector $\boldsymbol{x}_i$.

$$u_{j,i,G} = \begin{cases} v_{j,i,G}, & if \;\; rand(0,1) \le Cr \; or \; j = j_{rand} \\ x_{j,i,G}, & otherwise \end{cases}, \tag{14}$$

   where $j = 1, 2, \ldots, D$ is the dimension index, Cr is the crossover probability, $rand(0,1)$ is a uniformly distributed random number in the scale of [0, 1] and j_{rand} is a random component to accept the new mutant vector so that the trial vector is different from the target vector.
5. Selection: in the next generation the ith individual will select the better solution between $\boldsymbol{x}_{i,G}$ and $\boldsymbol{u}_{i,G}$ as Eq. 5.
6. If the stopping criteria are not satisfied, go to step 2.

NFO has two control parameters α and Crthat need to be tuned by users. We have evaluated the effects of the parameters in the experimental studies. The optimal settings of the two parameters lie in $\alpha \in [0.7, 1.7]$ and $\mathrm{Cr} \in [0.1, 0.7]$. It can also be noticed that the two parameters are robust for the evaluated problems. As the neighbors considered in NFO are close with the updated individual with an adjusting distance, NFO exhibits its ability to adapt the search behavior between the local exploitation and the global exploration.

3.2 Analysis of neighborhood field

Neighborhood Field Optimization has emulated the local phenomenon in the real world that each individual is mainly affected by its neighbors. In Fig. 2, the superior neighbor in the search space has the attractive field, and the inferior neighbor has the repulsive field. The philosophy of NFO is learning from neighbors, which can be reflected in some real-world examples. One typical example is the internet network. Via the internet, people can easily connect with emails or web sites. Although it is not feasible that a university student can directly connect with certain national celebrities, he may in fact connect with these celebrities indirectly via local cooperation. He can first connect to the nearby persons, such as the university president or a city major, who may know more celebrities. In this way, after several local connections, he may connect to the target person, as described in the small world model (Watts and Strogatz 1998). Similarly NFO shows some local characteristics of population structures as follows:

First, the direction of neighborhood field can approximate a descending direction of the objective function.

Lemma 1 *Assume that $\boldsymbol{x}_i$ is not a local optimum, $\boldsymbol{xw}_i$ and $\boldsymbol{xc}_i$ are two neighbors of $\boldsymbol{x}_i$ in a given local region. $\boldsymbol{xc}_i$ is the superior neighbor, and $\boldsymbol{xw}_i$ is the inferior neighbor. The neighborhood field in NFO can approximate the inverse direction of the gradient $\nabla f(\boldsymbol{x}_i)$ as*

$$\begin{cases} -\nabla f(\boldsymbol{x}_i)^T (\boldsymbol{xc}_i - \boldsymbol{x}_i) \ge 0 \\ -\nabla f(\boldsymbol{x}_i)^T (\boldsymbol{xw}_i - \boldsymbol{x}_i) \le 0 \end{cases}, \tag{15}$$

Proof Since $\boldsymbol{xc}_i$ is the neighbor close to $\boldsymbol{x}_i$, using the linear approximation we can obtain

$$f(\boldsymbol{xc}_i) \approx f(\boldsymbol{x}_i) + \nabla f(\boldsymbol{x}_i)^T (\boldsymbol{xc}_i - \boldsymbol{x}_i)$$

Then

$$-\nabla f(\boldsymbol{x}_i)^T (\boldsymbol{xc}_i - \boldsymbol{x}_i) \approx f(\boldsymbol{x}_i) - f(\boldsymbol{xc}_i).$$

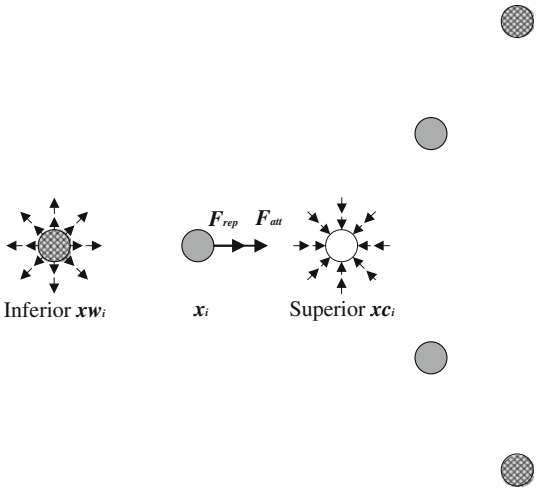


Fig. 2 In NFO, the individual is attracted by its superior neighbor and is repulsed by its inferior neighbor. The *blank node* means the superior individual, and the striped nodes mean those inferior individuals. The *shaded node* x_i is influenced by its neighbors

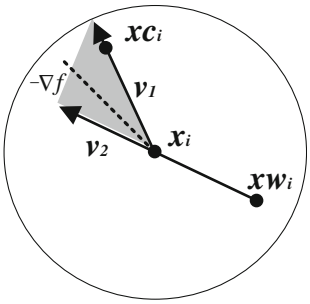


Fig. 3 The direction of neighborhood field in NFO in a 2-D example. xc_i and xw_i are the superior and inferior neighbors in a *small circle region*. The differential vectors of $xc_i - x_i$ and $x_i - xw_i$ are denoted as v_1 and v_2. The *shaded area* between v_1 and v_2 is likely to include the inverse direction of the gradient at x_i, which is denoted as the *dashed arrow*

Since $f(\boldsymbol{xc}_i) \leq f(\boldsymbol{x}_i)$, then $-\nabla f(\boldsymbol{x}_i)^T(\boldsymbol{xc}_i - \boldsymbol{x}_i) \geq 0$. Therefore, the first component can be obtained. The second component can be proven in the same way. So the neighborhood field $\mathbf{NF}_i$ satisfies $-\nabla f(x_i)^T \cdot \mathbf{NF}_i \geq 0$.

According to *Lemma 1*, the differential vectors $xc_i - x_i$, $x_i - xw_i$ are similar to the inverse gradient at x_i based on the cosine similarity. It is reasonable that we can expect these vectors as a tool of approximating the gradient, especially for some engineering applications with complicated format of gradient functions. Especially when the two neighbors are the best and worst individuals in the neighborhood region, the two differential directions are precise on the gradient. Therefore, in NFO the neighborhood field can approximate a descending direction of the objective functions shown in Fig. 3. The neighborhood field is close to the inverse gradient direction.

Second, NFO can generate a particular population structure with heterogeneous property. The heterogeneous structure means that most individuals have effects in small search regions and only a few individuals have effects in large search regions. In (Kennedy et al. 2002), it is stated that the heterogeneous structure can balance the local exploitation and the global exploration with better performance than some regular structures. To visualize the structure in NFO, we can regard the population as a graph $G(V, E)$, in which nodes V represent the individuals and edges E represent the neighborhood of two individuals, i.e. $\boldsymbol{x}_i$ and $\boldsymbol{xc}_i$, $\boldsymbol{x}_i$ and $\boldsymbol{xw}_i$, shown in Figs. 4, 5. We graphically analyze the population structures of NFO when solving Sphere function (Fig. 4) and Ackley's function (Fig. 5). For each function, the degree distribution of the population graph is plotted in the 100th iteration and the 500th iteration, respectively. Most individuals equally have two connections and a few individuals have larger connections in a heterogeneous structure. In other words, NFO can equally consider each individual in the population when learning from the neighbors. It is clear that the heterogeneity property can be maintained during the optimization process. Note that NFO can inherently generate a heterogeneous population structure, which is not deliberately generated using the complex network models in complex neighborhood-based PSO (CNPSO) (Godoy et al. 2009). It can be concluded that the population structure of NFO is not as regular as the fully connected PSO and the ring-shaped PSO. The structure of NFO is also more complex than that of CGO, which has a multi-layer tree structure (according to contour levels).

