Received December 26, 2018, accepted February 1, 2019, date of publication February 22, 2019, date of current version March 25, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2900925

Hunting Optimization: An New Framework for Single Objective Optimization Problems

ZHENCHONG ZHAO[1], XIAODAN WANG[1], CHONGMING WU[2], AND LEI LEI[1]
[1]Air and Missile Defense College, Air Force Engineering University, Xi'an 710051, China
[2]College of Business, Xijing University, Xi'an 710123, China

Corresponding author: Xiaodan Wang (wang_afeu@163.com)

This work was supported by the National Natural Science Foundation of China under Grant 61876189, Grant 61273275, Grant 61806219, Grant 61703426, and Grant 61503407.

ABSTRACT Swarm intelligence algorithms play vital roles in objective optimization problems. To solve diverse and increasingly complicated problems, a new algorithm is always desired. This paper proposes a new optimization algorithm named hunting optimization based on human hunting activities. The population has consisted of huntsmen and hunting dogs. Each of them represents a feasible solution. In the evolution process, each huntsman shrinks its hunting ground with an adaptive reduction factor to concentrate on searching the most promising area. Then, each huntsman uses its dogs to search its local hunting ground and updates its position to a more promising place by its own searching results as well as the results of other huntsmen. At the same time, to further balance the exploration and exploitation, huntsmen with the least prey will be eliminated and their dogs will be distributed to others. Congestion detection is also applied to avoid getting stuck at a local optimum. The experimental results on 12 benchmark functions and CEC2013 test suites compared with 12 state-of-the-art algorithms demonstrate the effectiveness of the proposed method.

INDEX TERMS Evolutionary computation, single objective optimization, swarm intelligence, hunting optimization.

I. INTRODUCTION

Optimization is one of the most interesting research fields. Among all of the optimization methods, the research on evolutionary and swarm intelligence algorithms is a hot topic in computer science since they have been proved to be effective methods for solving non-differentiable, NP-hard and difficult non-linear optimization problems, compared to the classical derivative-based techniques [1].

In the past decades, many population-based stochastic search methods have been proposed, such as genetic algorithm (GA) [2], differential evolution (DE) [3], particle swarm optimization (PSO) [4], ant colony optimization (ACO) [5], artificial fish (AF) [6], artificial bee colony (ABC) [7], simulated annealing (SA) [8], etc. GA includes three main evolution strategies, i.e., selection, crossover and mutation. The standard DE is a variation of GA. PSO, ACO, AF and ABC are inspired by the foraging behaviors of bird swarm, ant colony, fish swarm and bee colony, respectively, while SA simulates the annealing process of physical systems. In order to enhance the searching ability, many strategies have been proposed for different algorithms, which mainly include three categories:

(I) The commonest way for some parameter-sensitive methods is parameter adaptation/control, for example, linear/nonlinear dynamic adaptation PSO [9], [10], adaptive SA [11], PSO based on the fitness performance [12], etc. An ensemble sinusoidal approach was adopted by Awad *et al.* [13] to select the control parameters for the success-history based adaptive DE; Olivas *et al.* [14] presented a dynamic parameter adaptation methodology for ACO based on interval type-2 fuzzy systems. Gou *et al.* [15] allocated a competition coefficient and selected specific evolutionary method for each particle in PSO.

(II) Another popular way is adjusting the learning or evolution strategies, for example, coarse-grained parallel GA [16], dynamic neighborhood learning PSO (DNLPSO) [17], enhanced fitness-adaptive differential evolution algorithm (EFADE) [18], self-adaptive DE with discrete mutation parameters (DMPSADE) [19], Cellular neighborhood with Gaussian distribution ABC [20], etc. Dao *et al.* [2] designed restarting and local solution generating modules for

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

GA; Wu *et al.* [21] divided the original population into four subpopulations to select and apply the best mutation strategy. Lynn and Suganthan [22] divided the population into two subsets to focus solely on exploration and exploitation and used comprehensive learning strategy to generate exemplars for each. Sharma *et al.* [23] updated onlooker bees' positions based on beer forth phenomenon.

(III) Hybrid-based strategy also attracts researchers' attention since incorporating advantages of different algorithms can improve the entire performance, for example, self-adaptive DE with modified multi-trajectory search (SaDE) [24], DE/SA [25], ACO/GA [26], ACO/ABC [27], etc. Chen *et al.* [1] merged the differential mutation of DE into dynamic multi-swarm PSO for velocity updating. Lynn and Suganthan [28] used a self-adaptive scheme to identify the top algorithm from five PSO variants in each generation. Worasucheep [29] employed an efficient mutation operation of DE to enhance the convergence of ABC. Nguyen *et al.* [30] proposed a hybrid method between Bat algorithm and ABC with a communication strategy.

The balance between exploration and exploitation plays a critical role for good convergence behavior [15], [22]. Exploration stands for the global searching ability of locating the promising areas while exploitation stands for the local searching ability of converging to the near-optimal solutions as fast as possible. Many of the methods mentioned above tried to found appropriate balance based on some taking for granted strategies (e.g., a larger value of velocity of PSO in early iteration and a small value in late iteration [31]). However, they failed to measure and determine the proper balance rate exactly, which still remains to be a challenging work. Besides, it is reasonable to slice off the searched spaces which have poor fitness values, since they have low probability existing optimal solution. But most of the existing methods fixed the searching space along the whole searching procedure, which may waste computing resource and decrease the convergence speed.

According to the ''No Free Lunch'' theorem [32], there is no single optimization algorithm to solve every problem effectively and efficiently. Therefore, new optimization algorithm is always desired [28]. Inspired by the natural and social tendency of a self-organized swarm, Kulkarni *et al.* [33] proposed a new optimization method called cohort intelligence (CI). In CI, every candidate follows a certain other candidate using roulette selection and adopts the associated qualities by sampling a predefined number of qualities from the corresponding sampling intervals. The notable characteristic of CI is that it shrinks every candidate's searching space in each iteration. However, CI doesn't obtain proper balance between exploration and exploitation and its simple evolution approach is inefficient, which make it suffer from premature convergence and get stuck at local optimum easily. Hunting search (HuS) [34] is another new optimization algorithm derived by simulating the behaviors of group hunting of animals when searching for food, such as wolves and lions. HuS includs three main strategies: moving toward the leader, correcting positions by members' cooperation and reorganizing the hunting group. Buttar *et al.* [35] developed a novel methodology named ''Dog Group Wild Chase and Hunt Drive (DGWCHD) algorithm'' by the intelligent chasing and hunting behaviors adopted by the dogs to chase and hunt their prey in groups. DGWCHD has been used for solving some real-word problems such as TSP benchmark problems [36], [37], frequency assignment problems [38], etc.

In human's hunting activities, a huntsman with some hunting dogs, which can be treated as a group, tries to search and catch prey in a hunting ground. The extended case based on this single group activity is that many groups work together to catch the most valuable prey (or as many prey as possible), which is similar to the optimization process that tries to find a global optimum solution as determined by an objective function. Based on this assumption, this paper proposes a novel optimization framework named hunting optimization (HO). In HO, every huntsman has a number of hunting dogs and a particular visible distance which determines the scope of its hunting ground. During the searching process, each huntsman adjusts its hunting ground adaptively. All huntsmen compete and cooperate with each other to obtain better exploration and exploitation. To further increase the searching efficiency, population reorganization is also adopted. In general, HO has very simple two-layer structure and learning strategies.

What shall be pointed out is that: (1) HO and HuS are quite irrelevant both on conception and optimization strategies. The former one simulates the hunting activities of human while the later one focuses on the behaviors of animals hunting in a group. (2) Some searching strategies of HO are similar to CI. However, HO performs much better and is much easier for intuitive understanding. (3) Since HO has the similar structure and constraints to CI, PSO and DE, etc., therefore HO can also be applied in the domains in which PSO and DE has been successfully used, such as resource planning, production scheduling, image enhancement, classification, clustering, and some combinatorial optimization problems [12], [39]–[44], etc.

The rest of this paper is organized as follows: Section II briefly introduces some background. Some concepts and the proposed algorithm are introduced in section III. Experimental results and analysis based on 40 benchmark functions with different dimensions are given in section IV. Section V concludes the paper and proposes future works.

II. BACKGROUND

A. PROBLEM FORMULATION

Optimization problem may contain different kinds of constrains such as inequality, equality and boundary constrains. Without loss of generality, minimization problem only with boundary constrains is considered in this paper. A D-dimensional real-parameter minimization optimization problem with boundary constrains can be formulated as

$$\begin{aligned} &\min f(\mathbf{x}) \\ &s.t.\ \mathbf{x} \in S \end{aligned} \tag{1}$$

where $f(\mathbf{x})$ is the objective function, $\mathbf{x} = [x_1, x_2, \cdots x_D]$ is a feasible solution, $S = \{x_i | l_i \leq x_i \leq u_i, i = 1, 2, \cdots D\}$ is a non-empty finite set, l_i, u_i are lower and upper bound of the i^{th} dimension. The objective is to find a global optimal solution $\mathbf{x}^*$ so that

$$f(\mathbf{x}^*) \leq f(\mathbf{x}), \quad \forall \mathbf{x} \in S$$

B. BRIEF REVIEW OF RELATED WORK

To our best knowledge, CI is the first method that shrinks the searching space for each individual along with the searching process. It is inspired by the self-supervised learning behavior of the candidates in a cohort. A cohort refers to a group of candidates interacting and competing with one another to achieve a common goal. In a cohort, the behavior of each candidate is determined by its qualities. Every candidate tries to improve its own behavior by following a certain candidate's qualities [33]. This makes every candidate learns from one another and evolves the whole cohort. In CI, the i^{th} feasible solution ($i \in [1, 2, \cdots, N]$) is represented by the qualities $[x_1^i, x_2^i, \cdots x_D^i]$ of the i^{th} candidate $\mathbf{x}^i$ with certain behavior (i.e., fitness $f(\mathbf{x}^i)$). N is the number of candidates. In each iteration, the main procedure of CI contains three steps:

(1) Every candidate $\mathbf{x}^i$, $i \in [1, 2, \cdots, N]$ calculates its probability of being selected by

$$P_i = \frac{1/f(\mathbf{x}^i)}{\sum_{j=1}^{N} 1/f(\mathbf{x}^j)}, \quad i \in [1, 2, \cdots, N] \tag{2}$$

(2) Every candidate $\mathbf{x}^i$ selects a candidate $\mathbf{x}^s = [x_1^s, x_2^s, \cdots x_D^s]$ to follow using roulette wheel approach and shrinks the sampling interval ψ_i to the local neighborhood of $\mathbf{x}^s$ by

$$\begin{aligned} \psi_i = \{\psi_j^i | x_j^s - (\left\| \psi_j^i \right\| * r/2) \\ \leq \eta_j^i \leq x_j^s + (\left\| \psi_j^i \right\| * r/2), j \in 1, 2, \cdots, D\} \end{aligned} \tag{3}$$

where $r \in [0, 1]$ is interval reduction factor and in the first iteration ψ_i is the original searching range.

(3) Every candidate $\mathbf{x}^i$ samples Q sets of qualities from the updated interval. The one with the best fitness among the Q sets is used to update $\mathbf{x}^i$.

The cohort is saturated if there is no significant improvement and difference among all candidates for successive considerable number of learning iterations, then the sampling interval of every candidate will be expanded to the original one. The searching process will stop if the cohort saturates to the same behavior for a predefined times.

A noticeable characteristic of CI is that it shrinks the searching space for every candidate in each iteration. This strategy leads candidates to concentrate on searching the most promising areas, which can increase the convergence speed in return. However, fast convergence speed may also cause premature convergence to local optimum. For example, if one suboptimal candidate is far better than the other candidates in the current iteration, then it has higher probability being selected and followed by other candidates and leads the whole population to get stuck at local optimum. CI also doesn't achieve proper balance between exploration and exploitation since the candidate and sampling number are fixed along the entire searching process. In addition, the searching strategy of CI is inefficient because the best solution in the population is not preserved. Based on the "No Free Lunch" theorem, new algorithm is always desired for solving diverse and increasingly complicated optimization problems.

III. HUNTING OPTIMIZATION ALGORITHM

This section first introduces some concepts related to hunting, then presents the main procedure of HO and at last analyzes the computation complexity.

A. CONCEPTS OF HUNTING ACTIVITIES

A single hunting group mainly consists of a huntsman, a number of hunting dogs and a hunting ground which is determined by the visible distance. These concepts are presented in Fig.1. In details, each group has the following behaviors:

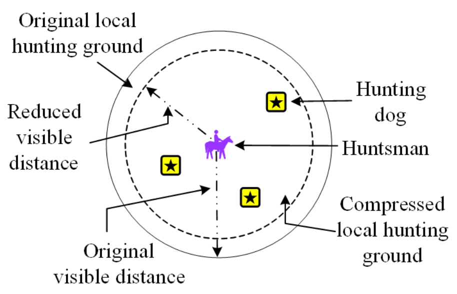
Original local hunting ground
Reduced visible distance
Original visible distance
Hunting dog
Huntsman
Compressed local hunting ground

FIGURE 1. Concepts of a hunting group with three hunting dogs.

(1) Searching: in each stage, every huntsman releases his dogs to randomly search its local hunting ground as in Fig. 1. If a new place with much more prey is found, huntsman will move to there and restart the searching process around the new position. At the same time, the huntsman will reduce its visible distance to shrink the hunting ground, since the dogs become more and more tired. When the dogs are exhausted, the searching space is too small and the group gets stuck at local optimum, then the tired dogs are replaced by new ones to enliven the searching.

(2) Following and assembling: besides searching the local hunting ground, every group always tries to exchange information with other groups by following and searching the most promising place found by the population. That is, each huntsman not only moves to its local optimum position but also moves a little step towards the place with the most valuable prey found by the whole groups. By the following the best behavior, the whole groups assemble gradually, this ensures that the population can converge to the same place in the end.

