Complex & Intelligent Systems (2023) 9:6905–6925
https://doi.org/10.1007/s40747-023-01082-8

ORIGINAL ARTICLE

Heterogeneous differential evolution particle swarm optimization with local search

Anping Lin[1,3] · Dong Liu[2,3] · Zhongqi Li[4] · Hany M. Hasanien[5] · Yaoting Shi[1]

Received: 6 July 2022 / Accepted: 17 April 2023 / Published online: 11 May 2023

Abstract

To develop a high performance and widely applicable particle swarm optimization (PSO) algorithm, a heterogeneous differential evolution particle swarm optimization (HeDE-PSO) is proposed in this study. HeDE-PSO adopts two differential evolution (DE) mutants to construct different characteristics of learning exemplars for PSO, one DE mutant is for enhancing exploration and the other is for enhance exploitation. To further improve search accuracy in the late stage of optimization, the BFGS (Broyden–Fletcher–Goldfarb–Shanno) local search is employed. To assess the performance of HeDE-PSO, it is tested on the CEC2017 test suite and the industrial refrigeration system design problem. The test results are compared with seven recent PSO algorithms, JADE (adaptive differential evolution with optional external archive) and four meta-heuristics. The comparison results show that with two DE mutants to construct learning exemplars, HeDE-PSO can balance exploration and exploitation and obtains strong adaptability on different kinds of optimization problems. On 10-dimensional functions and 30-dimensional functions, HeDE-PSO is only outperformed by the most competitive PSO algorithm on seven and six functions, respectively. HeDE-PSO obtains the best performance on sixteen 10-dimensional functions and seventeen-30 dimensional functions. Moreover, HeDE-PSO outperforms other compared PSO algorithms on the industrial refrigeration system design problem.

Keywords Differential evolution · Industrial refrigeration system design · Local search · Particle swarm optimization

Introduction

Since the introduction of the genetic algorithm [1], computing intelligence has attracted wide spread attention and become a powerful tool to handle complex theoretical and real-life optimization problems. Nowadays, many population-based stochastic optimization algorithms are proposed every year. Compared with traditional optimization methods, population-based algorithms are easy to implement and can provide near optima on most problems. Furthermore, they are applicable to non-continuous and non-differentiable problems. When the mathematical model is absent, population-based algorithms are still applicable. Population based algorithms provide an effective approach to solve NP-hard problems.

As one of the widely used population-based algorithms, particle swarm optimization (PSO) [2] is simple in concept and converges fast. Due to the effectiveness of PSO on solving complex optimization problem, many theoretic and real-life application PSO algorithms have been reported since its inception. The original PSO suffers from diversity

✉ Dong Liu
liudong@xnu.edu.cn

Anping Lin
anping719@126.com

Hany M. Hasanien
hanyhasanien@ieee.org

[1] School of Physics and Electronic Electrical Engineering, Xiangnan University, Chenzhou 423000, People's Republic of China

[2] School of Computer and Artificial Intelligence, Xiangnan University, Chenzhou 423000, People's Republic of China

[3] Hunan Engineering Research Center of Advanced Embedded Computing and Intelligent Medical Systems, Chenzhou 423000, People's Republic of China

[4] College of Transportation Engineering, Hunan University of Technology, Zhuzhou 412007, Hunan Province, People's Republic of China

[5] Electrical Power and Machines Department, Faculty of Engineering, Ain Shams University, Cairo 11517, Egypt

loss and premature convergence. On complex multimodal problems, PSO is apt to fall into local optima. To mitigate premature convergence problem, a series of excellent PSO algorithms have been proposed in the past two decades. These improved versions of PSO can be roughly divided into four classes, namely, parameter adjustment algorithms, neighborhood topology algorithms, hybridization with other evolutionary algorithms (EAs) and learning strategy algorithms. The representative literature for each class is as follows:

1. Parameter adjustment: Adjusting the control parameters (inertia weight and acceleration coefficients) can control the algorithms' exploration and exploitation capability. Linear [3], non-linear [4], fuzzy system [5, 6], chaotic [7], and adaptive [8] adjustment algorithms have been proposed to balance exploration and exploitation. Parameter adjustment can improve performance to some extent, while on complex multimodal problems its impact is limited. Nowadays dynamic parameter adjustment methods are employed to enhance the performance of novel learning strategies [9].
2. Topology: Neighborhood topology determines the information exchanging among the particle swarm. Common topologies include ring, gbest, wheel, random, star and von Neumann [10]. Besides static topology, a series of dynamic topologies have been proposed. For example, Liang et al. [11] proposed dynamic multi-swarm topology, where the whole swarm is divided into many sub-swarms and these sub-swarms are regrouped periodically. Li et al. [12] introduced a dynamic pyramid topology based on the particles' fitness ranking. Lim et al. [13] developed an increasing topology connectivity to achieve better control of exploration/exploitation. Lin et al. [14] proposed a Tournament Topology Particle Swarm Optimization. The previous literature indicates that high topology connectivity converges quickly while low topology connectivity performs well on multimodal problems. Nowadays dynamic topology with fitness selection is the primary development trend.
3. Hybridization: PSO is widely hybridized with other evolutionary algorithms to obtain better performance. There are three major hybrid methods. (1) Employing different algorithm sub-swarms to perform different functions [15]; (2) Adopting another EA's operators to overcome premature convergence [16]; and (3) Utilizing another EA's operators to construct learning examples [17]. In recently years, many hybrid PSO algorithms have been proposed. For example, Abdulhameed et al. [18] proposed a hybrid algorithm based on PSO and the crow search algorithm (CSA) for feature selection. The CSA is adopted for enhancing global search. Yang [19] presented a hybrid algorithm based on PSO and cuckoo search (CS) for tuning the parameters of PID. CS with random walk was employed to enhance swarm diversity of PSO. Abdülkadir et al. proposed [20] a hybrid firefly and PSO algorithm with chaotic local search. Sama et al. [21] used PSO to explore the global search area and utilized fast-simulated annealing to refine the visited search area. Zhen et al. [22] developed a hybrid wolf pack algorithm and PSO for parameter estimation. To develop high performance hybrid PSO, the component algorithms should be excellent and complementary.
4. Learning strategy: Novel learning strategies can improve the performance of PSO by constructing promising learning exemplars [23] or introducing effective competition and cooperation mechanisms [24]. Some representative learning exemplar-based PSO algorithms are as follows: Gong et al. [17] proposed genetic learning particle swarm optimization by employing the crossover, mutation and selection operators of genetic algorithms to breeding exemplars. Cheng et al. [25] introduced a "learning from any better particles" mechanism and proposed social learning particle swarm optimization. Xu et al. [26] proposed a dimensional learning strategy to discover and employ the promising particles' information. Wang et al. [27] proposed an adaptive learning strategy to adjust the self-learning component and competitive-learning component. To deal with large-scale optimization, Kaucic et al. propose [28] a level-based learning swarm optimizer with a hybrid constraint-handling technique for large-scale portfolio selection problems. To obtain good balance between exploration and exploitation on large-scale problems, Li et al. [29] proposed a learning structure decoupling exploration and exploitation for large-scale optimization. Sheng et al. [30] employed dynamic p-learning mechanisms and multi-level population to improve the performance of PSO on large-scale optimization.

To help the particle swarm converge to optima rapidly, local search strategies are employed. For example, Liang et.al. [31] employed a quasi-Newton method to further improve a portion of top local best regularly. Wu et al. [32] conducted comparison experiments and pointed out that the Broyden–Fletcher–Goldfarb–Shanno method (BFGS) performs better than Nelder–Mead simplex search, Davidon–Fletcher–Powell (DFP) and pattern search (PS) on complex optimization problems. Chen et al. [33] used BFGS to enhance a portion of Lbest regularly for a dynamic multi-swarm differential learning particle swarm optimizer. Hu et al. [34] merged sub-gradient local search into PSO iteration. Cao et al. [35] adopted the quasi-entropy index to trigger local search. The aforementioned literature indicates that local search is an effective auxiliary approach to enhance exploitation of PSO.

Novel learning strategies are an active research direction of PSO. To enhance the adaptability of learning exemplar-based algorithms, scholars have employed two or more types of exemplars with different characteristics to guide the motion of particle swarm. For example, Fu et al. [36] developed an adjustable driving force-based particle swarm optimization by employing two types of exemplars to update particle velocities. Chen et al. [37] employed two DE mutations for generating learning exemplars of PSO, so the hybrid algorithm enhances exploration at the start and gradually changes over to enhance exploitation. Lynn et al. [38] adopted an ensemble mechanism to adjust the sizes of sub-swarms with different learning exemplars. Wang et al. proposed [39] a multiple-strategy learning particle swarm optimization for large-scale optimization problems by utilizing different learning strategies in different stages. Ning et al. [40] employed three particle swarms and three velocity update methods for high dimensional problems. Lynn et al. [41] proposed heterogeneous comprehensive learning particle swarm optimization (HCLPSO) by employing two sub-swarms to enhance exploration and exploitation, respectively. The test results show that HCLPSO yields high performance on different problems and outperforms seven state-of-the-art PSO variants on the CEC2005 test suite.