Due to the above two properties, NFO can effectively enhance the diversity of population when searching the global optimum. We will compare the diversity during the search process between NFO and PSO. The super-fit metric (SF) defined in (Caponio et al. 2009) is used to evaluate the diversity as

$$\chi_k = \frac{|f_{k,\text{best}} - f_{k,\text{avg}}|}{\max_{G=1}^{k} |f_{G,\text{best}} - f_{G,\text{avg}}|}, \tag{16}$$

where $f_{G,\text{best}}$ and $f_{G,\text{avg}}$ are the best and average fitness values in the Gth generation, respectively. In the kth generation, the super-fit metric χ_k is defined as the ratio of the current difference of $f_{k,\text{best}}$ and $f_{k,\text{avg}}$ to the maximum difference found so far. It is clear that χ_k varies between 0 and 1, where $\chi_k = 1$ means a high diversity and $\chi_k = 0$ means a low diversity.

By evaluating the value of SF, we test the population diversity of NFO and PSO on the Sphere function (10-D) and Ackley's function (10-D) in Fig. 6. The figure shows the mean value of SF metric after 25 independent runs. The results illustrate that the NFO can obtain a larger SF metric

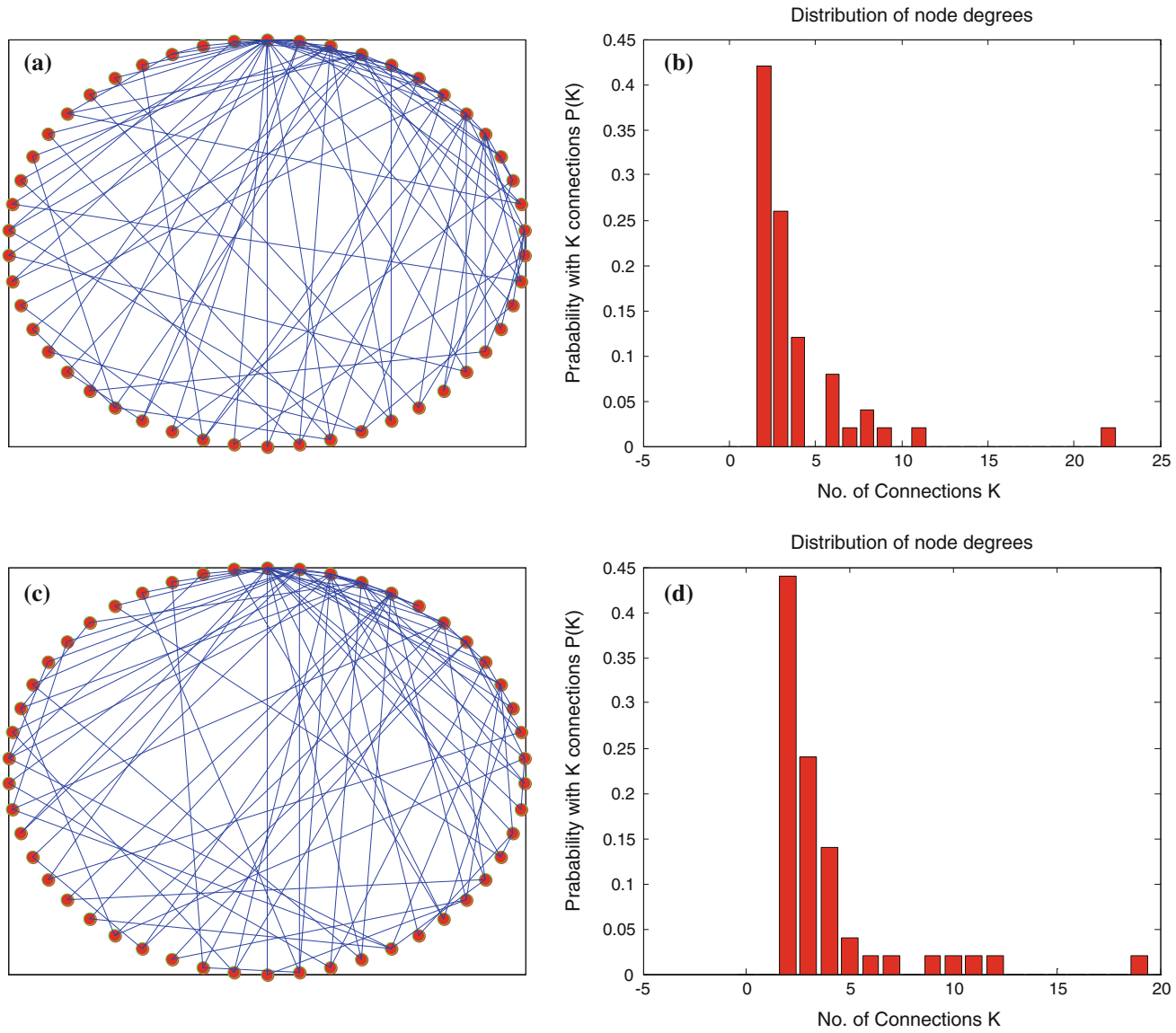


Fig. 4 The population structures of NFO when solving Sphere function (10-D) with 50 individuals. **a** and **b** are population structure and degree distribution in the 100th iteration for Sphere function. **c** and **d** are population structure and degree distribution in the 500th iteration for sphere function

than PSO in the search process. Clearly NFO can maintain better population diversity than PSO.

4 Simulation results

In this section, we compare the performance of NFO with other population-based algorithms PSO, SOMA, DE and CGO through minimizing several benchmarks functions. Table 1 lists the parameter settings of each algorithm in our simulations, which are the same as recommended in literatures (Storn and Price 1997; Eberhart and Shi 2000; Xu and Chow 2010). In PSO, the inertia weight ω is set to 0.72, and c_1, c_2 are set to 1.49. In SOMA, the number of migration k is set to 13, step size Δ is set to 0.11, and the mutation rate PRT is set to 0.1. In DE, the scaling factor F is set to 0.5, and the crossover rate is 0.9. In CGO, the number of individuals in each level N/m is set to 3, the learning rate is set to 1.3, and crossover probability is 0.1. In NFO, the crossover probability is set to 0.1, and the learning rate is set to 1.3. For each algorithm, the population size is set to 30 when solving the 10-dimensional (10-D) functions. The population size is set to 100 when solving the 50-dimensional (50-D) functions.