(3) Competing: along with the hunting process, huntsman who has the least prey has high probability in a wrong hunting direction. This group will be eliminated and its dogs will be distributed to other huntsmen. This can balance the

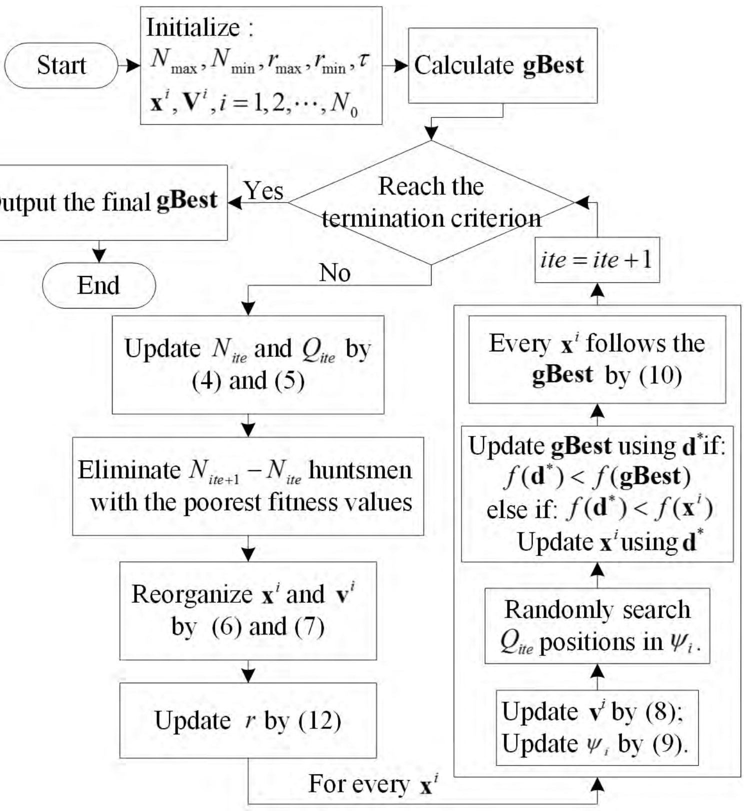
Start
Initialize : $N_{\max}, N_{\min}, r_{\max}, r_{\min}, \tau$ $\mathbf{x}^i, \mathbf{V}^i, i = 1, 2, \cdots, N_0$
Calculate **gBest**
Reach the termination criterion
Yes
No
Output the final **gBest**
End
$ite = ite + 1$
Update N_{ite} and Q_{ite} by (4) and (5)
Eliminate $N_{ite+1} - N_{ite}$ huntsmen with the poorest fitness values
Reorganize $\mathbf{x}^i$ and $\mathbf{v}^i$ by (6) and (7)
Update r by (12)
For every $\mathbf{x}^i$
Every $\mathbf{x}^i$ follows the **gBest** by (10)
Update **gBest** using $\mathbf{d}^*$ if: $f(\mathbf{d}^*) < f(\mathbf{gBest})$ else if: $f(\mathbf{d}^*) < f(\mathbf{x}^i)$ Update $\mathbf{x}^i$ using $\mathbf{d}^*$
Randomly search Q_{ite} positions in ψ_i.
Update $\mathbf{v}^i$ by (8); Update ψ_i by (9).

FIGURE 2. Framework of the HO algorithm.

exploration and exploitation. In the beginning, there are many huntsmen and each of them has few dogs, which results in good exploration. In the end, there are few huntsmen and each of them has lots of dogs, which results in good exploitation.

(4) Reorganizing: If the hunting grounds of two groups are too closed, they will search the same place, which is repetitive and inefficient. Therefore, the hunting ground of one of them will be reorganized. This ensures to fully utilize each group as much as possible.

By these behaviors, every group searches and cooperates with other groups to find the most valuable prey.

B. PROCEDURE OF THE PROPOSED ALGORITHM

In HO, each huntsman or dog is replaced by feasible solution. The hunting ground is replaced by searching space and the worth of prey is replaced by the fitness value of a particular objective function. The parameters and variables of HO are defined as follows

$N_{\max}, N_{\min}$	maximum and minimum numbers of huntsmen
$Q_{\max}, Q_{\min}$	maximum and minimum numbers of dogs
$r_{\max}, r_{\min}$	maximum and minimum reduction factor
ψ_0	initial searching space
ψ_i	searching space of $\mathbf{x}^i$
τ	congestion factor
$\mathbf{x}^i$	huntsmen/dogs
$\mathbf{d}^*$	local best dog
N_{ite}	number of huntsmen at the *ite* iteration
Q_{ite}	number of dogs at the *ite* iteration
$\mathbf{v}^i$	visible distance
v_r	elimination rate
gBest	huntsman with the global best fitness value

Fig. 2 presents the procedure of HO and the details of each step are given as follows (based on the problem defined in section II.A):

Step 1 (Initialization): initialize $N_{\max}, N_{\min}, Q_{\max}, Q_{\min}, r_{\max}, r_{\min} \in (0, 1)$, $\psi_0 = \{\psi_i | l_i \le \psi_i \le u_i, i = 1, 2, \cdots D\}$ and τ.

Initialize all huntsmen $\mathbf{x}^i = [x_1^i, x_2^i, \cdots x_D^i]$, $i = 1, 2, \cdots, N_0$ by $x_j^i = l_j + rand * (u_j - l_j), j = 1, 2, \cdots, D$, where *rand* is a uniformly distributed random number varies between 0 and 1, and let $N_0 = N_{\max}$. Initialize $\mathbf{v}^i = [v_1^i, v_2^i, \cdots v_D^i]$ for all $\mathbf{x}^i$ to be the original searching range, i.e., $v_j^i = u_j - l_j, j = 1, 2, \cdots, D$. After evaluating the fitness, the best huntsman **gBest** is recorded.

Step 2 (Elimination): To balance the exploration and exploitation, the number of huntsmen is decreased while the number of dogs of each huntsman is increased along with the searching process. Therefore, in each iteration huntsmen with the poorest fitness values will be eliminated and their dogs will be distributed to others which have good performance.

In HO, a shifted cosine function is used to control the elimination rate:

$$v_r = 0.5 * \cos(ite * \pi / maxIte) + 0.5 \tag{4}$$

Then adjust N_{ite} and Q_{ite} by

$$\begin{aligned} N_{ite} &= \max(v_r * N_{\max}, N_{\min}) \\ Q_{ite} &= \max((1 - v_r) * Q_{\max}, Q_{\min}) \end{aligned} \tag{5}$$

That is, in the *ite* iteration, a number of $N_{ite} - N_{ite-1}$ huntsmen are eliminated and the number of dogs for each huntsman increases from Q_{ite-1} to Q_{ite}.

The reason of adopting the cosine function is that it changes slowly at the beginning and ending compared to the middle part. Fig. 3 gives a sample with $N_{\max} = Q_{\max} = 100$, $N_{\min} = Q_{\min} = 2$. This property ensures that enough huntsmen for exploration at the beginning and enough dogs of each huntsman for exploitation at the ending.

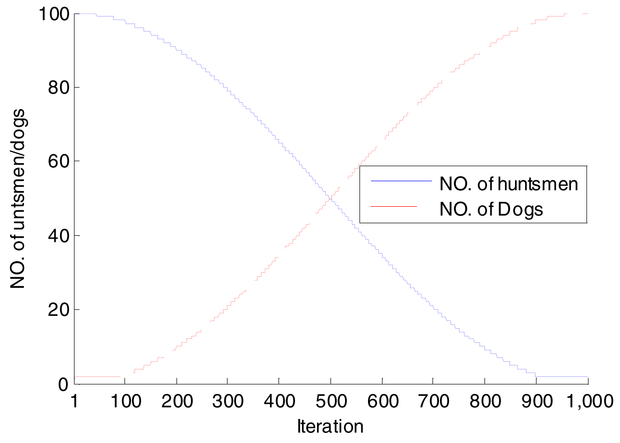

FIGURE 3. **Variations of numbers of huntsmen and dogs with $N_{\max} = Q_{\max} = 100$ $N_{\min} = Q_{\min} = 2$.**

Step 3 (Reorganization): If one huntsman $\mathbf{x}^i$ is too closed to one of other huntsmen or its visible distance is too small, it shall be reorganized. Since all groups always move to **gBest** dynamically while **gBest** only reduces its visible distance until a new best position is found (see Step 4-5), so only the distance between $\mathbf{x}^i$ and **gBest** is needed to checked. That is, if the distance between $\mathbf{x}^i$ and **gBest** is smaller than a predefined congestion factor τ, $\mathbf{x}^i$ will be re-positioned randomly and its visible distance is set to the original:

$$\begin{aligned} &if: \left\| \mathbf{x}^i - \mathbf{gBest} \right\| \leq \tau \\ &then: \begin{cases} x_j^i = l_j + rand * (u_j - l_j) \\ v_j^i = u_j - l_j, \end{cases} \quad j = 1, 2, \cdots, D \end{aligned} \tag{6}$$

In addition, if $\mathbf{v}^i$ is too small, $\mathbf{x}^i$ will get stuck at local area. In this situation the vision will also be reexpanded to the original vision multiplied by the current reduction factor:

$$\begin{aligned} &if: \sum_{j=1}^{D} v_j^i / D \leq \tau \\ &then: \ v_j^i = (u_j - l_j) \cdot r, \quad j = 1, 2, \cdots, D \end{aligned} \tag{7}$$

This reorganization step ensures that all huntsmen and dogs are used as much as possible.

Step 4 (Searching): Every remainder huntsman $\mathbf{x}^i$ reduces its visible distance by

$$\mathbf{v}^i = \mathbf{v}^i * r, \quad i = 1, 2, \cdots N_{ite} \tag{8}$$

Then every huntsman updates its searching space according to its vision $\mathbf{v}^i$ and current position $\mathbf{x}^i$:

$$\psi_i = \{\psi_i^j | (x_j^i - v_j^i/2) \leq \eta_i^j \leq (x_j^i + v_j^i/2), j \in 1, 2, \cdots, D\} \tag{9}$$

According to (9), the searching space of $\mathbf{x}^i$ is a hypercube centered on $\mathbf{x}^i$. Since $r < 1$, according to (8), the visible distance in each iteration become smaller. Therefore, the resulting searching space obtained by (9) become tight gradually. This enables every huntsman to focus on searching its local area.

At last $\mathbf{x}^i$ randomly scatters its Q_{ite} dogs in ψ_i to search Q_{ite} positions $\mathbf{d}^i$, $i = 1, 2, \cdots, Q_{ite}$ and finds the best position $\mathbf{d}^*$ with the minimum fitness value among them. If $f(\mathbf{d}^*) < f(\mathbf{gBest})$, **gBest** and $\mathbf{x}^i$ will be replaced by $\mathbf{d}^*$, and then directly go to Step 6. If $f(\mathbf{d}^*) > f(\mathbf{gBest})$ and $f(\mathbf{d}^*) < f(\mathbf{x}^i)$, the huntsman will move to the new place, i.e., $\mathbf{x}^i$ is replaced by $\mathbf{d}^*$. Otherwise, $\mathbf{x}^i$ remains unchanged.

Step 5 (Following): After the searching process above, every huntsman moves a little step towards to the **gBest** by

$$\mathbf{x}^i = \mathbf{x}^i + (\mathbf{gBest} - \mathbf{x}^i) * (1 - r) \tag{10}$$

In this way every huntsman can perceive and learn from the searching results of other huntsmen. This 'step' can be treated as the information sharing procedure and is crucial for convergence since it always guarantees the whole population move towards the best position found in the current iteration from every huntsmen's local optimum.

The searching and following process above can be summarized as

$$\mathbf{x}^i = \mathbf{x}^i + (\mathbf{d}^* - \mathbf{x}^i) + (\mathbf{gBest} - \mathbf{d}^*) * (1 - r) \tag{11}$$

According to (8) and (11), r is used to control the visible distance and step length. Similar to [10], to balance the exploration and exploitation, a nonlinear decreasing function is adopted for r, i.e.,

$$r_{ite} = r_{\max} - (r_{\max} - r_{\min}) * (ite/maxite)^2 \tag{12}$$

With regard to $\mathbf{v}^i$, a larger r will enhance exploring a promising direction while a smaller r will force the population concentrate on the local searching. With regard to the moving step in (11), groups with larger r lean to search everyone's local area while groups with smaller r will converge to the **gBest** fast. Therefore, (12) can not only balance the exploration and exploitation, but also balance the convergence speed and premature convergence.

Step 6 (Termination): If the stop criterion is satisfied, the algorithm will stop and the global best huntsman will be accepted, or else continue to Step 2.

```
Input: $N_{\max}, N_{\min}$, $Q_{\max}, Q_{\min}$, $r_{\max}, r_{\min} \in (0,1)$, $\tau$, $\psi_0$, $maxIte$
Initialize $\mathbf{x}^i, \mathbf{v}^i, \psi_i$, $i = 1,2,\cdots,N_{\max}$;
Calculate the fitness values of $\mathbf{x}^i$ and record the $\mathbf{gBest}$;
Set $ite = 0$
While $ite < maxIte$
  $ite = ite + 1$;
  Calculate $v_r$ according to (4);
  Calculate $r$ according to (12);
  Update $N_{ite}, Q_{ite}$ according to (5);
  Sort $\mathbf{x}^i$ and delete the worst $N_{ite} - N_{ite-1}$ huntsmen;
  For $i = 1 \to N_{ite}$
    If $\mathbf{x}^i$ is not $\mathbf{gBest}$ & $\|\mathbf{x}^i - \mathbf{gBest}\| \le \tau$
      reorganize $\mathbf{x}^i$ according to (6);
    End if
    If $\sum_{j=1}^{D} v_j^i / D \le \tau$
      Reexpanded $\mathbf{x}^i$ according to (7);
    else
      Update $\mathbf{v}^i = \mathbf{v}^i * r$;
    End if
    Update $\psi_i$ according to (9);
    Search $Q_{ite}$ positions $\mathbf{d}^i, i = 1,2,\cdots,Q_{ite}$ and finds the best position $\mathbf{d}^*$;
    If $f(\mathbf{d}^*) < f(\mathbf{gBest})$
      Replace $\mathbf{x}^i$ and $\mathbf{gBest}$ by $\mathbf{d}^*$;
    else if $f(\mathbf{d}^*) < f(\mathbf{x}^i)$
      Replace $\mathbf{x}^i$ by $\mathbf{d}^*$;
    End if
    Update $\mathbf{x}^i$ according to (10);
    Calculate the fitness value of $\mathbf{x}^i$;
    If $f(\mathbf{x}^i) < f(\mathbf{gBest})$
      Update $\mathbf{gBest}$ by $\mathbf{x}^i$;
    End if
  End for
End while
Output: $\mathbf{gBest}$
```

FIGURE 4. The pseudo code of HO.