Due to the robustness performance of HCLPSO, this study further extends the heterogeneous theory by employing two DE mutants to construct diversified exploration and exploitation learning exemplars, respectively. The major contribution of this study is

1. Adopting two DE mutants to construct exploration and exploitation learning exemplars, respectively.
2. Introducing accompany operation to strengthen the impact of excellent particles.
3. Employing BFGS local search to further improve search accuracy.
4. The proposed heterogeneous differential evolution particle swarm optimization with local search (HeDE-PSO) is tested on CEC2017 test suite and Industrial Refrigeration System design problem.

The resultant algorithm is referred to as heterogeneous differential evolution particle swarm optimization with local search (HeDE-PSO). The rest of this paper is organized as follows: the next section reviews related works, the subsequent section presents the methodology, the following section reports the experimental results and the last section summarizes the paper.

Related works

Particle swarm optimization

PSO imitates the bird flock's foraging behavior to find the global optima. In PSO, each solution is regarded as a particle. The particles are guided by their own best position (Pbest) and the global best position (Gbest) according to Eqs. (1, 2) [42]. $\mathrm{Vel}_{i,d}$ denotes the particle's velocity, the subscripts i and d stand for the particle index and dimension, respectively. Pos and P stand for the Position and Pbest of the ith particle, respectively, and G denotes the global optima found by particle swarm. w, c_1 and c_2 are inertia weight and two acceleration coefficients, respectively. $r_{1,d}$ and $r_{2,d}$ are two uniformly distributed random numbers in the range of 0 and 1. In the learning process, the particles are oscillating in the neighborhood of Pbest and Gbest. The greedy selection algorithm is employed to update Pbest and Gbest.

$$\begin{aligned}\mathrm{Vel}_{i,d} = {} & w \cdot \mathrm{Vel}_{i,d} + c_1 \cdot r_{1,d} \cdot \left(P_{i,d} - \mathrm{Pos}_{i,d}\right) \\ & + c_2 \cdot r_{2,d} \cdot \left(G_d - \mathrm{Pos}_{i,d}\right),\end{aligned} \tag{1}$$

$$\mathrm{Pos}_{i,d} = \mathrm{Pos}_{i,d} + \mathrm{Vel}_{i,d}, \tag{2}$$

PSO converges fast while suffering from premature convergence. To acquire better exploration, some exemplar-based learning strategies are proposed. For example, Wu et al. [32] proposed a superior solution guided PSO (SSG-PSO) framework to fully utilize the valuable information of superior solutions found in the optimization process. References [9, 43] employed crossover, mutation and selection operators to construct high quality learning exemplars for PSO. Chen et al. [44] presented a biogeography-based learning strategy for PSO. Lu et al. [45] employed valid history information to guide the behavior of particles through a reinforcement learning strategy. Zhang et al. [46] merged Bayesian iteration method into comprehensive learning particle swarm optimization. Zhang et al. [47] employed self-organizing topological structure and self-adaptive adjustable parameters to improve the performance of PSO. Cheng et al. [48] have reviewed the developments of particle swarm optimization in the past quarter century.

Differential evolution

Differential evolution (DE) [49] is a versatile evolutionary optimization technique. Kinds of DE mutants are proposed to fulfill different optimization problems. The representative DE mutations are as follows:

$$\text{DE/rand/1}: V_i = X_{r1} + F(X_{r2} - X_{r3}), \tag{3}$$

$$\text{DE/best/1}: V_i = X_{\text{Best}} + F(X_{r2} - X_{r3}), \tag{4}$$

$$\text{DE/rand/2}: V_i = X_{r5} + F(X_{r1} - X_{r2}) + F(X_{r3} - X_{r4}), \tag{5}$$

$$\text{DE/best/2}: V_i = X_{\text{Best}} + F(X_{r1} - X_{r2}) + F(X_{r3} - X_{r4}), \tag{6}$$

$V_i = [v_{i,1}, v_{i,2},\ldots,v_{i,D}]$ and $X = [x_{i,1}, x_{i,2},\ldots,x_{i,D}]$ are the trial vector and target vector, respectively. F is the scale factor.r_1, r_2, r_3, r_4, and r_5 are five are mutually exclusive random integers within the range of $[1, NP]$, and NP denotes for the population size of DE.