For the benchmark functions, a set of generalized functions commonly found in literatures (Brest et al. 2006; Xu and Chow 2010) is used to test our algorithms. Among these functions, f_1, f_2 are unimodal functions and $f_3, \ldots, f_7$

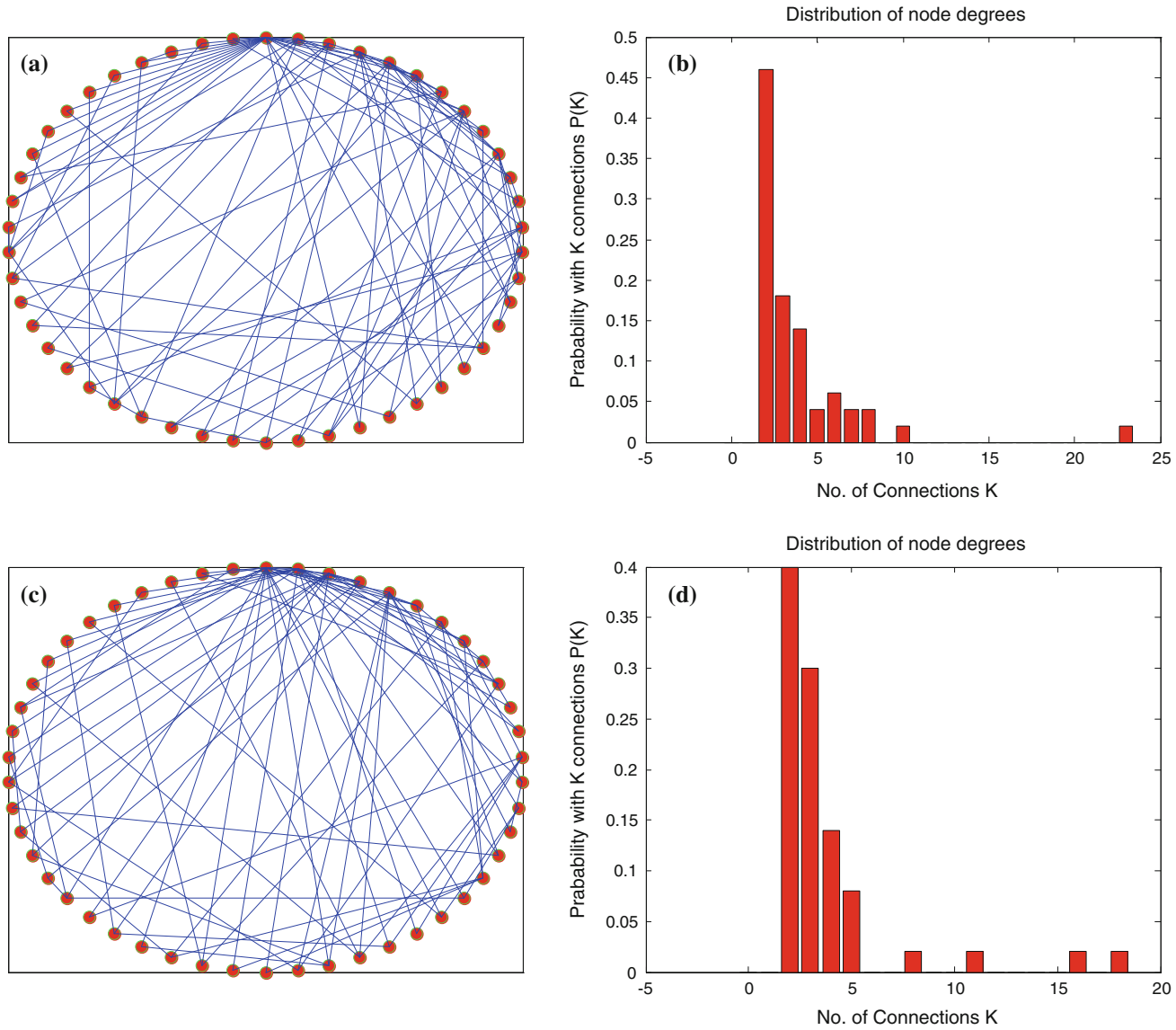
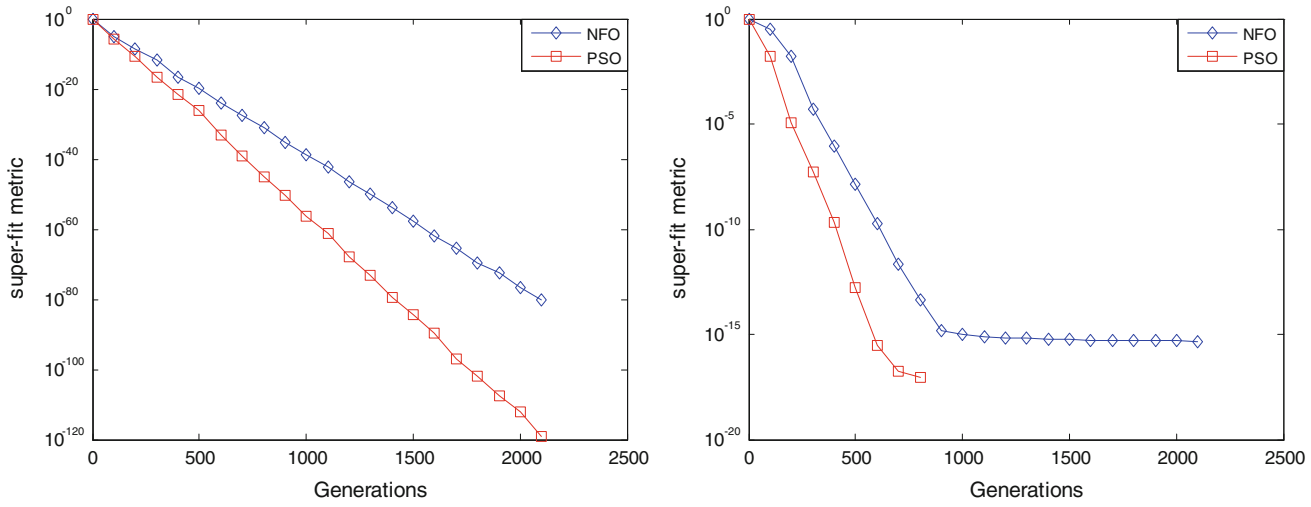


Fig. 5 The population structures of NFO when solving Ackley's function (10-D) with 50 individuals. **a** and **b** are population structure and degree distribution in the 100th iteration for Ackley's function. **c** and **d** are population structure and degree distribution in the 500th iteration for Ackley's function

are un-rotated multimodal functions as Table 2. Table 2 has listed the formula, the optima and the initialization spaces of the seven functions. In addition, another set of functions chosen from CEC 2005 (Suganthan et al. 2005) is also included. The used CEC2005 functions can be divided into three classes: shifted and rotated unimodal functions F_1–F_5; shifted and rotated multimodal functions F_6–F_{12} and expanded multimodal functions F_{13}–F_{14}. CEC2005 functions are shifted by random vectors to make them asymmetric in the search space. For the rotated functions, the original variable $\boldsymbol{x}$ is multiplied with an orthogonal matrix $\boldsymbol{M}$ to obtain the new variable as $\boldsymbol{z} = \boldsymbol{x} * \boldsymbol{M}$ (Salomon 1996; Liang et al. 2006). Note that this paper evaluates each algorithm on 10- and 50-dimensional test functions. For each algorithm and each function, 25 independent runs were conducted for calculating the mean values and the standard derivations. All runs are terminated when they meet the maximum number of fitness evaluations (FEs). The maximum number of FEs is set to be 100,000 for 10-D functions and 500,000 for 50-D functions. For clarity, the results of the best and second best algorithms are marked in boldface and italic; if not all or most algorithms produce identical results.