The pseudo code of HO is given in Fig. 4.

In order to elaborate the searching procedure, a simple example with 3 huntsmen (represented by the riders) in 2-dimension space are presented in Fig. 5. Each huntsman has 3 dogs (represented by the stars). For simplicity, the visions for all huntsmen are assumed to be the same in each iteration and the reorganization step is not considered. The superscripts indicate the individual number and the subscripts indicate the iteration number. (a) Firstly, three huntsmen, i.e., $\mathbf{x}_0^1$, $\mathbf{x}_0^2$, $\mathbf{x}_0^3$ are randomly scattered to three positions with the original vision v_0 and the original searching space ψ_0^1, ψ_0^2, ψ_0^3. Let $\mathbf{x}_0^3$ be the global best (the huntsman in red). Before searching, each huntsman reduces its visions from v_0 to v_1 and the searching spaces are shrunk to $s\psi_0^1$, $s\psi_0^2$, $s\psi_0^3$ (indicated by the dashed circles in Fig. 5(a)). Then $\mathbf{x}_0^1$, $\mathbf{x}_0^2$, $\mathbf{x}_0^3$ scatter their dogs randomly to $s\psi_0^1$, $s\psi_0^2$, $s\psi_0^3$, respectively. The local best $\mathbf{d}_1^{*1}$, $\mathbf{d}_1^{*2}$ which are better than their huntsmen are marked in green and no dog obtains better fitness value than $\mathbf{x}_0^3$. (b) After the searching step, $\mathbf{x}_0^1$, $\mathbf{x}_0^2$ update their position according to (11) while $\mathbf{x}_0^3$ stays at the same position. These procedures are indicated by the purple lines in Fig. 5(b). The new vision and searching spaces for the new position $\mathbf{x}_1^1$, $\mathbf{x}_1^2$, $\mathbf{x}_1^3$ are v_1 and ψ_1^1, ψ_1^2, ψ_1^3, respectively. (c) Each huntsman reduces its visions from v_1 to v_2 and the searching spaces are shrunk to $s\psi_1^1$, $s\psi_1^2$, $s\psi_1^3$ (indicated by the dotted circles in Fig. 5(c)). After searching, the local best $\mathbf{d}_2^{*1}$, $\mathbf{d}_2^{*2}$, $\mathbf{d}_2^{*3}$ are obtained and $\mathbf{d}_2^{*3}$ become the global best (indicated by the red star in Fig. 5(c)). (d) $\mathbf{x}_1^1$ is directly replaced by $\mathbf{d}_2^{*1}$ and become the new global best huntsman, while $\mathbf{x}_1^2$, $\mathbf{x}_1^3$ update their position according to (11). These procedures are indicated by the purple lines in Fig. 5(d). These shrinking, searching and following steps continue iteratively until the stop criterion is satisfied.

C. RUNTIME COMPLEXITY ANALYSIS

Runtime complexity analysis is an important issue for population-based stochastic algorithms like PSO, DE. In many cases the average runtime of an optimization algorithm usually depends on its stopping criterion [17]. The most common method is reaching either certain number of iterations or function evaluations. In practice, only the fundamental operations of an algorithm are taken into account for calculating its runtime complexity. In HO, the number of huntsmen and dogs change dynamically. To analyze the complexity, we fix them as the maximum values $N_{\max}$, $Q_{\max}$. For each iteration, firstly all the huntsmen are sorted, which complexity can be treated as $O(N_{\max}^2)$[1]. Then for each huntsman, its D-dimensional searching space is shrunk followed by evaluating $Q_{\max}D$-dimensional fitness functions. Assume the algorithm is conducted for *mangen* iterations, the total runtime complexity in the worst case is $O((N_{\max}^2 + (D + Q_{\max} * D) * N_{\max}) * mangen)$. However, $N_{\max} \approx Q_{\max}$, and due to the variation of $N_{\max}$, $Q_{\max}$, in fact the computation complexity of HO is much smaller than $O(Q_{\max} * D * N_{\max} * mangen)$.

IV. EXPERIMENTS AND ANALYSES

In this section, the effectiveness of the proposed method is validated. Experiments mainly consist of three parts. First, the parameter settings of HO are analyzed on 12 classical functions to recommend the proper parameter values. Then, the evolution curves of HO and other 12 state-of-art methods on 8 of 12 benchmark functions are presented for intuitive grasp. Finally, all the methods are compared on the CEC 2013 benchmark functions [46] for further analysis.

A. BENCHMARK FUNCTIONS AND EXPERIMENT SETTINGS

Benchmark functions consist of two test suites. 12 classical functions as the first suite are described in Table 1, including 6 unimodal functions and 6 multimodal functions [1]. The former is used to test the convergence performance and the latter is used to test the global searching capability.

Another 28 shifted and rotated functions from CEC 2013 test suites are also adopted in this section. These 28 test functions can be divided into three classes:

(1) unimodal functions $F_1 - F_5$;

[1]This is obtained by many simple sorting algorithms.

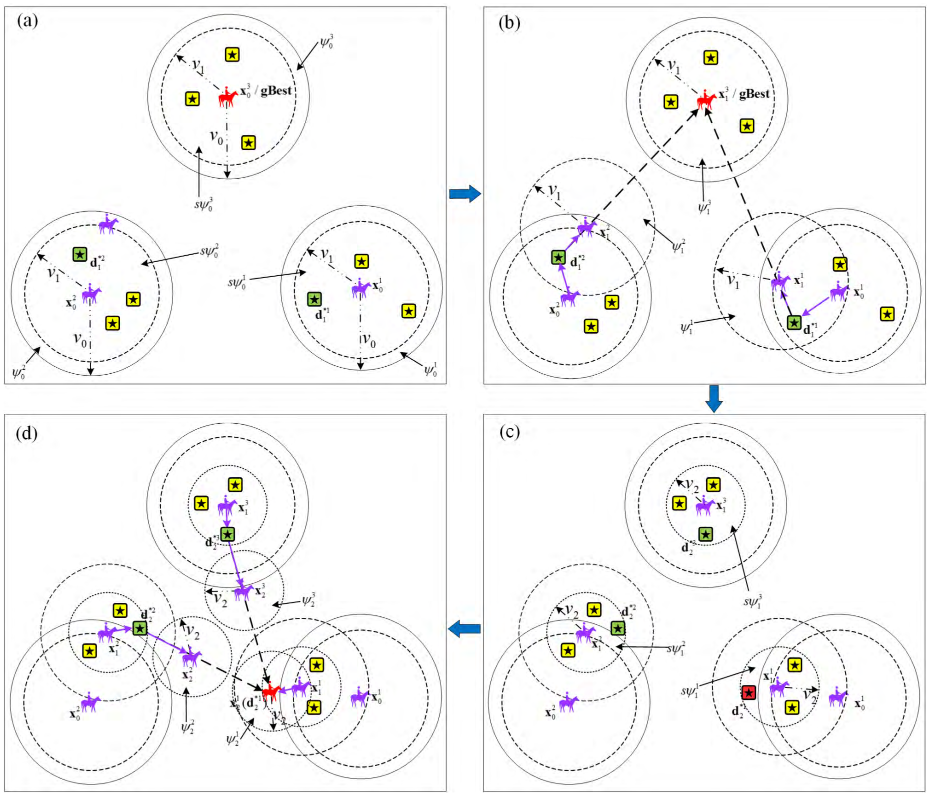
FIGURE 5. A simple example for the searching procedure in two iterations. The huntsmen are represented by the riders and the dogs are represented by the stars. The superscripts indicate the individual number and the subscripts indicate the iteration number. (a) and (c) Shrinking and searching. (b) and (d) Following the global best.

(2) multimodal functions $F_6 - F_{20}$;
(3) composition functions $F_{21} - F_{28}$;

A detailed description of these functions can be found in [46]. These benchmark functions in CEC 2013 can further verify the performance of HO in a more comprehensive manner.

Except for HO, another 12 state-of-art algorithms are used for comparison, including:

- Self-adaptive cohort intelligence (SACI) [39].
- CI with following median rule (CIM) [45].[2]
- Global version of PSO (GPSO) [9].
- Dynamic nonlinear inertia weight PSO (NWAPSO) [10].
- Dynamic neighborhood learning PSO (DNLPSO) [17].
- Ensemble PSO (EPSO) [28].
- Heterogeneous comprehensive learning PSO (HCLPSO) [22].
- Self-adaptive DE with discrete mutation control parameters (DMPSADE) [19].
- Enhanced fitness-adaptive DE with novel mutation (EFADE) [18].
- DE with multi-population based ensemble of mutation strategies (MPEDE) [21].
- Ensemble sinusoidal parameter adaptation with LSHADE (LSHADE) [13].
- Self-adaptive DE with modified multi-trajectory search (SADE) [24].

Most of the above algorithms are proposed in recent three years. The parameter configurations of involved algorithms are shown in Table 2. Similar to [15], the maximum number of function evaluations *MaxFEs* is set to $2000{*}D$. 50 independent runs are performed for each test function with 10 and 30 dimensions. The accuracy level of optimization results is set to be 10^{-8}. If the result is obtained within the fixed accuracy, it is defined as the run is successful. The algorithm terminates if it reaches *MaxFEs* or the error is less than 10^{-8}.

Three non-parametric statistical tests, i.e., Wilcoxon's rank sum test, Friedman's test, and Kruskal-Wallis test, with a significance level of 0.05 are employed to analyze the performance of all algorithms [19]. The experiments are conducted using Matlab R2014a and performed on a PC with an Intel Core i7-4790 @ 3.60GHz CPU and 4Gb RAM.

B. PARAMETRIC SENSITIVITY ANALYSIS

Three key parameter settings of HO, i.e., $N_{\max}, N_{\min}$, $Q_{\max}, Q_{\min}$, $r_{\max}, r_{\min}$ and τ, are analyzed by a non-parametric statistical test (i.e., Kruskal-Wallis test) on 12 10-dimensional classical functions. For simplicity, the

[2]In [45], seven variations of CI are discussed and CIM exhibited the best results.

TABLE 1. 6 test functions of the first test suite.

Type	Name	Search range	Optimal value
Unimodal	Sphere: $f_1(x)=\sum_{i=1}^{D}x_i^2$	$[-100,100]^D$	0
	Noise: $f_2(x)=\sum_{i=1}^{D}ix_i^4+random[0,1]$	$[-1.28,1.28]^D$	0
	Schwefel P2.21: $f_4(x)=MAX\{\lvert x_i\rvert,1\le i\le D\}$	$[-100,100]^D$	0
	Step: $f_4(x)=\sum_{i=1}^{D}(\lfloor x_i+0.5\rfloor)^2$	$[-100,100]^D$	0
	Schwefel P2.22: $f_5(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	$[-10,10]^D$	0
	Schwefel P1.2: $f_6(x)=\sum_{i=1}^{D}(\sum_{i=1}^{i}x_i)^2$	$[-100,100]^D$	0
Multimodal	Ackley: $f_7(x)=-20\exp(-0.2\sqrt{1/D\sum_{i=1}^{D}x_i^2})-\exp(1/D\sum_{i=1}^{D}\cos(2\pi x_i))+20+e$	$[-32,32]^D$	0
	Griewank: $f_8(x)=1/4000\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos(x_i/\sqrt{i})+1$	$[-600,600]^D$	0
	Penalized1: $f_9(x)=\frac{\pi}{30}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_D-1)^2\right\}+\sum_{i=1}^{D}u(x_i,10,100,4),$	$[-50,50]^D$	0
	Penalized2: $f_{10}(x)=0.1\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{D-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+(x_D-1)^2[1+\sin^2(2\pi x_D)]\right\}+\sum_{i=1}^{D}u(x_i,5,100,4)$	$[-50,50]^D$	0
	Schwefel: $f_{11}(x)=418.9829*D-\sum_{i=1}^{D}x_i*\sin(\sqrt{\lvert x_i\rvert})$	$[-500,500]^D$	0
	Rastrigin: $f_{12}(x)=\sum_{i=1}^{D}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	$[-5.12,5.12]^D$	0

TABLE 2. Parameters settings.

Algorithms	Parameters
SACI, CIM	$r=0.8,\ S=Q=10,\ \tau^{\max}=10$
GPSO	$w=0.9\sim0.4, c_1=c_2=2.0$
NWAPSO	$w=0.2\sim-0.3, n=1.2, c_1=c_2=1.49445$
DNLPSO	$w=0.9\sim0.4, c_1=c_2=1.49445$ $m=3, g=5, P_c=0\sim0.5, N=2$
HCLPSO	$w=0.99\sim0.2, c_1=2.5\sim0.5$ $c_2=0.5\sim2.5, c=3\sim1.5$
DMPSADE	$Setp=0.175, Msp=0.02, \sigma=0.3\sim0.8$
MPEDE	$\lambda_1=\lambda_2=\lambda_3=0.2, ng=20$ $\mu F=\mu CR=0.5, c=0.1$
LSHADE	$\mu F=\mu CR=0.5, H=5, G_{ls}=250, freq=0.5$
SADE	$p=0.5, F=0.5, CR=0.2/0.9, LP=50$

minimum and maximum values of huntsman and dog are set to be the same, i.e., $N_{\max}=Q_{\max}$, $N_{\min}=Q_{\min}$. The initial values are: $N_{\max}=10$, $N_{\min}=2$, $r_{\max}=0.95$, $r_{\min}=0.5$, $\tau=10^{-4}$. For each parameter setting, 20 independent runs are performed. In the figures, the blue line denotes the best performance on the corresponding parameter setting among all cases, and the red or black line indicates that the performances on the corresponding parameter setting are significantly worse than or similar to the best case (blue line) with 0.05 significance level, respectively.

1) SENSITIVITY TO HUNTSMAN AND DOG SIZE

In this section, the impact of population size on the performance of HO is investigated. Firstly, the minimum size is fixed as $N_{\min}=Q_{\min}=2$ while the maximum size ranges from 6 to 20 with step 2. The results of Kruskal-Wallis test is presented in Fig. 6.

The results show that no constant $N_{\max}$ ($Q_{\max}$) can provide the best performance for all test functions. Small values of $N_{\max}$ will result in poor exploration and large values of $N_{\max}$ will consume too much times of function evaluations in the beginning stage. Furthermore, the choice of $N_{\max}$ is also influenced by the scope of searching space and the complexity and dimension of objective function. According to Fig. 6, after sorting the ranks of different$N_{\max}$ on all 12 functions, $N_{\max}=18$ obtains the minimum average rank. In the following of this paper, $N_{\max}=Q_{\max}=18$ is chosen for experiments.

Secondly, the maximum size is fixed as $N_{\max}=Q_{\max}=18$ while the minimum size ranged from 1 to 10 with step 1.

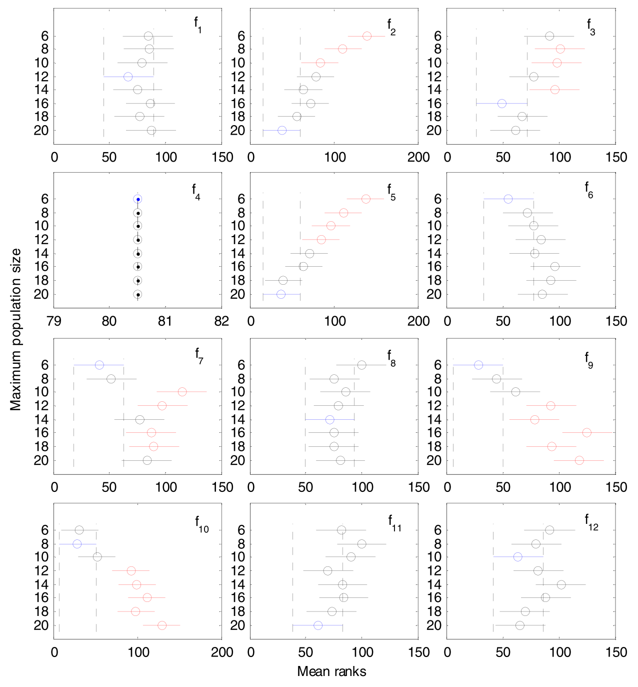

FIGURE 6. Interactive graphs of different maximum population sizes.

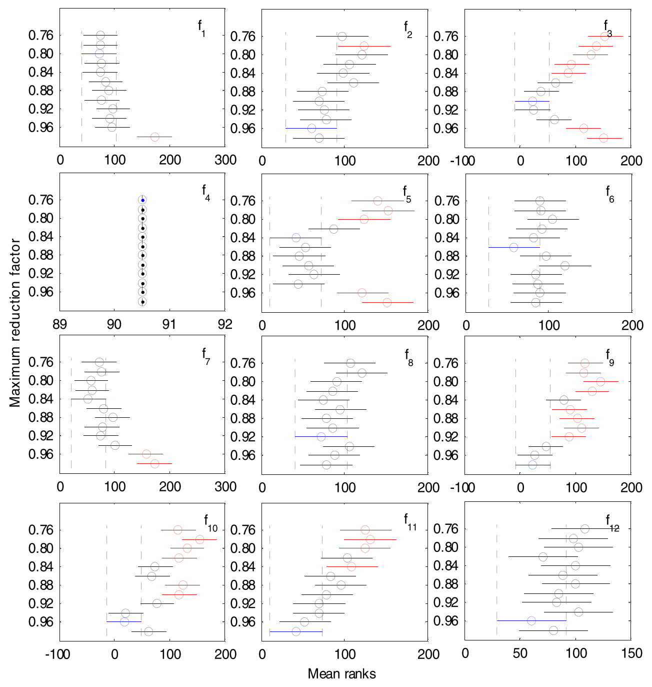

FIGURE 8. Interactive graphs of different maximum reduction factors.

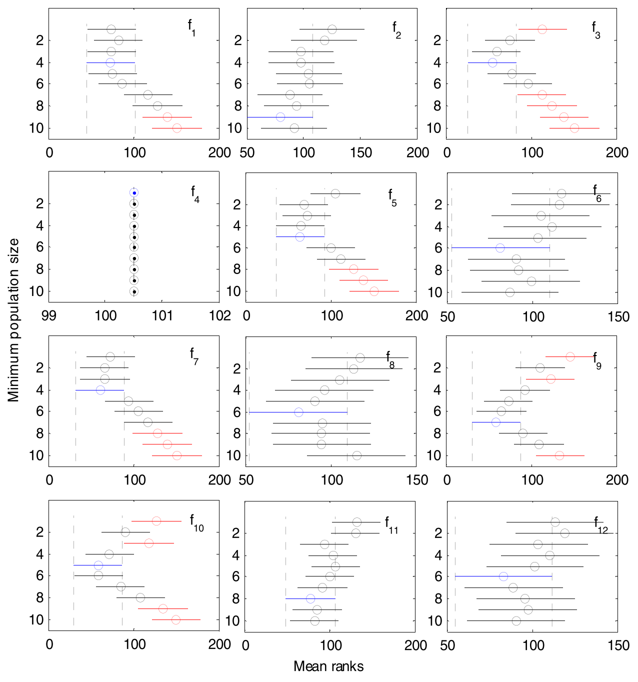

FIGURE 7. Interactive graphs of different minimum population sizes.

The results of Kruskal-Wallis test in Fig. 7 also show that no constant $N_{\min}$ ($Q_{\min}$) can provide the best performance in all test functions. However, $N_{\min} = 4, 5, 6$ always obtains the best or are similar to the best performance. Therefore, a mean value $N_{\min} = Q_{\min} = 5$ is chosen.

2) SENSITIVITY TO REDUCTION FACTOR

In this section, the impact of reduction factor on the performance of HO is investigated. Firstly, the minimum value is fixed as $r_{\min} = 0.5$ while the maximum value ranged from 0.76 to 0.98 with step 0.2. The results of Kruskal-Wallis test are presented in Fig. 8. Fig. 8 indicates that $r_{\max} = 0.92 \sim 0.96$ obtains the best or are similar to the best performance on most functions and $r_{\max} = 0.94$ is the only one that is not worse than any others on all functions. Therefore, the middle value $r_{\max} = 0.94$ is chosen.

Secondly, the maximum value is fixed as $r_{\max} = 0.94$ while the minimum size ranged from 0.36 to 0.76 with step 0.04. The results of Kruskal-Wallis test in Fig. 9 show that no constant $r_{\min}$ can provide the best performance in all test functions. $r_{\min} = 0.44 \sim 0.52$ obtains the best or are similar to the best performance on most functions. A smaller $r_{\min}$ can accelerate the convergence speed and a relative larger $r_{\min}$ can promote exploration. To balance the exploration and exploitation, a trade-off value, $r_{\min} = 0.48$, is chosen.

3) SENSITIVITY TO CONGESTION FACTOR

This section investigates the influence of congestion factor τ. τ is set to 1e-2, 5e-3, 1e-3, 5e-4, 1e-4, 5e-5, 1e-5, 5e-6, 1e-6, 5e-7 and 1e-7. The other parameters are fixed as discussed above. The results of Kruskal-Wallis test in Fig. 10 show that $\tau = 1e-6 \sim 1e-7$ obtains the best or are similar to the best performance on most functions. A larger τ will result in inadequate searching near the best individual. In contrast, a smaller τ will make the population get stuck at local optimum. Therefore, $\tau = 5 * 10^{-7}$ is chosen.

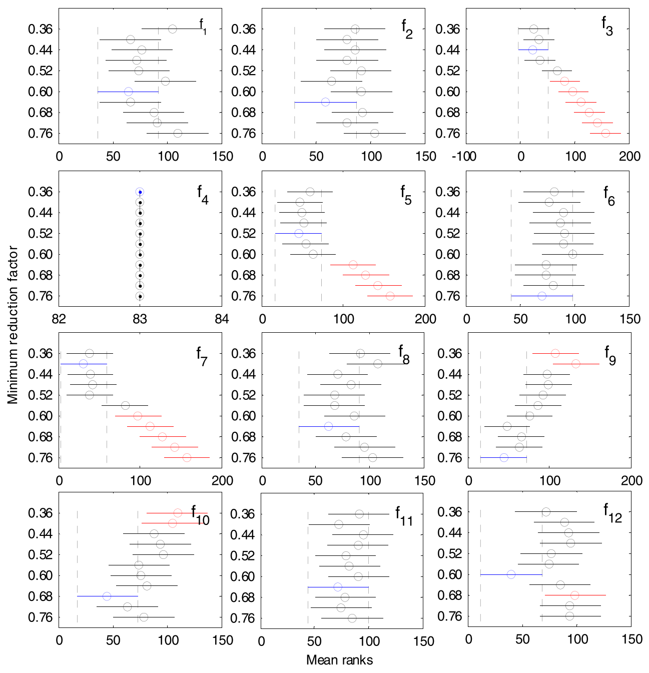

FIGURE 9. **Interactive graphs of different minimum reduction factors.**

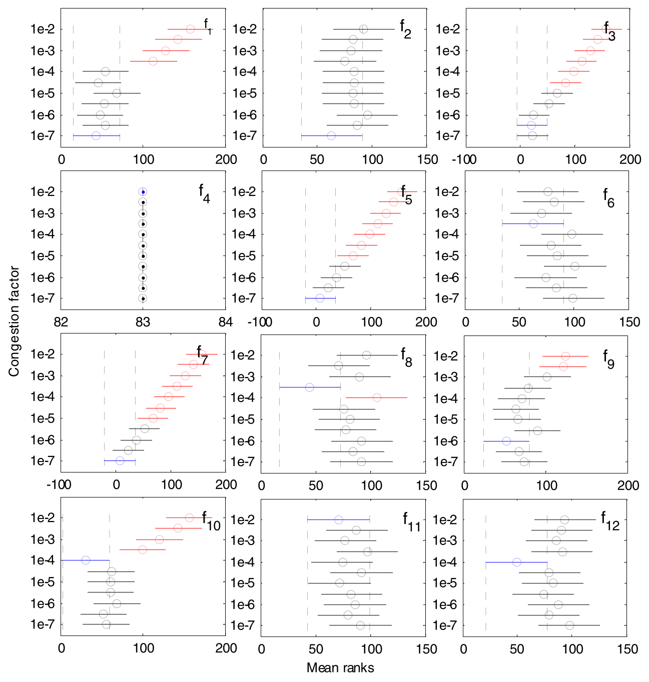

FIGURE 10. **Interactive graphs of different congestion factors.**

C. COMPARISON WITH OTHER METHODS

To further verify the effectiveness of the proposed HO, this section compares it with 12 state-of-art methods. The first comparison is conducted on 8 classical functions with regard to convergence speed for intuitive grasp, then addition comparisons on CEC 2013 benchmark functions are conducted with regard to accuracy and stability. Except for SACI and CIM, the population size for other comparison methods is 40. The recommended parameter settings in Section IV.B are adopted for HO.

1) RESULTS ON 6 CLASSICAL FUNCTIONS

In this section, the evolution processes of HO and other 12 methods on 8 10-dimensional functions ($f_1 - f_4, f_7 - f_{10}$) are presented to compare their convergence speed. The 8 classical functions are divided into two classes: unimodal and multimodal. In order to display the details of the convergence processes more clearly, all the outputs are in log. To avoid 0 output, all the outputs add a small value 10^{-9}. The first experiment results on 4 unimodal functions are presented in Fig. 11. The results on f_1 and f_4 show that, most methods can find the desired optimums but HO has a much faster speed than others except for DMPSADE and LSHADE. For f_2, f_3, most algorithms get stuck at local optimum, but HO also can obtain a faster speed at the same convergence level. In conclusion, HO has a much faster convergence speed than most other algorithms.

Additional experiments on 4 multimodal functions are also conducted and the results is presented in Fig. 12. On three functions f_8, f_9, f_{10}, HO obtains the desired values. Especially for f_8, HO wins other 10 algorithms except for DMPSADE, MPEDE and LSHADE. Only for f_7, HO loses to find the desired value. All these verify the searching ability of HO.

To further compare the calculating efficiency, the calculating time of all methods on f_2 ($D = 1$) with the maximum number of function evaluations ranging from 1000 to 20000 are recorded. The results in Fig. 13 show that HO is clearly faster than DNLPSO, EPSO, EFADE, LSHADE and SADE, and presents similar efficiency to SACI, CIM, HCLPSO, DMPSADE and MPEDE. Only GPSO and NWAPSO show better performance than HO. The experiments above validate the convergence speed and efficiency of HO.

2) RESULTS ON CEC 2013 FUCNTIONS

In this section, experiments on 28 CEC 2013 benchmark functions are conducted to further verify the performance of HO. The mean (denote as: *Mean*) and minimum (denote as: *Mini*) errors between the best solution in each run and the global optimum are employed as performance criteria. The experiments first consider the lower dimension ($D = 10$) followed by the higher dimensional experiment ($D = 30$) to further verify the robustness and scalability of the proposed algorithm. Wilcoxon's rank sum test is applied between 12 comparative methods and HO. The results are presented in Table 3∼6. The "Δ","∇" and "$\approx$" marks imply that the result of HO is significantly better than, worse than and similar to the compared result with 0.05 significance level, respectively.