4.1 Parameter evaluations of NFO

There are two parameters in NFO, the learning rate α and the crossover probability Cr. The effect of α is to control



(a) Sphere function **(b)** Ackley's function

Fig. 6 Comparison of population diversity between NFO and PSO, in which a large SF mean a high diversity. **a** Mean value of SF when solving Sphere function (10-D). **b** Mean value of SF metric when solving Ackley's function (10-D)

Table 1 Parameter setting of algorithms

Algorithms	Parameters
PSO	$\omega = 0.72$, $c_1 = 1.49$, $c_2 = 1.49$
SOMA	$k = 13$, $\Delta = 0.11$, $PRT = 0.1$
DE	$F = 0.5$, Cr $= 0.9$
CGO	$N/m = 2$, $\alpha = 1.3$, Cr $= 0.1$
NFO	$\alpha = 1.3$, Cr $= 0.1$

the step length of moving, and Cr is used to control the convergence speed. For users, the two parameters are expected to be insensitive and problem independent. In this experiment, we have evaluated how α and Cr affect the performance of NFO. Different settings of α and Cr are tested on the 10-D functions $f_1, f_3, f_4, f_5, f_6, f_7$. In this section α is set to 0.3, 0.5, .0.7, 0.9, 1.1, 1.3, 1.5 and 1.7. For each learning rate, Cr is set to 0.1, 0.3, 0.5, 0.7 and 0.9 for comparisons. With each pair of parameters, the mean values of final results are listed in Table 3. In this table, it can be noticed that appropriate α lies in the scale from 0.7 to 1.7. If α is smaller than 0.7, NFO is easily getting trapped in the local minimum due to the insufficient search. Furthermore, it is obvious that Cr should be set in the scale [0.1, 0.7]. If Cr is large than 0.7, NFO cannot converge to the global optimum regardless of the learning rate. The optimal settings of the two parameters are $\alpha = 1.3$ and Cr $= 0.1$. In the following studies, α and Cr are set to 1.3 and 0.1.

4.2 Comparisons with PSO, SOMA, DE and CGO

The means and standard derivations of results obtained by each algorithm are listed in Tables 4 and 5. Tables 4 and 5 report the means and standard derivation for the 10-D and

Table 2 Benchmark functions

Function name	$f_i(x)$	Search space	Minimum
Sphere function	$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-5.12, 5.11]^D$	0
Rosenbrock's function	$f_2(x) = \sum_{i=1}^{D-1} \left(100\left(x_i^2 - x_{i+1}\right)^2 - (x_i - 1)^2\right)$	$[-2.048, 2.047]^D$	0
Rastrigin's function	$f_3(\boldsymbol{x}) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos 2\pi x_i + 10\right)$	$[-5.12, 5.11]^D$	0
Schaffer's function	$f_4(x) = \sum_{i=1}^{D-1} \left(0.5 + \frac{\sin^2\sqrt{x_i^2 + x_{i+1}^2} - 0.5}{\left(1 + 0.001\left(x_i^2 + x_{i+1}^2\right)^2\right)^2}\right)$	$[-2.048, 2.047]^D$	0
Ackley's function	$f_5(x) = 20 + e - 20\exp\left(-0.2\sqrt{1/D\sum_{i=1}^{D} x_i^2}\right) - \exp\left(1/D\sum_{i=1}^{D}\cos 2\pi x_i\right)$	$[-30, 30]^D$	0
Griewank's function	$f_6(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^D$	0
Stretched V sine wave function	$f_7(x) = \sum_{i=1}^{D-1} \left(x_i^2 + x_{i+1}^2\right)^{0.25} \left(1 + \sin^2\left(50\left(x_i^2 + x_{i+1}^2\right)^{0.1}\right)\right)$	$[-10, 10]^D$	0

50-D functions, respectively. These statistics are calculated at the end of the optimization. For certain functions, middle results are also reported because the final results obtained by certain algorithms are identical to zeros or a small error, caused by the precision threshold according to IEEE Standard 754 (such as the results of f_3 and f_5). In these cases, some results at middle generations are compared. The last three rows in the tables summarize the results of Wilcoxon rank-sum tests with significance level 0.05 (Derrac et al. 2011). The statistical tests have been conducted to compare the proposed NFO with PSO, SOMA, DE and CGO. "−" denote that each compared algorithm is worse than NFO; "+" denotes that each compared algorithm is better than NFO" $\approx$ " denotes that each compared algorithm is similar with NFO.

In Tables 6 and 7, we summarize the success rate (SR) and the average number of fitness evaluations over successful runs (FEs) for each algorithm. In this experiment, the success of an algorithm means that the best result is no worse than the specified optimal threshold, i.e., $f(\boldsymbol{x}*) + 1e^{-6}$ for all functions. FEs and SR are useful to illustrate the performance in terms of the convergence rate (in successful runs) and the reliability.

Figure 7 demonstrates the convergence graphs for 50-D problems. Because the convergence graphs of 10-D problems are similar with their 50-D cases, they are omitted here. Note that the convergence graphs are the median run instead of the mean values reported in the tables. The median value can provide additional information when certain algorithms may fail to converge occasionally.

For 10-D functions, statistics in Table 4 indicate that NFO is able to deliver the best performance in terms of accuracy and reliability on most un-rotated functions except f_2 and f_3. In Table 4, we also summarize the number of functions, on which the compared algorithms are worse than, better than and similar with NFO in the Wilcoxon tests. It can be noticed in Table 4 that NFO is significantly better than PSO, SOMA, DE and CGO on 5, 7, 5 and 5 functions, respectively. Table 6 shows that NFO need less FEs when solving f_1, f_4, f_5, f_7, than other algorithms.

Table 3 Mean results with different parameters of NFO (25 runs)

α	Cr = 0.1	Cr = 0.3	Cr = 0.5	Cr = 0.7	Cr = 0.9	α	Cr = 0.1	Cr = 0.3	Cr = 0.5	Cr = 0.7	Cr = 0.9
	f_1						f_3				
0.3	1.90e−07	2.45e−01	1.48e+00	3.88e+00	6.25e+00	0.3	2.07e+00	1.65e+00	2.24e+00	2.79e+00	4.03e+00
0.5	0	9.90e−06	2.24e−02	4.20e−01	1.46e+00	0.5	2.49e−14	1.04e−12	1.16e+00	2.58e+00	3.89e+00
0.7	0	0	0	0	7.89e−31	0.7	0	0	0	3.03e−02	1.71e−01
0.9	0	0	0	0	0	0.9	3.55e−15	3.55e−15	3.55e−15	7.11e−15	1.65e+00
1.1	0	0	0	0	0	1.1	0	0	0	2.15e−02	5.18e−02
1.3	0	0	0	0	0	1.3	3.55e−15	0	0	3.55e−15	3.55e−15
1.5	0	0	0	0	0	1.5	3.55e−15	3.55e−15	3.55e−15	3.55e−15	3.55e−15
1.7	0	0	0	0	0	1.7	0	0	0	0	4.57e−02
	f_4						f_5				
0.3	1.08e−01	5.05e−01	1.07e+00	5.74e+00	1.77e+01	0.3	1.90e+00	1.78e+00	2.48e+00	6.39e+00	1.34e+01
0.5	3.69e−02	9.50e−02	2.20e−01	4.48e−01	2.49e−14	0.5	3.47e−02	1.08e−02	1.08e−02	3.03e−02	5.09e−01
0.7	0	0	9.95e−01	2.98e+00	9.95e+00	0.7	0	0	0	3.68e−10	1.65e−01
0.9	4.06e−08	1.51e−11	9.86e−03	1.23e−02	3.20e−02	0.9	0	0	0	3.03e−02	2.35e−01
1.1	0	0	0	0	2.98e+00	1.1	0	0	0	6.46e−13	1.65e−01
1.3	2.11e−15	2.22e−16	9.64e−13	7.50e−03	4.18e−02	1.3	0	0	0	0	1.07e−02
1.5	4.18e−15	1.31e−13	1.06e−06	1.35e−02	3.94e−02	1.5	0	0	0	0	0
1.7	0	0	0	2.04e+00	1.11e+01	1.7	0	0	0	2.22e−15	2.09−04
	f_6						f_7				
0.3	1.32e+01	1.92e+01	2.41e+01	3.27e+01	4.06e+01	0.3	1.65e−01	1.65e−01	1.98e−01	1.91e−01	2.18e−01
0.5	9.97e−01	1.99e+00	6.99e+00	1.29e+01	2.13e+01	0.5	1.74e−02	8.15e−02	3.67e−01	4.13e−01	7.07e−01
0.7	3.55e−15	3.55e−15	1.49e−13	1.65e+00	2.58e+00	0.7	8.67e−03	9.86e−03	2.22e−02	4.67e−02	1.62e−01
0.9	0	0	9.95e−01	1.99e+00	8.96e+00	0.9	0	0	0	0	2.22e−16
1.1	3.55e−15	3.55e−15	0	3.55e−15	3.55e−15	1.1	1.11e−16	1.99e−13	8.11e−11	9.86e−03	2.71e−02
1.3	0	0	0	2.84e−14	7.83e+00	1.3	0	0	0	0	1.22e−15
1.5	3.55e−15	0	0	1.31e+00	1.15e+01	1.5	1.22e−15	0	0	0	2.22e−16
1.7	3.55e−15	3.55e−15	3.55e−15	0	3.55e−15	1.7	4.43e−09	7.40e−03	7.40e−03	3.45e−02	5.95e−02

Table 4 Means and standard derivations of experimental results in 10-D case (25 runs)

	FEs	PSO mean ± Std. dev.	SOMA mean ± std. dev.	DE mean ± std. dev.	CGO mean ± std. dev.	NFO mean ± std. dev.
f_1	10000	2.61e−12 ± 2.85e−12−	1.22e−11 ± 1.76e−11−	*2.58e−12 ± 1.96e−12−*	8.35e−10 ± 5.93e−10−	**2.75e−17 ± 4.07e−17**
	100000	0±0	0±0	0±0	0±0	0±0
f_2	100000	**1.13e−09 ± 2.41e−09**+	5.84e−01 ± 1.37e+00−	*3.18e−09 ± 2.41e−09*+	5.02e−04 ± 8.98e−04 ≈	2.58e−03 ± 4.30e−03
f_3	10000	**0±0**+	2.19e+00 ± 2.11e+00−	6.63e−02 ± 2.52e−01−	5.16e−06 ± 1.71e−05−	*6.73e−14 ± 3.38e−13*
	100000	0±0	2.19e+00 ± 2.11e+00	6.63e−02 ± 2.52e−01	0±0	0±0
f_4	10000	1.33e−02 ± 5.58e−02−	6.06e−02 ± 1.11e−01−	6.82e−03 ± 3.14e−02−	*6.78e−06 ± 2.85e−05−*	**6.54e−09 ± 3.58e−08**
	100000	5.51e−03 ± 3.02e−02	5.51e−02 ± 1.09e−01	2.41e−16 ± 1.32e−15	*9.90e−14 ± 5.37e−13*	0±0
f_5	10000	4.55e−12 ± 2.57e−12−	2.10e−05 ± 1.06e−05−	*4.05e−12 ± 2.18e−12−*	4.52e−09 ± 2.13e−09−	**4.97e−15 ± 2.00e−15**
	100000	0 ± 0	3.67e−15 ± 1.98e−15	0 ± 0	3.20e−15 ± 1.71e−15	2.25e−15 ± 1.98e−15
f_6	100000	8.22e−03 ± 2.53e−03−	2.45e−02 ± 2.40e−02−	**2.47e−04 ± 1.35e−03**+	*1.46e−03 ± 2.25e−03* ≈	1.48e−03 ± 3.94e−03
f_7	10000	6.89e−05 ± 4.31e−05−	2.52e−02 ± 9.13e−03−	*5.19e−05 ± 2.65e−05−*	5.63e−02 ± 1.80e−02−	**2.08e−06 ± 2.29e−06**
	100000	0±0	1.37e−03 ± 5.80e−03	0 ± 0	0 ± 0	0 ± 0
−		5	7	5	5	
+		2	0	2	0	
≈		0	0	0	2	

Note: "−", "+" and "≈" denote that the performance of each compared algorithm is worse than, better than, and similar with that of NFO, respectively "FEs" means the number of fitness evaluations

Table 5 Means and standard derivations of experimental results in 50-D case (25 runs)

	FEs	PSO mean ± std. dev.	SOMA mean ± std. dev.	DE mean ± std. dev.	CGO mean ± std. dev.	NFO mean ± std. dev.
f_1	60000	3.63e−04 ± 9.31e–04−	1.78e−11 ± 9.76e−12−	*2.04e−10 ± 3.29e−11−*	1.69e−06 ± 2.04e−07−	**1.95e−16 ± 9.40e−17**
	500000	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0
f_2	500000	2.28e+02 ± 6.39e+01−	2.33e+01 ± 3.05e+01−	6.11e+01 ± 1.54e+01−	*1.76e+00 ± 1.50e+00−*	**5.93e−02 ± 6.39e−02**
f_3	60000	1.21e+02 ± 2.76e+01−	5.81e+00 ± 2.41e+00−	**2.27e−14 ± 2.84e−14**+	7.23e+01 ± 2.99e+00−	*5.60e+00 ± 4.86e+00*
	500000	1.21e+02 ± 2.75e+01	5.81e+00 ± 2.41e+00	0 ± 0	2.84e−05 ± 6.93e−05	0 ± 0
f_4	500000	2.20e+00 ± 8.34e−01−	*6.27e−01 ± 3.79e−01−*	2.31e+00 ± 2.69e−01−	1.70e+00 ± 3.08e−01−	**1.10e−10 ± 4.94e−10**
f_5	60000	3.70e+00 ± 7.86e−01−	*9.17e−06 ± 2.46e−06−*	5.04e−05 ± 3.38e−06−	6.66e−03 ± 5.40e−04−	**3.62e−08 ± 9.33e−09**
	500000	3.40e+00 ± 6.88e−01	1.61e−14 ± 2.92e−15	1.69e−14 ± 2.77e−15	4.31e−14 ± 2.37e−15	2.46e−14 ± 2.50e−15
f_6	60000	2.82e−01 ± 3.11e−01−	*1.61e−08 ± 2.50e−08−*	2.54e−07 ± 8.39e−08−	2.33e−03 ± 2.63e−04−	**5.86e−13 ± 2.63e−13**
	500000	1.15e−01 ± 9.02e−00	8.88e−18 ± 3.07e−17	0 ± 0	0 ± 0	0 ± 0
f_7	60000	5.15e+01 ± 9.51e+00−	4.10e−03 ± 7.18e−03−	*6.21e−05 ± 7.70e−06−*	3.48e−01 ± 2.53e−02−	**0 ± 0**
	500000	5.10e+01 ± 9.66e+00	4.10e−03 ± 7.18e−03	0 ± 0	0 ± 0	0 ± 0
−		7	7	6	7	
+		0	0	1	0	
≈		0	0	0	0	

Note: "−", "+" and "≈" denote that the performance of each compared algorithm is worse than, better than, and similar with that of NFO, respectively "FEs" means the number of fitness evaluations

Generally, NFO can obtain more accurate results with faster convergence rate for these 10-D functions.