The experiments when $D = 30$ present the similar results. From the results in Table 5, HO obtains nearly the best performance than other algorithms on much more 11 functions out of all 28 functions, including $F_1, F_3, F_6, F_{11}, F_{12}, F_{15}, F_{16}, F_{20}, F_{21}, F_{23}$ and F_{26}. However, the overall performance is deteriorated comparing to the

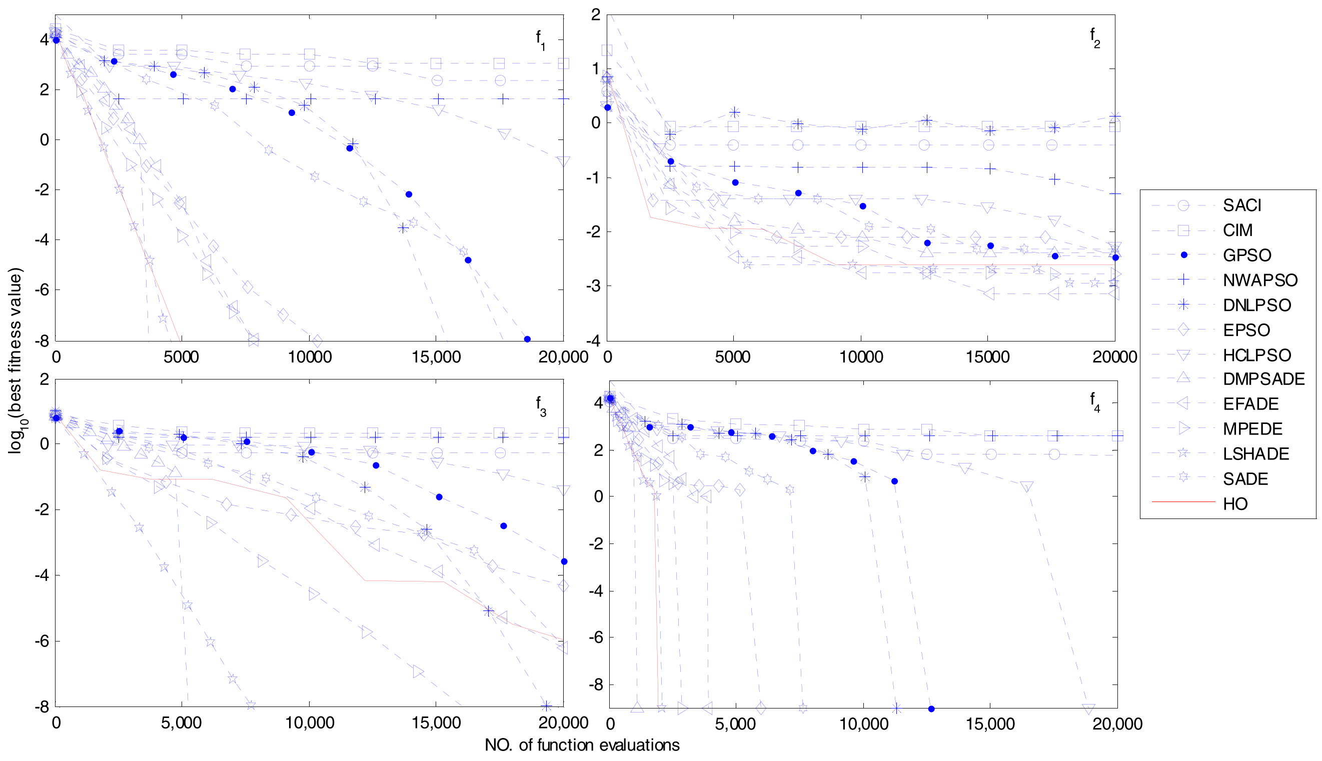

FIGURE 11. Convergence processes of different methods on unimodal functions.

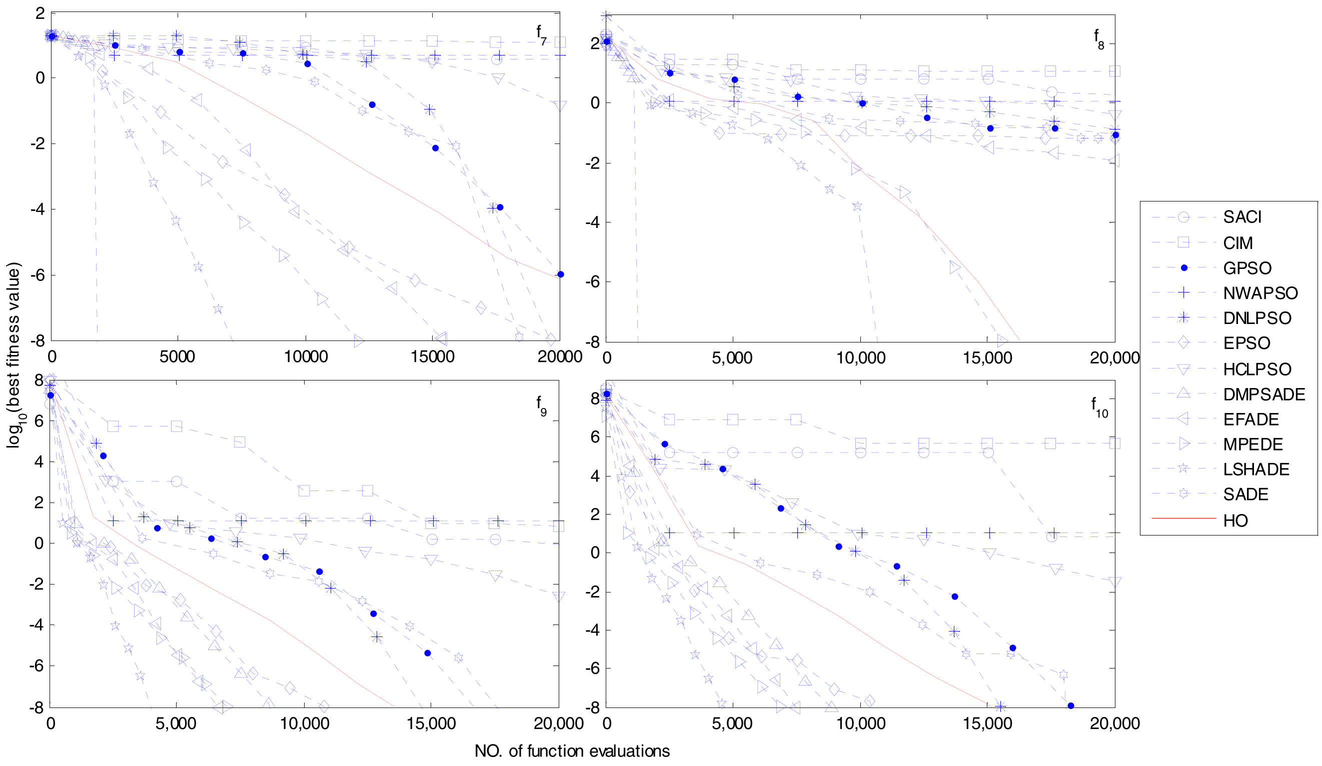

FIGURE 12. Convergence processes of different methods on multimodal functions.

results when $D = 10$, because of increased complexity. The summary results in Table 6 also show that HO is in the middle position among all 13 algorithms.

The results in Table 3 show that HO obtains nearly the best performance on $F_8, F_{12}, F_{15}, F_{16}, F_{18}$ compared to other algorithms. All of these are multimodal functions, indicating

TABLE 3. Results on the CEC2013 benchmark functions with $D = 10$.

Functions	F_1		F_2		F_3		F_4		F_5		F_6		F_7		F_8		F_9		F_{10}		F_{11}		F_{12}		F_{13}		F_{14}	
Criteria	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*
SACI	1.59e2 Δ	1.32e1	5.63e6 Δ	8.45e5	2.38e9 Δ	1.70e7	2.63e4 Δ	1.51e4	2.95e2 Δ	8.13e1	4.48e1 Δ	9.38e0	4.23e1 Δ	1.36e1	2.05e1 Δ	2.02e1	6.25e0 Δ	2.91e0	4.63e1 Δ	3.71e0	2.79e1 Δ	1.11e1	2.82e1 Δ	1.47e1	3.95e1 Δ	1.39e1	1.02e3 Δ	3.43e2
CIM	1.06e3 Δ	4.48e2	1.76e7 Δ	2.05e6	1.30e10 Δ	2.15e9	3.01e4 Δ	4.76e3	1.00e3 Δ	2.85e2	1.31e2 Δ	3.42e1	1.16e2 Δ	5.39e1	2.05e1 Δ	2.03e1	9.10e0 Δ	7.00e0	1.70e2 Δ	4.58e1	6.02e1 Δ	3.09e1	5.99e1 Δ	3.30e1	6.52e1 Δ	4.08e1	1.69e3 Δ	1.04e3
GPSO	**8.02e-9** ≈	**3.11e-9**	1.61e6 ≈	2.31e4	1.02e7 ∇	3.65e1	8.87e3 Δ	1.72e3	6.77e-8 ∇	**8.84e-9**	1.02e1 Δ	4.40e-3	7.84e0 ∇	2.53e-1	**2.04e1** ≈	2.02e1	3.70e0 ∇	1.13e0	4.70e-1 ≈	3.20e-2	5.52e0 Δ	1.03e0	1.73e1 Δ	6.96e0	2.80e1 Δ	4.97e0	3.01e2 ∇	4.34e1
NWAPSO	2.04e2 Δ	7.76e0	8.71e6 Δ	1.02e6	6.77e9 Δ	1.38e8	1.66e4 Δ	3.33e3	2.35e2 Δ	5.90e0	7.61e1 Δ	4.47e0	9.13e1 Δ	2.37e1	**2.04e1** ≈	2.02e1	7.00e0 Δ	3.15e0	9.86e1 Δ	4.30e0	3.54e1 Δ	5.18e0	4.76e1 Δ	7.17e0	6.10e1 Δ	2.54e1	5.95e2 ∇	1.33e2
DNLPSO	**7.32e-9** ≈	**3.46e-9**	1.81e6 Δ	2.17e5	2.53e6 ∇	4.32e-2	2.34e4 Δ	3.64e3	**8.15e-9** ∇	**3.61e-9**	8.01e0 Δ	2.61e0	1.35e1 ≈	1.39e-1	2.05e1 Δ	2.02e1	3.45e0 ∇	4.16e-2	2.52e-1 ∇	3.20e-2	5.51e0 Δ	9.97e-1	2.36e1 Δ	6.69e0	2.13e1 Δ	1.46e0	1.32e3 Δ	4.99e2
EPSO	**8.91e-9** ≈	**4.83e-9**	2.65e5 ∇	3.70e4	1.24e7 ≈	1.62e3	6.09e3 Δ	1.36e3	**9.11e-9** ∇	**6.72e-9**	6.76e0 Δ	1.16e-4	9.16e0 ∇	1.05e0	2.05e1 Δ	**2.01e1**	3.36e0 ∇	7.84e-1	6.84e-1 Δ	1.11e-1	3.56e0 Δ	2.65e-8	1.84e1 Δ	4.97e0	2.04e1 Δ	2.95e0	1.79e2 ∇	2.50e-1
HCLPSO	**8.04e-9** ≈	**3.71e-9**	6.99e5 ∇	2.85e4	1.46e6 ∇	1.15e3	5.32e3 Δ	1.39e3	6.24e-7 ∇	7.18e-8	5.15e0 ≈	1.22e-4	3.87e0 ∇	1.72e-1	2.05e1 Δ	2.03e1	3.33e0 ∇	5.90e-1	4.91e-1 ≈	7.14e-2	1.88e0 Δ	4.82e-7	1.26e1 Δ	2.99e0	1.96e1 ≈	4.29e0	1.30e2 ∇	3.74e0
DMPSADE	**7.99e-9** ≈	**4.39e-9**	1.51e5 ∇	1.21e4	6.42e3 ∇	1.88e0	5.69e3 Δ	1.98e3	**8.75e-9** ∇	**6.33e-9**	8.46e0 Δ	5.91e-2	**9.33e-1** ∇	6.18e-2	2.05e1 Δ	2.03e1	5.25e0 ≈	2.69e0	1.70e-1 ∇	2.35e-2	5.87e-3 ∇	3.53e-7	1.25e1 Δ	6.34e0	1.58e1 ≈	4.85e0	1.26e1 ∇	2.67e0
EFADE	**7.87e-9** ≈	**3.57e-9**	5.45e3 ∇	3.87e1	1.02e2 ∇	1.40e-3	3.58e2 ∇	2.62e0	**8.25e-9** ∇	**1.90e-9**	2.76e0 ∇	6.43e-6	1.56e0 ≈	9.97e-2	2.05e1 Δ	2.03e1	5.83e1 Δ	1.70e0	1.91e-1 ∇	7.40e-3	3.98e-2 ∇	**4.10e-9**	2.15e1 Δ	4.97e0	2.22e1 Δ	4.72e0	**9.75e-1** ∇	3.71e-2
MPEDE	**7.62e-9** ≈	**3.35e-9**	4.11e2 ∇	**5.03e-8**	**6.98e0** ∇	2.71e-6	**2.68e-1** ∇	**7.66e-9**	**8.68e-9** ∇	**4.90e-9**	6.08e0 Δ	**3.75e-9**	1.00e0 ∇	2.30e-3	**2.04e1** ≈	2.03e1	**2.16e0** ∇	**3.65e-2**	5.66e-2 ∇	**7.45e-9**	**7.88e-9** ∇	**4.40e-9**	6.92e0 ≈	1.99e0	1.09e1 ∇	1.99e0	2.83e0 ∇	2.20e-4
LSHADE	**8.43e-9** ≈	**4.50e-9**	1.23e3 ∇	2.61e-1	1.10e4 ∇	**8.18e-9**	2.40e2 ∇	**6.37e-9**	**8.87e-9** ∇	**5.99e-9**	7.85e0 Δ	**5.18e-9**	1.56e0 ≈	**1.20e-3**	**2.04e1** ≈	2.02e1	3.78e0 ≈	4.30e-2	**2.66e-2** ∇	**7.14e-9**	3.98e-2 ∇	**1.34e-9**	**4.80e0** ∇	1.99e0	**7.54e0** ∇	**1.04e0**	1.67e1 ∇	**5.28e-9**
SADE	**1.06e-9** ∇	**4.55e-13**	**8.24e0** ∇	3.89e-5	1.87e6 ∇	1.20e-4	8.75e-1 ∇	2.00e-3	6.13e-6 ∇	6.52e-7	**2.50e-9** ∇	**1.02e-12**	1.27e1 ∇	1.73e0	2.05e1 Δ	**2.01e1**	6.18e0 Δ	4.24e0	1.32e-1 ∇	4.43e-2	3.98e-2 ∇	**3.01e-12**	1.18e1 Δ	2.98e0	2.18e1 Δ	4.03e0	8.98e0 ∇	1.87e-1
HO	**8.10e-9**	**3.73e-9**	1.45e6	3.07e5	1.27e7	6.66e5	8.69e2	4.06e0	3.63e-3	**4.70e-9**	5.02e0	1.40e-3	1.47e1	6.78e0	**2.04e1**	**2.01e1**	4.82e0	1.15e0	4.50e-1	1.38e-1	4.43e-1	1.39e-2	6.75e0	**1.89e0**	1.63e1	2.42e0	8.69e2	3.83e2
Δ	3		4		3		8		3		9		3		8		5		4		7		10		8		3	
≈	8		1		1		0		0		1		3		4		2		2		0		1		2		0	
∇	1		7		8		4		9		2		6		0		5		6		5		1		2		9	

TABLE 3. *(Continued.)* Results on the CEC2013 benchmark functions with $D = 10$.