For more complicated 50-D functions, NFO obviously delivers excellent performance compared with other algorithms. Details can be referred from Table 5. NFO can find the most accurate solutions on functions f_1, f_2, f_4, f_5, f_6 and f_7. In Table 5, results of Wilcoxon tests show that NFO is significantly better than PSO, SOMA, DE and CGO on 7, 7, 6 and 7 functions, respectively, while it is significantly worse than the compared algorithms on 0, 0, 1 and 0 functions. In Table 7, it is clear that NFO is more reliable than the studied algorithms, because NFO has the smallest FEs and has 100 % SR for all the solvable functions. The convergence graphs (on 50-D functions) of PSO, SOMA, DE, CGO and NFO are plotted in Fig. 7. The observation is the same with the numerical analysis. It is clear that NFO

Table 6 Successful rates and number of fitness evaluations in 10-D case (25 runs)

Algorithms	f_1	f_2	f_3	f_4	f_5	f_6	f_7
PSO							
SR	100	100	100	96	100	88	100
FEs	*8.50e+3*	**1.85e+5**	*1.425e+4*	2.92e+04	*1.76e+04*	*2.55e+04*	3.71e+04
SOMA							
SR	100	0	20	72	100	8	92
FEs	1.86e+4	–	2.90e+04	2.60e+4	3.63e+4	5.57e+4	6.36e+4
DE							
SR	100	100	92	100	100	96	100
FEs	8.55e+3	*1.89e+5*	**1.420e+4**	2.60e+4	1.76e+4	**2.50e+4**	*3.65e+4*
CGO							
SR	100	0	100	100	100	48	100
FEs	1.06e+4	–	2.94e+4	2.43e+4	2.33e+4	2.17e+5	5.94e+4
NFO							
SR	100	4	100	100	100	80	100
FEs	**6.23e+3**	2.08e+5	1.77e+4	**1.40e+4**	**1.33e+4**	6.70e+4	**3.05e+4**

Table 7 Successful rates and number of fitness evaluations in 50-D case (25 runs)

Algorithms	f_1	f_2	f_3	f_4	f_5	f_6	f_7
PSO							
SR	100	0	0	0	0	12	0
FEs	2.23e+5	–	–	–	–	*1.97e+5*	–
SOMA							
SR	100	0	4	12	100	100	72
FEs	*1.94e+5*	–	**3.17e+5**	**6.16e+5**	*3.46e+5*	2.56e+5	**5.96e+5**
DE							
SR	100	0	100	0	100	100	100
FEs	2.05e+5	–	*4.00e+5*	–	3.89e+5	2.85e+5	*9.48e+5*
CGO							
SR	100	0	40	0	100	100	0
FEs	3.09e+5	–	2.36e+6	–	5.93e+5	4.63e+5	1.59e+6
NFO							
SR	100	0	100	100	100	100	100
FEs	**1.40e+5**	–	7.87e+5	*6.46e+5*	**2.58e+5**	**1.94e+5**	*6.19e+5*

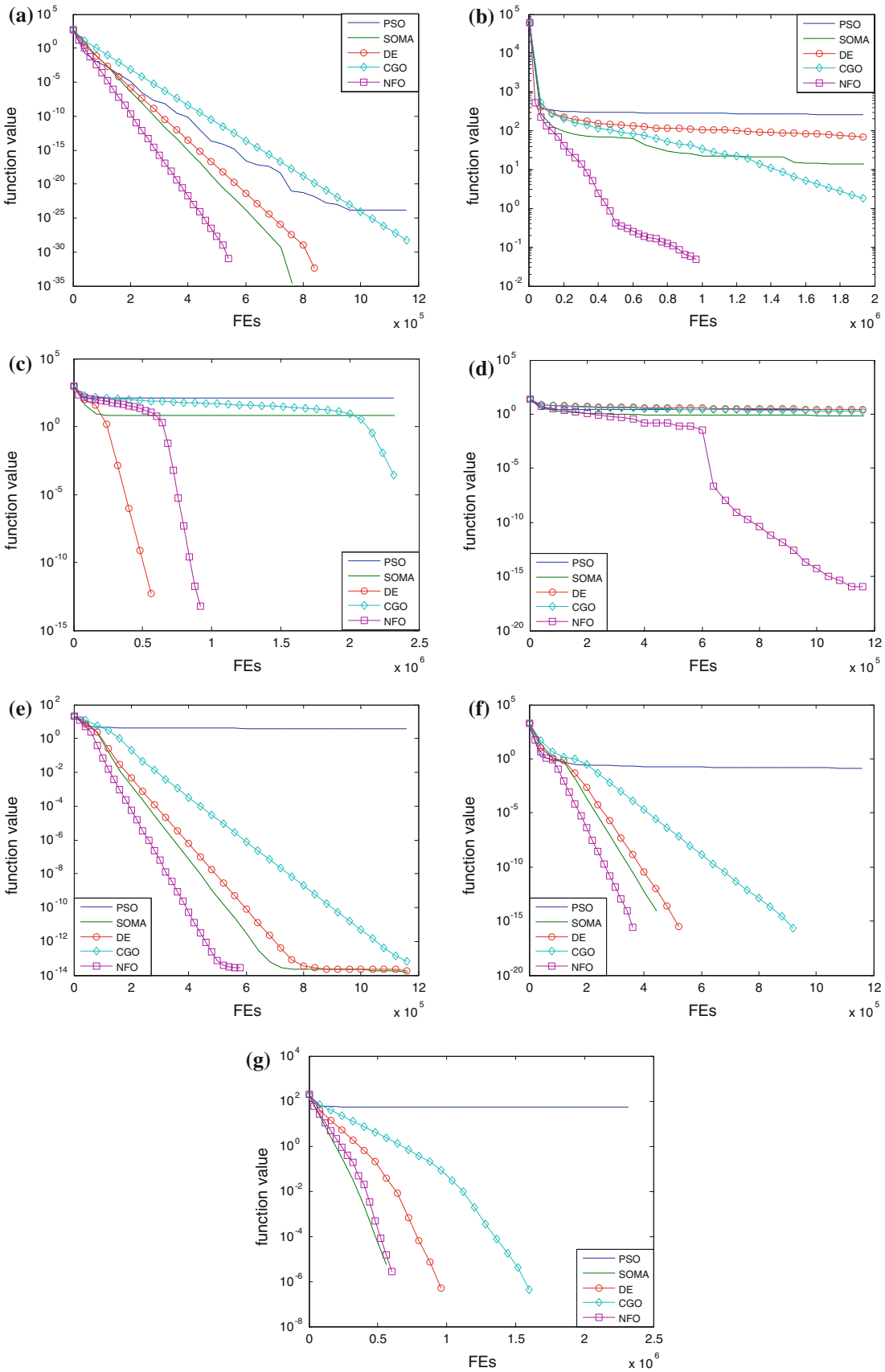
can solve most functions with the fastest convergence rate. In the subfigures (a), (b), (d), (e), (f) and (g) NFO delivers the fastest convergence rate on most un-rotated functions (except f_3).

In terms of accuracy, convergence rate and robustness, we can conclude that NFO can deliver excellent performance on finding the global minimum. As stated in (Wolpert and Macready 1997), we all realize that there is no single algorithm that can solve all different classes of problems with high performance. In this study, we find that NFO could not outperform all studied algorithms on all functions, but NFO could deliver excellent performance in most of our studied cases, especially in multimodal problems.