Functions	F_{15}		F_{16}		F_{17}		F_{18}		F_{19}		F_{20}		F_{21}		F_{22}		F_{23}		F_{24}		F_{25}		F_{26}		F_{27}		F_{28}	
Criteria	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*
SACI	1.08e3 Δ	5.06e2	6.82e-1 ≈	1.89e-1	4.81e1 Δ	1.89e1	5.03e1 Δ	1.60e1	2.44e0 Δ	7.47e-1	3.74e0 Δ	2.79e0	4.00e2 Δ	1.62e2	1.15e3 ∇	6.68e2	1.36e3 Δ	5.51e2	2.16e2 ≈	1.15e2	2.20e2 Δ	1.60e2	1.56e2 Δ	1.09e2	4.84e2 Δ	3.97e2	6.53e2 Δ	1.95e2
CIM	1.61e3 Δ	8.91e2	1.47e0 Δ	4.44e-1	8.59e1 Δ	4.39e1	8.99e1 Δ	5.71e1	1.46e1 Δ	2.17e0	4.11e0 Δ	3.39e0	4.90e2 Δ	4.38e2	2.08e3 Δ	1.31e3	2.05e3 Δ	1.54e3	2.15e2 ≈	1.41e2	2.28e2 Δ	1.61e2	2.28e2 Δ	1.36e2	6.17e2 Δ	4.39e2	8.69e2 Δ	6.88e2
GPSO	1.12e3 Δ	1.67e2	1.28e0 Δ	5.25e-1	2.03e1 ≈	5.17e0	4.33e1 Δ	1.60e1	1.01e0 ∇	4.60e-1	3.38e0 ≈	2.59e0	3.58e2 ∇	**1.00e2**	3.27e2 ∇	6.84e1	1.16e3 ≈	3.05e2	2.13e2 ≈	2.01e2	2.12e2 Δ	2.01e2	2.09e2 Δ	1.07e2	4.86e2 Δ	3.02e2	2.96e2 Δ	**1.00e2**
NWAPSO	1.03e3 Δ	4.34e2	9.32e-1 Δ	2.48e-1	4.36e1 Δ	1.70e1	5.48e1 Δ	2.70e1	4.61e1 Δ	1.52e0	3.86e0 Δ	2.98e0	4.10e2 Δ	2.75e2	9.72e2 ∇	3.83e2	1.48e3 Δ	7.81e2	2.21e2 Δ	1.37e2	2.21e2 Δ	1.20e2	2.29e2 Δ	1.21e2	5.57e2 Δ	3.72e2	7.58e2 Δ	1.84e2
DNLPSO	1.66e3 Δ	1.12e3	1.51e0 Δ	9.14e-1	3.70e1 Δ	1.56e1	4.27e1 Δ	2.06e1	1.72e0 Δ	4.99e-1	3.55e0 ≈	2.58e0	3.96e2 Δ	2.00e2	1.25e3 ∇	2.76e2	1.70e3 Δ	1.17e3	2.09e2 ≈	2.02e2	2.09e2 ≈	2.02e2	2.58e2 Δ	2.00e2	3.80e2 Δ	3.14e2	3.04e2 Δ	**1.00e2**
EPSO	1.11e3 Δ	4.73e2	1.45e0 Δ	7.78e-1	3.34e1 Δ	1.46e1	4.30e1 Δ	3.09e1	1.22e0 ≈	3.81e-1	3.07e0 ∇	1.88e0	3.93e2 ≈	2.31e2	2.38e2 ∇	1.78e1	8.52e2 ∇	**5.90e1**	1.82e2 ∇	**1.06e2**	2.07e2 ≈	1.62e2	1.31e2 ≈	1.03e2	3.57e2 Δ	3.07e2	2.97e2 Δ	**1.00e2**
HCLPSO	8.88e2 Δ	**6.81e1**	1.38e0 Δ	8.92e-1	1.98e1 ≈	1.28e1	3.73e1 Δ	2.07e1	8.02e-1 ∇	3.33e-1	2.87e0 ∇	1.84e0	**3.54e2** ≈	2.35e2	2.30e2 ∇	2.07e1	1.00e3 ∇	2.56e2	**1.53e2** ∇	1.07e2	1.97e2 ≈	1.19e2	1.24e2 ≈	1.03e2	3.46e2 ≈	**2.29e2**	**2.29e2** ∇	**1.00e2**
DMPSADE	9.83e2 Δ	5.38e2	1.28e0 Δ	6.53e-1	1.29e1 ∇	1.11e1	3.08e1 Δ	2.24e1	8.47e-1 ∇	5.29e-1	3.25e0 ≈	2.39e0	3.96e2 Δ	2.00e2	1.03e2 ∇	1.94e1	1.17e3 ≈	4.98e2	2.00e2 ∇	1.26e2	**1.96e2** ≈	**1.15e2**	**1.14e2** ∇	1.05e2	**3.05e2** ∇	3.00e2	2.92e2 ≈	**1.00e2**
EFADE	1.38e3 Δ	8.69e2	1.39e0 Δ	8.40e-1	1.21e1 ∇	1.10e1	4.50e1 Δ	3.46e1	9.11e-1 ∇	4.50e-1	3.34e0 ≈	2.51e0	3.92e2 ≈	2.00e2	1.66e2 ∇	6.15e1	1.53e3 Δ	8.67e2	2.04e2 ≈	1.83e2	2.03e2 ≈	1.83e2	1.43e2 ≈	1.08e2	3.09e2 ∇	3.00e2	2.64e2 ∇	**1.00e2**
MPEDE	9.80e2 Δ	3.58e2	1.46e0 Δ	8.99e-1	9.94e0 ∇	**1.63e-4**	3.40e1 Δ	1.85e1	5.38e-1 ∇	3.63e-1	2.90e0 ∇	1.85e0	3.92e2 ≈	2.00e2	6.72e1 ∇	5.02e0	9.44e2 ∇	8.49e1	2.06e2 ≈	2.00e2	2.13e2 ≈	1.78e2	1.54e2 Δ	**1.02e2**	3.44e2 ≈	3.00e2	3.08e2 Δ	3.00e2
LSHADE	**5.07e2** ∇	2.00e2	6.44e-1 ≈	2.33e-1	1.01e1 ∇	1.01e1	**1.77e1** ∇	**9.96e0**	**2.88e-1** ∇	1.65e-1	**2.69e0** ∇	**9.04e-1**	4.00e2 Δ	4.00e2	**3.87e1** ∇	**9.33e-9**	**6.01e2** ∇	9.46e1	2.07e2 ≈	2.00e2	2.03e2 ≈	1.96e2	1.50e2 Δ	**1.02e2**	3.27e2 ≈	3.00e2	3.02e2 Δ	3.00e2
SADE	6.70e2 ∇	1.77e2	1.07e0 Δ	3.80e-1	**9.08e0** ∇	5.08e-1	2.37e1 Δ	1.42e1	4.78e-1 ∇	**5.93e-2**	3.06e0 ∇	1.69e0	3.96e2 ≈	**1.00e2**	7.93e1 ∇	1.34e1	9.11e2 ∇	2.32e2	1.90e2 ∇	1.12e2	2.10e2 ≈	1.33e2	1.26e2 ≈	1.04e2	3.69e2 Δ	3.15e2	2.60e2 ∇	**1.00e2**
HO	6.94e2	3.17e2	**6.34e-1**	**1.65e-1**	2.14e1	1.01e1	2.01e1	1.41e1	1.22e0	6.19e-1	3.56e0	2.13e0	3.94e2	**1.00e2**	1.43e3	8.23e2	1.14e3	5.01e2	2.11e2	1.52e2	2.06e2	1.77e2	1.31e2	1.07e2	3.34e2	3.03e2	2.86e2	**1.00e2**
Δ	10		10		5		11		4		3		6		1		5		1		4		7		7		8	
≈	0		2		2		0		1		4		5		0		2		7		8		4		3		1	
∇	2		0		5		1		7		5		1		11		5		4		0		1		2		3	

TABLE 4. Summary of the results on the CEC2013 benchmark functions with $D = 10$.

Methods	SACI	CIM	GPSO	NWAPSO	DNLPSO	EPSO	HCLPSO	DMPSADE	EFADE	MPEDE	LSHADE	SADE
Δ	26	27	12	25	18	13	7	8	8	6	4	7
≈	2	1	8	1	5	6	9	7	7	7	8	3
∇	0	0	8	2	5	9	12	13	13	15	16	18

that HO has strong searching ability for multimodal problems. With regard to composed problems, HO also obtain more preferable performance than most of other algorithms on 5 (i.e., F_{21} and $F_{25} - F_{28}$) out of the 8 functions. Out of the total 28 functions, only on 5 functions (i.e., $F_3, F_5, F_{14}, F_{19}, F_{22}$) HO exhibits poorer performance than

TABLE 5. Results on the CEC2013 benchmark functions with $D = 30$.

Functions	F_1		F_2		F_3		F_4		F_5		F_6		F_7		F_8		F_9		F_{10}		F_{11}		F_{12}		F_{13}		F_{14}	
Criteria	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*
SACI	8.75e3 Δ	5.66e3	9.19e7 Δ	3.47e7	4.94e10 Δ	1.68e10	1.01e5 Δ	6.99e4	2.34e3 Δ	1.19e3	7.37e2 Δ	3.83e2	1.55e2 Δ	1.05e2	2.10e1 ≈	2.09e1	3.25e1 Δ	2.55e1	1.12e3 Δ	6.41e2	2.66e2 Δ	2.01e2	2.64e2 Δ	1.96e2	3.08e2 Δ	2.47e2	5.40e3 Δ	4.32e3
CIM	1.71e4 Δ	1.11e4	2.77e8 Δ	5.77e7	1.50e12 Δ	4.76e10	1.00e5 Δ	7.14e4	5.68e3 Δ	2.42e3	1.63e3 Δ	8.03e2	9.40e2 Δ	1.77e2	2.10e1 Δ	2.09e1	3.92e1 Δ	3.37e1	2.42e3 Δ	1.22e3	4.10e2 Δ	3.00e2	4.09e2 Δ	3.05e2	4.28e2 Δ	3.40e2	7.37e3 Δ	6.10e3
GPSO	8.85e-8 Δ	**7.61e-9**	2.10e7 Δ	4.99e6	6.80e8 Δ	2.51e7	2.48e4 Δ	1.09e4	2.33e4 Δ	1.22e-5	1.06e2 Δ	2.24e1	5.43e1 ∇	2.22e1	2.10e1 ≈	2.08e1	2.42e1 ∇	1.92e1	3.16e0 Δ	1.20e0	3.56e1 Δ	2.00e1	1.42e2 Δ	5.12e1	1.86e2 Δ	8.09e1	1.25e3 ≈	5.47e2
NWAPSO	2.21e4 Δ	7.04e3	1.70e8 Δ	4.32e7	4.70e13 Δ	2.14e10	7.13e4 Δ	2.16e4	5.81e3 Δ	1.53e3	2.06e3 Δ	6.91e2	1.38e3 Δ	1.15e2	2.10e1 ≈	2.09e1	3.32e1 Δ	2.47e1	2.82e3 Δ	1.28e3	4.24e2 Δ	2.61e2	4.28e2 Δ	2.45e2	5.19e2 Δ	2.84e2	4.36e3 Δ	3.09e3
DNLPSO	**8.81e-9** ≈	**2.49e-9**	1.23e7 Δ	3.35e6	7.87e8 Δ	6.12e6	4.85e4 Δ	1.16e4	**9.31e-9** ∇	**6.55e-9**	3.02e1 Δ	2.15e1	1.06e2 ∇	2.04e1	2.10e1 ≈	2.09e1	2.38e1 ∇	1.55e1	8.14e-2 ∇	9.90e-3	3.70e1 Δ	2.19e1	1.32e2 Δ	3.60e1	1.74e2 Δ	6.89e1	7.00e3 Δ	3.79e3
EPSO	**9.41e-9** ≈	**8.07e-9**	2.93e6 ∇	1.20e6	3.60e8 Δ	3.44e6	2.31e4 Δ	1.35e4	**9.74e-9** ∇	**8.79e-9**	3.53e1 Δ	2.17e0	4.39e1 ∇	1.44e1	2.10e1 ≈	2.09e1	2.22e1 ∇	1.40e1	1.70e-1 ∇	3.94e-2	2.20e1 Δ	6.97e0	1.40e2 Δ	4.40e1	1.87e2 Δ	9.58e1	1.14e3 ∇	3.26e2
HCLPSO	1.19e-5 Δ	9.39e-7	8.84e6 ∇	1.72e6	7.46e7 Δ	3.55e6	1.79e4 ≈	7.61e3	1.01e-3 ∇	3.93e-4	5.34e1 Δ	1.61e1	2.86e1 ∇	8.02e0	2.10e1 ≈	2.07e1	**2.13e1** ∇	**1.18e1**	2.20e0 ≈	1.26e0	1.73e1 Δ	6.20e0	6.97e1 Δ	2.40e1	1.37e2 ∇	7.06e1	1.43e3 ≈	3.74e2
DMPSADE	**9.16e-9** ≈	**7.01e-9**	1.93e6 ∇	5.51e5	1.26e7 ∇	**4.55e4**	2.96e4 Δ	1.17e4	**9.63e-9** ∇	**8.45e-9**	1.84e1 Δ	1.47e0	**1.53e1** ∇	6.40e0	2.10e1 ≈	2.09e1	3.00e1 ≈	2.16e1	1.28e-1 ∇	7.40e-3	2.78e-1 ∇	6.34e-7	5.90e1 Δ	2.37e1	1.02e2 ∇	7.23e1	9.01e1 ∇	5.98e1
EFADE	**8.78e-9** ≈	**4.76e-9**	5.76e5 ∇	1.24e5	**3.61e6** ∇	7.05e4	5.41e3 ∇	1.35e3	**9.19e-9** ∇	**7.56e-9**	2.37e1 Δ	1.38e1	2.15e1 ∇	**4.09e0**	2.10e1 ≈	2.09e1	3.02e1 ≈	1.44e1	**5.47e-2** ∇	7.40e-3	4.03e0 Δ	3.49e-2	3.67e1 Δ	**1.98e1**	9.13e1 ∇	**2.85e1**	8.47e2 ∇	5.43e2
MPEDE	**9.24e-9** ≈	**7.22e-9**	2.08e5 ∇	7.16e4	8.19e7 Δ	1.05e5	1.96e3 ∇	3.67e2	**9.52e-9** ∇	**8.26e-9**	1.80e1 Δ	1.22e-2	7.34e1 ∇	2.38e1	2.10e1 ≈	2.09e1	2.32e1 ∇	1.54e1	5.74e-2 ∇	**9.11e-9**	3.52e0 Δ	**9.05e-9**	5.41e1 Δ	2.58e1	1.20e2 ∇	5.95e1	1.31e2 ∇	7.14e0
LSHADE	**9.51e-9** ≈	**5.83e-9**	3.74e5 ∇	6.19e4	1.07e8 Δ	4.71e6	6.16e3 ∇	3.82e2	**9.80e-9** ∇	**9.04e-9**	2.79e1 Δ	7.21e0	3.50e1 ∇	1.26e1	2.10e1 ≈	**2.07e1**	2.79e1 ∇	1.37e1	1.03e-1 ∇	9.90e-3	3.52e0 Δ	**2.23e-9**	3.57e1 Δ	2.11e1	**8.66e1** ∇	3.94e1	6.21e1 ∇	**1.04e-1**
SADE	**7.47e-9** ≈	**3.98e-9**	**6.55e3** ∇	**1.49e2**	9.86e7 Δ	2.86e6	**2.86e0** ∇	**4.60e-2**	2.75e-7 ∇	2.42e-8	**1.28e0** ∇	**4.77e-9**	5.53e1 ∇	2.38e1	2.10e1 ≈	2.08e1	2.91e1 ≈	1.85e1	**5.47e-2** ∇	7.40e-3	**3.98e-2** ∇	**2.77e-9**	6.16e1 Δ	3.38e1	1.27e2 ∇	6.39e1	**5.36e1** ∇	3.54e0
HO	**8.60e-9**	**9.51e-9**	1.06e7	5.50e6	2.36e7	2.82e6	1.77e4	1.66e3	2.63e-1	5.83e-5	1.23e1	6.22e0	1.31e2	8.55e1	2.10e1	2.08e1	3.08e1	2.41e1	1.96e0	1.79e-2	1.76e0	2.52e-1	**2.76e1**	2.00e1	1.58e2	4.82e1	1.25e3	6.87e2
Δ	5		5		10		7		4		11		3		1		3		4		10		12		6		4	
≈	7		0		0		1		0		0		0		11		3		1		0		0		0		2	
∇	0		7		2		4		8		1		9		0		6		7		2		0		6		6	

TABLE 5. *(Continued.)* Results on the CEC2013 benchmark functions with $D = 30$.

Functions	F_{15}		F_{16}		F_{17}		F_{18}		F_{19}		F_{20}		F_{21}		F_{22}		F_{23}		F_{24}		F_{25}		F_{26}		F_{27}		F_{28}	
Criteria	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*	*Mean*	*Mini*
SACI	5.86e3 Δ	4.58e3	1.05e0 ≈	4.83e-1	4.90e2 Δ	4.09e2	4.97e2 Δ	4.04e2	3.00e3 Δ	3.77e2	1.50e1 Δ	1.45e1	1.54e3 Δ	1.04e3	6.53e3 Δ	5.36e3	6.94e3 Δ	5.46e3	3.00e2 Δ	2.78e2	3.28e2 Δ	3.12e2	2.62e2 Δ	2.03e2	1.17e3 Δ	1.04e3	2.93e3 Δ	1.91e3
CIM	7.20e3 Δ	6.34e3	2.24e0 Δ	1.27e0	7.76e2 Δ	6.70e2	7.89e2 Δ	6.50e2	2.09e4 Δ	1.56e3	1.50e1 Δ	1.50e1	2.47e3 Δ	2.03e3	8.13e3 Δ	6.95e3	8.20e3 Δ	6.32e3	3.22e2 Δ	3.00e2	3.52e2 Δ	3.37e2	3.51e2 Δ	2.07e2	1.38e3 Δ	1.29e3	3.78e3 Δ	3.07e3
GPSO	7.14e3 Δ	5.05e3	2.58e0 Δ	1.69e0	8.56e1 ∇	3.88e1	2.60e2 Δ	1.85e2	5.36e0 ∇	2.69e0	1.46e1 Δ	1.15e1	2.85e2 ≈	2.00e2	1.25e3 ∇	5.48e2	7.45e3 Δ	5.13e3	2.71e2 ≈	2.49e2	2.91e2 ≈	2.73e2	3.29e2 Δ	2.00e2	9.48e2 Δ	7.54e3	3.71e2 ∇	**1.00e2**
NWAPSO	5.11e3 Δ	3.58e3	1.83e0 Δ	7.90e-1	6.53e2 Δ	3.48e2	6.50e2 Δ	3.79e2	1.49e5 Δ	4.52e3	1.49e1 Δ	1.34e1	2.23e3 Δ	1.46e3	5.42e3 Δ	3.95e3	6.30e3 Δ	3.87e3	3.27e2 Δ	2.96e2	3.47e2 Δ	3.15e2	3.81e2 Δ	2.09e2	1.30e3 Δ	1.08e3	3.90e3 Δ	2.72e3
DNLPSO	7.92e3 Δ	6.99e3	2.96e0 Δ	2.27e0	1.83e2 ∇	8.07e1	2.31e2 Δ	1.93e2	6.96e0 ∇	1.98e0	1.48e1 Δ	1.25e1	3.19e2 Δ	2.00e2	7.27e3 Δ	5.68e3	8.04e3 Δ	7.21e3	2.59e2 ≈	2.21e2	**2.63e2** ∇	2.31e2	3.59e2 Δ	3.31e2	9.00e2 Δ	6.33e2	3.70e2 ∇	3.00e2
EPSO	6.63e3 Δ	5.56e3	2.84e0 Δ	1.78e0	2.03e2 ∇	9.90e1	2.56e2 Δ	2.13e2	1.26e1 ∇	3.89e0	1.24e1 ≈	1.13e1	3.22e2 Δ	**1.00e2**	9.67e2 ∇	3.67e2	6.56e3 Δ	3.62e3	2.49e2 ∇	2.30e2	2.82e2 ≈	2.44e2	2.09e2 Δ	2.00e2	8.01e2 ≈	5.89e2	4.03e2 ∇	**1.00e2**
HCLPSO	5.87e3 Δ	3.41e3	2.70e0 Δ	1.97e0	1.40e2 ∇	9.26e1	2.35e2 Δ	1.86e2	3.97e0 ∇	2.37e0	1.20e1 ≈	1.02e1	2.97e2 Δ	2.00e2	9.51e2 ∇	2.09e2	5.46e3 Δ	3.12e3	2.31e2 ∇	2.13e2	2.71e2 ∇	**2.14e2**	**2.00e2** ∇	2.00e2	**6.12e2** ∇	4.14e2	3.15e2 ∇	3.00e2
DMPSADE	5.27e3 Δ	4.33e3	2.11e0 Δ	1.26e0	4.56e1 ∇	3.64e1	1.45e2 ∇	1.19e2	4.05e0 ∇	1.43e0	1.50e1 Δ	1.50e1	3.51e2 Δ	2.00e2	8.79e2 ∇	1.67e2	5.74e3 Δ	4.04e3	**2.19e2** ∇	2.08e2	2.77e2 ∇	2.52e2	2.08e2 Δ	2.00e2	6.63e2 ∇	**3.96e2**	3.00e2 ∇	3.00e2
EFADE	7.26e3 Δ	6.25e3	2.90e0 Δ	2.19e0	6.61e1 ∇	5.82e1	2.28e2 ≈	1.95e2	6.52e0 ∇	4.95e0	1.27e1 Δ	1.16e1	3.08e2 Δ	2.00e2	1.62e3 ∇	8.98e2	7.50e3 Δ	6.72e3	2.26e2 ∇	**2.06e2**	2.91e2 ≈	2.56e2	**2.00e2** ∇	2.00e2	8.54e2 ≈	4.60e2	3.00e2 ∇	3.00e2
MPEDE	3.65e3 Δ	2.11e3	2.85e0 Δ	2.06e0	3.17e1 ∇	3.06e1	7.86e1 ∇	**4.55e1**	2.82e0 ∇	1.61e0	1.13e1 ≈	**9.79e0**	3.17e2 Δ	2.00e2	2.14e2 ∇	3.95e1	3.90e3 ∇	2.05e3	2.55e2 ∇	2.30e2	2.81e2 ∇	2.48e2	2.70e2 Δ	2.00e2	8.14e2 ≈	6.10e2	3.51e2 ∇	3.00e2
LSHADE	3.36e3 Δ	2.16e3	**1.02e0** ≈	**3.02e-1**	**3.07e1** ∇	**3.04e1**	**7.81e1** ∇	5.22e1	2.16e0 ∇	1.22e0	1.23e1 Δ	1.02e1	2.99e2 Δ	2.00e2	**1.37e2** ∇	**9.52e0**	**3.59e3** ∇	2.51e3	2.47e2 ∇	2.15e2	2.69e2 ∇	2.57e2	2.35e2 Δ	2.00e2	7.72e2 ∇	5.08e2	2.97e2 ∇	**1.00e2**
SADE	3.69e3 Δ	**1.65e3**	1.95e0 Δ	8.92e-1	3.20e1 ∇	3.11e1	8.25e1 ∇	5.61e1	**1.45e0** ∇	**4.99e-1**	1.18e1 ≈	1.03e1	**2.52e2** ∇	2.00e2	1.50e2 ∇	2.35e1	4.26e3 ≈	3.18e3	2.60e2 ≈	2.31e2	2.90e2 ≈	2.60e2	2.03e2 ≈	2.00e2	9.68e2 Δ	7.21e2	**2.60e2** ∇	**1.00e2**
HO	**2.50e3**	2.01e3	1.11e0	**3.02e-1**	3.54e2	1.77e2	2.00e2	1.06e2	4.40e1	1.64e1	**1.04e1**	1.03e1	2.78e2	**1.00e2**	4.22e3	2.13e3	4.20e3	**1.75e3**	2.66e2	2.31e2	3.00e2	2.71e2	2.05e2	2.00e2	8.21e2	7.01e2	1.07e3	**1.00e2**
Δ	12		10		3		7		3		8		10		4		9		3		3		9		6		3	
≈	0		2		0		1		0		4		1		0		1		3		4		1		3		0	
∇	0		0		9		4		9		0		1		8		2		6		5		2		3		9	

TABLE 6. Summary of the results on the CEC2013 benchmark functions with $D = 30$.

Methods	SACI	CIM	GPSO	NWAPSO	DNLPSO	EPSO	HCLPSO	DMPSADE	EFADE	MPEDE	LSHADE	SADE
Δ	26	28	17	27	17	12	10	9	8	8	8	5
≈	2	0	5	1	3	5	5	3	6	4	3	8
∇	0	0	6	0	8	11	13	16	14	16	17	15

most of other algorithms. On 7 functions HO obtains the best performance with regard to the minimum errors. However, MPEDE, LSHADE and SADE exhibit the best performance on more functions with regard to *Mean* and *Mini*, indicating that the hybrid and multi-population are proper methods to enhance the searching ability.

TABLE 7. Results of HO and the 5 DE variants on the 12 benchmark functions.

Functions	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}			
HO	**8.13e-9**	**1.52e-3**	**4.31e-7**	**0**	1.35e-6	7.60e1	**6.65e-7**	**7.65e-2**	5.06e-6	**5.96e-8**	1.13e3	1.03e1	Δ	≈	∇
DE/rand/1	4.84e-1 Δ	3.37e-2 Δ	2.91e0 Δ	5.50e-1 Δ	3.97e-1 Δ	4.86e0 ∇	3.69e0 Δ	7.10e-1 Δ	1.85e-1 Δ	1.48e-1 Δ	6.32e2 ∇	3.81e1 Δ	10	0	2
DE/rand/2	5.49e-1 Δ	2.99e-2 Δ	2.47e0 Δ	9.00e-1 Δ	4.33e-1 Δ	5.11e0 ∇	4.57e0 Δ	7.26e-1 Δ	1.32e-1 Δ	1.37e-1 Δ	**5.59e2** ∇	4.03e1 Δ	10	0	2
DE/best/1	**7.78e-9** ≈	8.31e-3 Δ	3.92e-5 Δ	**0** ≈	4.28e-8 ∇	5.00e2 Δ	3.86e0 Δ	1.32e-1 Δ	**7.86e-9** ∇	1.10e-3 ≈	8.10e2 ∇	1.31e1 ≈	5	4	3
DE/best/2	3.11e2 Δ	1.27e-1 Δ	2.61e1 Δ	3.51e2 Δ	1.19e1 Δ	2.87e3 Δ	1.18e1 Δ	4.53e0 Δ	3.49e2 Δ	4.83e4 Δ	1.41e3 Δ	6.28e1 Δ	12	0	0
DE/current-to-best/1	**7.74e-9** ≈	8.73e-3 Δ	2.07e-5 Δ	5.00e-2 ≈	**2.85e-8** ∇	**7.73e-9** ∇	5.78e-2 Δ	9.64e-2 ≈	3.11e-2 Δ	5.49e-4 Δ	6.25e2 ∇	**6.70e0** ∇	5	3	4

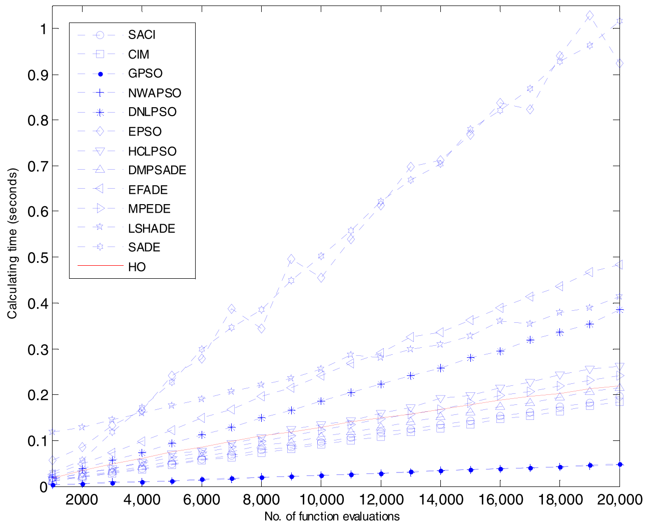

FIGURE 13. Calculating efficiency of different methods.