Fig. 7 The convergence graph profiles on 50-D functions (median function value). **a** Sphere function. **b** Rosenbrock's function. **c** Rastrigin's function. **d** Schaffer's function. **e** Ackley's function. **f** Griewank's function. **g** Stretched V sine wave function

4.3 Comparisons with some state-of-the-art algorithms

We also compare NFO with some state-of-the-art algorithms. They are comprehensive learning particle swarm optimization algorithms (CLPSO) (Liang et al. 2006),


(a)
(b)
(c)
(d)
(e)
(f)
(g)
function value
FEs
PSO
SOMA
DE
CGO
NFO

covariance matrix adaptation evolution strategy (CMA-ES) (Hansen and Ostermeier 2001; Auger and Hansen 2005), global and local search genetic algorithm (GL-25) (Garcia-Martinez et al. 2008) and J. Brest's adaptive DE (jDE) (Brest et al. 2006). In CLPSO, a particle uses its own historical best information and the historical best information of another random particle to update its velocity. The standard CMS-EA (Hansen and Ostermeier 2001) is used in the comparison, in which offspring solutions are sampled according to a multivariate normal distribution represented by a covariance matrix. GL-25 is a hybrid real-coded genetic algorithm which combines the global and local search for the global optimization. In jDE, it is stated that better parameter values tend to generate individuals which are more likely to survive and thus these values should be adaptively propagated to offspring generations.

In this experiment, the parameter settings in CLPSO, CMA-ES, GL-25 and jDE are the same as the optimal settings in their original papers. Table 8 summarizes the mean values and standard derivations over 25 runs on 50-D generalized functions. Table 9 summarizes the mean values and standard derivations over 25 runs on 50-D CEC2005 functions. For each run, each algorithm is stopped when its fitness evaluation times meet the maximum fitness evaluation number $5e^{+5}$. The results of Wilcoxon rank-sum tests between NFO and the four methods are also given in the table.

For the generalized functions $f_1, \ldots, f_7$, NFO performs better than CLPSO, CMA-ES, GL-25 and jDE with most accurate results. In Table 8, results of Wilcoxon tests show that NFO is significantly better than CLPSO, CMA-ES, GL-25 and jDE on 7, 7, 7 and 3 functions, respectively, while it is not significantly worse than the compared algorithms. For the CEC2005 functions F_1–F_{14}, the analysis will be given in three folds according to features of functions.

1. Rotated unimodal functions F_1–F_5:

CMA-ES can solve three unimodal functions, F_1, F_2 and F_3, which have outperformed other algorithms. The possible reason is that the adaptive strategy leads to fast convergence speeds. It is obvious that NFO still has competitive performance, delivering significantly better results than CLPSO and GL-25 on four and two functions respectively. NFO is a little worse than jDE because adaptive strategy in jDE can enhance the robustness of an algorithm.

2. Basic multimodal functions F_6–F_{12}:

On these seven functions, CMA-ES is still the best performer with the most accurate results on five functions F_5, F_6, F_7, F_8 and F_{11}; jDE performs the best on functions F_9 and F_{12}; NFO performs the best on F_{10}; CLPSO performs the best on F_9. Compared with GL-25, NFO is significantly better on five functions and is significantly worse on two functions. Compared with CPLSO, NFO is significantly better on three functions, and is significantly worse on one function. Thus, NFO is better or at least comparable with these algorithms on basic multimodal functions.

3. Expanded multimodal functions F_{13}–F_{14}:

On the two expanded functions, it can be noticed that NFO shows better than CLPSO, CMA-ES, GL-25 and jDE. The statistical tests also show that NFO is significantly different from them.

In all, CMA-ES performs better than CLPSO, GL-25, jDE and NFO with most accurate results on these 14 CEC2005 functions. For the functions, NFO is slightly worse than jDE and CMA-ES. One possible reason is that adaptation of parameters can enhance the accuracy and robustness of algorithms, which has not been considered in NFO. For the expanded functions, NFO must be the best performer with more accurate results. In Table 9, the

Table 8 Comparison with state-of-the-art algorithms on 50-D generalized functions (25 runs)

	CLPSO mean ± std. dev.	CMA-ES mean ± std. dev.	GL-25 mean ± std. dev.	jDE mean ± std. dev.	NFO mean ± std. dev.
f_1	3.22e−19 ± 9.88e−20−	1.24e−27 ± 2.21e−28−	3.59e−26 ± 1.00e−25−	0 ± 0≈	0 ± 0
f_2	4.69e+01 ± 8.33e+00−	4.35e+01 ± 1.06e+00−	4.32e+01 ± 2.56e−05−	*4.32e+01 ± 1.51e−14*−	**5.93e−02 ± 6.39e−02**
f_3	3.98e−02 ± 1.99e−01−	6.83e+02 ± 1.49e+02−	5.96e+01 ± 1.25e+01−	0 ± 0≈	0 ± 0
f_4	*1.23e+00 ± 7.98e−01*−	3.57e+00 ± 6.32e−01−	2.35e+00 ± 4.13e−01−	1.67e+00 ± 8.90e−01−	**1.10e−10 ± 4.94e−10**
f_5	1.35e−08 ± 3.36e−09−	1.98e+01 ± 8.64e−02−	6.29e−13 ± 1.14e−12−	*6.68e−14 ± 1.18e−14*−	**2.46e−14 ± 2.50e−15**
f_6	1.97e−12 ± 3.93e−12−	7.89e−04 ± 2.82e−03−	5.12e−15 ± 4.64e-15−	0 ± 0≈	0 ± 0
f_7	1.07e−02 ± 1.72e−03−	1.48e+02 ± 1.64e+01−	8.68e+00 ± 1.85e+00−	0 ± 0≈	0 ± 0
−	7	7	7	3	
+	0	0	0	0	
≈	0	0	0	4	

Note: "−", "+" and "≈" denote that the performance of each compared algorithm is worse than, better than, and similar with that of NFO, respectively

Table 9 Comparison with state-to-the-art algorithms on 50-D CEC2005 functions (25 runs)