To present the difference of HO and other algorithms' performance more clearly, the results of Wilcoxon's rank sum tests are summarized in Table 4. The results indicate that HO shows apparent superiority than SACI, CIM, GPSO, NWAPSO,DNLPSO and EPSO. In contrast, HO fails to surpass the other 6 algorithms in most cases. This is because the searching behavior of HO is very simple while many complicated searching strategies, such as novel mutation strategy [18], multi-population and multiple strategies assembling [21], comprehensive learning [22], etc., are adotped in the last 6 methods. However, according to Fig. 11, HO is six times faster than SADE, three times faster than EFADE.

Since 5 of the last 6 methods which obtain better performance than HO are state-of-art DE variants, in order to further verify the effectiveness of the simple structure of HO, additional experiments comparing 5 simplest DE variants [19], [24], i.e., DE/rand/1, DE/rand/2, DE/best/1, DE/best/2, DE/current-to-best/1, are conducted and the results are presented in Table 7. The results show that HO is better than all of these five variants. Especially for DE/rand/1, DE/rand/2 and DE/best/2, HO wins on more than 10 of the 12 functions compared to each method. These verify the effectiveness of the simple searching strategies of HO.

In conclusion, HO exhibits preferable performance and high calculating efficiency. Due to its simple searching strategy, HO fails to surpass some of the state-of-art methods. However, as a new algorithm, the potential of HO will be unlocked if some complicated searching strategies are adopted.

V. CONCLUSION

In this study, according to the hunting behaviors of humans, a novel optimization method named HO is proposed. During the evolution process, searching space of each huntsman is shrunk adaptively, information can be exchanged by competition and cooperation between different huntsmen, congestion detection is also adopted. By making full use of these mechanisms, HO has god searching efficiency and optimization quality. Experimental results on 34 benchmark functions demonstrate the effectiveness of the proposed method.

The main contributions can be outlined as follows:

(I) A new optimization framework is proposed. Different from most of the existing methods that evolve the whole population by adjusting the searching direction of each individual, HO modifies the searching direction and shrinks searching space simultaneously. That is, HO reduces each huntsman's visible distance to shrink searching space in each iteration to concentrate on searching the most promising area gradually, which can increase the searching efficiency and obtain a faster convergence speed.

(II) In HO, exploration and exploitation are represented by huntsmen and dogs, respectively. Different from the traditional methods such as adjusting the learning steps to balance the exploration and exploitation, HO can obtain a straightforward balance by adjusting the rate between huntsmen and dogs explicitly.

(III) An adaptive scheme is applied in HO to adjust the number of huntsmen and dogs and the moving behavior of huntsmen. Restarting and congestion detection are applied to avoid being trapped in local optimum. These searching schemes adopted in HO can enhance the searching ability.

As a new optimization method, the performance of HO is limited by its simple searching strategies. However, the novel framework enables HO to adopt and assemble complicated searching strategies easily. Therefore, the further work will concentrate on applying some effective searching strategies in HO, such as multi-population, hybrid with other methods as well as some adaptive strategies.

REFERENCES

[1] Y. Chen *et al.*, "Dynamic multi-swarm differential learning particle swarm optimizer," *Swarm Evol. Comput.*, vol. 39, pp. 209–221, 2018.

[2] S. D. Dao, K. Abhary, and R. Marian, "An innovative framework for designing genetic algorithm structures," *Expert Syst. Appl.*, vol. 90, pp. 196–208, Dec. 2017.

[3] K. R. Opara and J. Arabas, "Differential Evolution: a survey of theoretical analyses," *Swarm Evol. Comput.*, vol. 44, pp. 546–558, Feb. 2019. doi: 10.1016/j.swevo.2018.06.010.

[4] K. R. Harrison, A. P. Engelbrecht, and B. M. Ombuki-Berman, "Self-adaptive particle swarm optimization: A review and analysis of convergence," *Swarm Intell.*, vol. 12, no. 3, pp. 187–226, Sep. 2018.

[5] Z. Chen, S. Zhou, and J. Luo, "A robust ant colony optimization for continuous functions," *Expert Syst. Appl.*, vol. 81, pp. 309–320, Sep. 2017.

[6] M. Neshat, G. Sepidnam, M. Sargolzaei, and A. N. Toosi, "Artificial fish swarm algorithm: A survey of the state-of-the-art, hybridization, combinatorial and indicative applications," *Artif. Intell. Rev.*, vol. 42, no. 4, pp. 965–997, 2014.

[7] W.-F. Gao, L.-L. Huang, S.-Y. Liu, and C. Dai, "Artificial bee colony algorithm based on information learning," *IEEE Trans. on*, vol. 45, no. 12, pp. 2827–2839, Dec. 2015.

[8] D. Bertsimas and O. Nohadani, "Robust optimization with simulated annealing," *J. Glob. Optim.*, vol. 48, no. 2, pp. 323–334, Oct. 2010.

[9] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proc. CEC*. Washington, DC, USA, Aug. 1999, pp. 1945–1950.

[10] A. Chatterjee and P. Siarry, "Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization," *Comput. Oper. Res.*, vol. 33, no. 3, pp. 859–871, Mar. 2006.

[11] G. Dudek, "Adaptive simulated annealing schedule to the unit commitment problem," *Electr. Power Syst. Res.*, vol. 80, no. 4, pp. 465–472, Apr. 2010.

[12] S. F. Li and C. Y. Cheng, "Particle swarm optimization with fitness adjustment parameters," *Comput. Ind. Eng.*, vol. 113, pp. 831–841, Nov. 2017.

[13] N. H. Awad, M. Z. Ali, P. N. Suganthan, and R. G. Reynolds, "An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC2014 benchmark problems," in *Proc. IEEE CEC*. Vancouver, Canada, Jul. 2016, pp. 2958–2965.

[14] F. Olivas *et al.*, "Ant colony optimization with dynamic parameter adaptation based on interval type-2 fuzzy logic systems," *Appl. Soft. Comput.*, vol. 53, pp. 74–87, Apr. 2017.

[15] J. Gou, Y. X. Lei, W. P. Guo, C. Wang, Y. Q. Cai, and W. Liu, "A novel improved particle swarm optimization algorithm based on individual difference evolution," *Appl. Soft. Comput.*, vol. 57, pp. 468–481, Aug. 2017.

[16] Z. Chen *et al.*, "A parallel genetic algorithm based feature selection and parameter optimization for support vector machine," *Sci. Program.*, vol. 2016, Aug. 2016, Art. no. 2739621.

[17] M. Nasir *et al.*, "A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization," *Inf. Sci.*, vol. 209, pp. 16–36, Nov. 2012.

[18] A. W. Mohamed and P. N. Suganthan, "Real-parameter unconstrained optimization based on enhanced fitness-adaptive differential evolution algorithm with novel mutation," *Soft Comput.*, vol. 22, no. 10, pp. 3215–3235, May 2018.

[19] Q. Fan and X. Yan, "Self-adaptive differential evolution algorithm with discrete mutation control parameters," *Expert Syst. Appl.*, vol. 42, no. 13, pp. 1551–1572, Feb. 2015.

[20] M. Zhang, N. Tain, V. Palade, Z. Ji, and Y. Wang, "Cellular artificial bee colony algorithm with Gaussian distribution," *Inf. Sci.*, vol. 462, no. 9, pp. 374–401, Sep. 2018.

[21] 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.

[22] N. Lynn and P. N. Suganthan, "Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation," *Swarm Evol. Comput.*, vol. 24, pp. 11–24, Oct. 2015.

[23] N. Sharma, H. Sharma, and A. Sharma, "Beer froth artificial bee colony algorithm for job-shop scheduling problem," *Appl. Soft. Comput.*, vol. 68, pp. 507–524, Jul. 2018.

[24] S. Z. Zhao, P. N. Suganthan, and S. Das, "Self-adaptive differential evolution with multi-trajectory search for large scale optimization," *Soft Comput.*, vol. 15, no. 11, pp. 2175–2185, Nov. 2011.

[25] H. Guo *et al.*, "Differential evolution improved with self-adaptive control parameters based on simulated annealing," *Swarm Evol. Comput.*, vol. 19, pp. 52–67, Dec. 2014.

[26] I. Ciornei and E. Kyriakides, "Hybrid ant colony-genetic algorithm (GAAPI) for global continuous optimization," *IEEE Trans. Syst., Man, Cybern., B, Cybern.*, vol. 42, no. 1, pp. 234–245, Feb. 2012.

[27] P. GaneshKumar, C. Rani, D. Devaraj, and T. A. A. Victoire, "Hybrid ant bee algorithm for fuzzy expert system based sample classification," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 11, no. 2, pp. 347–360, Mar. 2014.

[28] N. Lynn and P. N. Suganthan, "Ensemble particle swarm optimizer," *Appl. Soft Comput.*, vol. 55, pp. 533–548, Jun. 2017.

[29] C. Worasucheep, "An opposition-based hybrid artificial bee colony with differential evolution," in *Proc. CEC*. Sendai, Japan, May 2015, pp. 2611–2618.

[30] T. T. Nguyen *et al.*, "Hybrid bat algorithm with artificial bee colony," in *Advances in Intelligent Systems and Computing*, vol. 298, J. S. Pan, V. Snasel, E. Corchado, A. Abraham, and S. L. Wang (eds). Cham, Switzerland, Springer, 2014, pp. 45–55.

[31] G. Xu, "An adaptive parameter tuning of particle swarm optimization algorithm," *Appl. Math. Comput.*, vol. 219, no. 9, pp. 4560–4569, Jan. 2013.

[32] D. H. Wolper and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[33] A. J. Kulkarni, I. P. Durugkar, and M. Kumar, "Cohort intelligence: A self supervised learning behavior," in *Proc. SMC*. Manchester, England, Oct. 2013, pp. 1396–1400.

[34] R. Oftadeh, M. J. Mahjoob, and M. Shariatpanahi, "A novel metaheuristic optimization algorithm inspired by group hunting of animals: Hunting search," *Comput. Math. Appl.*, vol. 60, no. 7, pp. 2087–2098, Oct. 2010.

[35] A. S. Buttar, A. K. Goel, and S. Kumar, "Evolving novel algorithm based on intellectual behavior of Wild dog group as optimizer," in *Proc. IEEE Symp. Swarm Intell.*, Apr. 2010, pp. 73–78.

[36] A. S. Buttar, A. K. Goel, and S. Kumar, "Evolving novel algorithm based on intellectual behavior of Wild dog group as optimizer," in *Proc. SIS*, Orlando, FL, USA, Dec. 2014, pp. 1–7.

[37] A. S. Buttar, A. K. Goel, and S. Kumar, "Intellectual behavior of a group of wild animals: A computational intelligence study," *Int. J. Soft Comput. Eng.*, vol. 3, no. 1, pp. 127–132, Mar. 2013.

[38] A. S. Buttar, A. K. Goel, and S. Kumar, "Solving 55-cell benchmark frequency assignment problem by novel nature inspired algorithm," in *Proc. SPIN*, Feb. 2014, pp. 407–411.

[39] M. Aladeemy, S. Tutun, and M. T. Khasawneh, "A new hybrid approach for feature selection and support vector machine model selection based on self-adaptive cohort intelligence," *Expert Syst. Appl.*, vol. 88, pp. 118–131, Dec. 2017.

[40] A. Bhambhani and P. Shah, "PID parameter optimization using cohort intelligence technique for D.C motor control system," in *Proc. CACDOT*, Sep. 2016, pp. 465–468.

[41] G. Krishnasamy, A. J. Kulkarni, and R. Paramesran, "A hybrid approach for data clustering based on modified cohort intelligence and K-means," *Expert Syst. Appl.*, vol. 41, no. 13, pp. 6009–6016, Oct. 2014.

[42] A. J. Kulkarni and H. Shabir, "Solving 0–1 knapsack problem using cohort intelligence algorithm," *Int. J. Mach. Learn. Cybern.*, vol. 7, no. 3, pp. 1–15, Jun. 2016.

[43] A. J. Kulkarni, M. F. Baki, and B. A. Chaouch, "Application of the cohort-intelligence optimization method to three selected combinatorial optimization problems," *Eur. J. Oper. Res.*, vol. 250, no. 2, pp. 427–447, Apr. 2016.

[44] Y. Song, X. Wang, W. Quan, and W. Huang, "A new approach to construct similarity measure for intuitionistic fuzzy sets," *Soft Comput.*, vol. 23, no. 6, pp. 1985–1998, Mar. 2019.

[45] N. S. Patankar and A. J. Kulkarni, "Variations of cohort intelligence," *Soft Comput.*, vol. 22, no. 6, pp. 1731–1747, Mar. 2018.

[46] J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization," *Comput. Intell. Lab., Zhengzhou Univ.*, vol. 12, no. 34, pp. 281–295, Jan. 2013.

ZHENCHONG ZHAO was born in Zhoukou, Henan, China, in 1990. He is currently pursuing the Ph.D. degree with the Air and Missile Defense College, Air Force Engineering University. His research interests include pattern recognition, machine learning, and artificial intelligence.

XIAODAN WANG was born in Hanzhong, Shaanxi, China, in 1967. She received the Ph.D. degree in computer science from North Western Polytechnical University, China. She is currently a Professor and a Ph.D. Advisor with the Air and Missile Defense College, Air Force Engineering University. Her research interests include pattern recognition, machine learning, and artificial intelligence.

CHONGMING WU was born in Xi'an, Shaanxi, China, in 1966. He received the Ph.D. degree in computer science from Air Force Engineering University, in 2010. He is currently an Associate Professor with the College of Business, Xijing University. His research interests include artificial intelligence, investment, and financing decision.

LEI LEI was born in Mianyang, Sichuan, China, in 1988. She received the Ph.D. degree in computer science from Air Force Engineering University, in 2017, where she is currently a Lecturer with the Air and Missile Defense College. Her research interests include pattern recognition, machine learning, and artificial intelligence.