	CLPSO mean ± std. dev.	CMA-ES mean ± std. dev.	GL-25 mean ± std. dev.	jDE mean ± std. dev.	NFO mean ± std. dev.
F_1	*0 ± 0*≈	4.16e−25 ± 7.30e−26−	5.45e−24 ± 1.73e−23−	*0 ± 0*≈	*0 ± 0*
F_2	8.52e+03 ± 1.22e+03−	**6.42e−24 ± 2.03e−24+**	1.05e+03 ± 7.28e+02−	*1.23e−02 ± 1.30e−02*+	1.00e+03 ± 2.18e+03
F_3	4.73e+07 ± 5.65e+06−	**4.20e−20 ± 8.49e−21+**	5.63e+06 ± 2.27e+06+	*5.04e+05 ± 2.33e+05*+	3.64e+07 ± 9.18e+06
F_4	3.20e+04 ± 4.31e+03−	2.83e+06 ± 1.12e+07−	*8.21e+03 ± 2.40e+03*+	**3.24e+02 ± 3.10e+02+**	3.01e+04 ± 6.24e+03
F_5	9.36e+03 ± 6.92e+02−	**5.36e−01 ± 1.47e+00+**	5.66e+03 ± 4.71e+02+	*3.21e+03 ± 6.87e+02*+	8.90e+03 ± 1.18e+03
F_6	*1.29e+01 ± 1.80e+01*+	**1.20e+00 ± 1.86e+00+**	5.12e+01 ± 2.01e+01+	4.44e+01 ± 3.09e+01+	5.89e+01 ± 3.39e+01
F_7	3.39e−01 ± 6.27e−02−	**1.23e−03 ± 3.28e−03**+	6.92e+03 ± 0−	3.77e−03 ± 8.10e−03+	*1.37e−02 ± 5.80e−03*
F_8	2.11e+01 ± 5.72e−02≈	**2.08e+01 ± 7.68e−01**≈	2.11e+01 ± 3.93e−02≈	2.11e+01 ± 4.28e−02≈	*2.10e+01 ± 4.39e−02*
F_9	*0*±*0*+	7.82e+02 ± 1.37e+02−	5.95e+01 ± 1.22e+01−	*0 ± 0*+	1.26e+01 ± 2.50e+00
F_{10}	2.65e+02 ± 3.26e+01−	9.51e+01 ± 1.96e+01−	2.83e+02 ± 1.32e+02−	*9.20e+01 ± 1.68e+01*−	**9.05e+01 ± 2.67e+01**
F_{11}	*5.09e+01 ± 2.87e+00*+	**1.26e+01 ± 2.78e+00+**	6.74e+01 ± 9.30e+00−	5.44e+01 ± 2.79e+00−	5.61e+01 ± 1.72e+00
F_{12}	6.96e+04 ± 1.66e+04−	*3.66e+04 ± 2.83e+04*+	6.21e+04 ± 1.88e+04−	**1.19e+04 ± 1.03e+04+**	5.52e+04 ± 2.14e+04
F_{13}	4.10e+00 ± 2.71e−01−	6.55e+00 ± 1.36e+00−	1.12e+01 ± 8.47e+00−	*3.01e+00 ± 2.09e−01*−	**2.48e+00 ± 6.09e−01**
F_{14}	*2.24e+01 ± 2.43e−01*−	2.45e+001 ± 3.26e−01−	2.26e+01 ± 3.04e−01−	2.27e+01 ± 2.23e−01−	**2.23e+01 ± 2.17e−01**
−	10	6	9	4	
+	2	7	4	8	
≈	2	1	1	2	

Note: "−", "+" and "≈" denote that the performance of each compared algorithm is worse than, better than, and similar with that of NFO, respectively

results of Wilcoxon tests show that NFO is significantly better than CLPSO, CMA-ES, GL-25 and jDE on 10, 6, 9 and 4 functions, respectively, while these compared algorithms are significantly better than NFO on 2, 7, 4 and 8 functions. It can be noticed that NFO has obtained competitive performance with significant difference compared with the state-of-the-art algorithms, especially for more complex expanded problems.

5 Conclusion

A new stochastic algorithm, NFO, is developed to solve multimodal problems globally. The newly proposed algorithm fully employs the local cooperation behavior to generate a new type of search mechanism "learning from the neighbors". NFO has been tested on several demanding benchmark problems. The presented results show that NFO is able to surpass several popular algorithms under comprehensive evaluations with respect to accuracy, convergence rate and robustness. This is a significant improvement because only a few algorithms with local information can deliver global optimization efficiently for the multimodal problems.

NFO uses a totally different approach that models the principle "learning from the neighbors" instead of the popular principle "learning from the bests". The concept of "learn from the neighbors" models practical situations that superior neighbor is more realistic as a target for learning rather than the found best one in the global environment. This concept is proved to be useful and efficient for performing global optimization. In NFO, these neighborhood field are more or less on the descending direction of fitness function, which can accelerate the search process.

References

Auger A, Hansen N (2005) A restart CMA evolution strategy with increasing population size. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, pp 1769–1776

Barraquand J, Langlois B, Latombe JC (1992) Numerical potential field techniques for robot path planning. IEEE Trans Syst Man Cybern 22(2):224–241

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

Caponio A, Neri F, Tirronen V (2009) Super-fit control adaptation in memetic differential evolution frameworks. Soft Comput 13(8–9):811–831

Derrac J, García S, Molina D, Herrera F (2011) 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):3–18

Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the 6th International Symposium on Micro machine and Human Science, Nagoya, Japan, pp 39–43

Eberhart RC, Shi Y (2000) Comparing inertia weights and constriction factors in particle swarm optimization. Proceedings of the 2000 IEEE Congress on Evolutionary Computation 2000 (CEC2000). Newyork, pp 84–89

Eberhart R, Shi Y (2001) Particle swarm optimization: developments, applications and resources. In: Proceedings of the 2001 IEEE Congress on Evolutionary Computation 2001 (CEC2001), Seoul, Korea, pp 81–86

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

Glover F (1990) Tabu search—part II. ORSA J Comput 2:4–32

Godoy A, Von Zuben FJ (2009) A complex neighborhood based particle swarm optimization. In: Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC2009), Trondheim, Norway, pp. 720–727

Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning, 1st edn. Addison-Wesley Professional, Reading

Greiner R (1996) PALO: a probabilistic hill-climbing algorithm. Artif Intell 84(1–2):177–208

Hansen N, Ostermeier A (2001) Completely derandomized selfadaptation in evolution strategies. Evol Comput 9(2):159–195

Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks, WA, pp 1942–1948

Kennedy J, Mendes R (2002) Population structure and particle swarm performance. In: Proceedings of the 2002 IEEE Congress of Evolutionary Computation (CEC2002), vol 2 Oregon, pp 1671–1676

Kennedy J, Mendes R (2002) Population structure and particle swarm performance. In: Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC 2002). Hawaii, pp 1671–1676

Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. Int J Rob Res 5(1):90–98

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680

Lampinen J, Storn R (2004) Differential evolution. In: Onwubolu G, Babu BV (eds) New optimization techniques in engineering. Springer, Germany, pp. 123–166

Liang JJ, Qin AK, Suganthan PN, Baskar S (2006a) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 10(3):281–295

Salomon R (1996) Reevaluating genetic algorithm performance under coordinated rotation of benchmark functions. BioSystems 39:263–278

Shi Y, Eberhart RC (1998) A modified particle swarm optimizer. In: Proceedings of the 1998 IEEE Congress on Evolutionary Computation (CEC1998). Alaska, pp 69–73

Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous space. J Global Optim 11(4):341–359

Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the cec2005 special session on real parameter optimization. Technical report, Nanyang Technological University

Vesterstrom J, Thomsen (2004) A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC2004), vol 2. Hawaii, pp 1980–1987

Vose MD (1999) Simple genetic algorithm: foundation and theory. MIT Press, MI

Watts DJ, Strogatz SH (1998) Collective dynamics of 'small-world' networks. Nature 393(6684):440–442

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

Wu Z, Chow TWS (2012) A local multiobjective optimization algorithm using neighborhood field. Struct Multidiscip Optim 45(6):853–870

Xu L, Chow TWS (2010) Self-organizing potential field network: a new optimization algorithm. IEEE Trans Neural Netw 21(9):1482–1495

Zelinka I (2004) SOMA-self-organizing migrating algorithm. In: Onwubolu G, Babu BV (eds) New optimization techniques in engineering. Springer, Germany, pp 167–217

Zhong W, Liu J, Xue M, Jiao L (2004) A multiagent genetic algorithm for global numerical optimization. IEEE Trans Syst Man Cybern Part B: Cybern 34(2):1128–1141