DE generates new candidate solutions in three steps:

1. Mutation: Generate a mutant vector according to one DE mutant among Eqs. (3–6).
2. Crossover: Generate a random number for each dimension and compare it with the crossover probability according to Eq. (7) to determine whether to update the relative dimension of the trial vector $U_i = [u_{i,1}, u_{i,2},\ldots,u_{i,D}]$. cr_i is the crossover probability, and k is a random dimension to guarantee at least one dimension is chosen from mutant vector V_i.

$$u_{i,j} = \begin{cases} v_{i,j} & \text{rand} < \text{cr}_i \text{ or } j = k \\ x_{i,j} & \text{else} \end{cases}. \tag{7}$$

3. Selection: The fitness value of trial vector U_i is calculated and compared with the target vector X_i; if the fitness value of trial vector U_i is better than the target vector X_i, the target vector is replaced by the trial vector.

$$X_i = \begin{cases} U_i & f(U_i) \le f(X_i) \\ X_i & \text{else} \end{cases}. \tag{8}$$

DE/rand/1 and DE/rand/2 bear strong exploration properties, while DE/best/1 and DE/best/2 possess high exploitation properties. To balance exploration and exploitation, Zhang adopts DE/current-to-*p*best/1 to generate the trial vector according to Eq. (9) and proposes adaptive differential evolution with optional external archive (JADE) [50]. *rp* denotes a randomly selected target vector from the top *p*% individuals in Eq. (9). DE/current-to-*p*best/1 is widely used nowadays in DE variants for its excellent performance [51]. For more research on DE variants, please refer to reference [52].

$$\text{DE/current-to-pbest/1}: V_i = X_i + F\big(X_{rp} - X_i\big) + F(X_{r1} - X_{r2}). \tag{9}$$

Hybrid DE-PSO

PSO and DE are widely used stochastic algorithms. To utilize the advantages of both sides, a series of hybrid DE and PSO algorithms (referred to as DE-PSO in the following passages) are proposed. For example, Melton et al. [53] propose a hybrid DE-PSO for time-optimal slew-maneuver problems with path constraints. Once stagnation is detected, a set of DE iteration is employed. The experiments show that this hybrid DE-PSO can handle stagnation successfully and reduce CPU time by 40%. Song et al. [54] presented Gaussian particle swarm optimization with differential evolution to optimize three dimensional wind turbines. Wang et al. [55] proposed a self-adaptive mutation differential evolution algorithm based on particle swarm optimization. In this hybrid algorithm, DE/rand/1 is adopted to maintain swarm diversity in the early stage and PSO is employed to achieve high convergence speed. Mohammadi et al. [56] develop a hybrid algorithm for suspended sediment load estimation by merging multilayer perception and DE into PSO. To adjust a PID controller, Amal et al. [57] presented a hybrid algorithm based on differential evolution and particle swarm optimization with an aging leader and challengers. Li et al. [58] combined differential evolution with particle swarm optimization to calibrate camera parameters. Wang et al. [59] presented a self-adaptive particle swarm optimization differential evolution for parameter identification of lithium-ion batteries.

Methodologies

Heterogeneous exemplars

Heterogeneity is an effective approach to improve algorithms' adaptability. Both DE/rand/1 and DE/current-to-*p*best/1 are widely used DE mutants. Comparing with DE/rand/1, DE/current-to-*p*best/1 has stronger exploitation capability and converges relatively faster. In this study, DE/rand/l is adopted to construct exploration exemplars, while DE/current-to-*p*best/1 is utilized to construct exploitation exemplars. Compared with DE/best/1, DE/current-to-*p*best/1 is less liable to fall into local optima. The two DE mutants work in parallel, and the valuable information can be exchanged sufficiently. DE mutants employ the Pbests of the current swarm and archive to generate exemplars to challenge the relevant particle's Pbest. DE/rand/1 and DE/current-to-*p*best/1 generate exemplars according to Eqs. (10,11).

$$V_i = \text{Pbest}_{r1} + F(\text{Pbest}_{r2} + E\text{Pbest}_{r3}) \quad (i < Ps_1), \tag{10}$$

$$V_i = \text{Pbest}_i + F(\text{Pbest}_{i\text{best}} - \text{Pbest}_i) + F(\text{Pbest}_{r4} - E\text{Pbest}_{r5}). \tag{11}$$

Epbestr$_3$ and Epbestr$_5$ denote randomly selected Pbests from the union of the current swarm and archive. Ps_1 stands for the sub-swarm size of DE/rand/1, and the remaining particles generate learning exemplars according to DE/current-to-*p*best/1.

After mutation, the crossover operation and out of bounds treatment are carried out according to Eqs. (7) and (12, 13), correspondingly. Then the fitness value of the exemplar is evaluated and compared. If the fitness of the exemplar is better than the corresponding particle's Pbest, the exemplar will replace the previous Pbest, and the previous Pbest will challenge the worst solution in the archive. Otherwise, the generated exemplar will be abandoned.

$$u_{i,j} = (\text{Pbest}_{i,j} + \text{Up}_j)/2 \quad \text{if} \quad u_{i,j} > Up_{j,} \tag{12}$$

$$u_{i,j} = \frac{\text{Pbest}_{i,j} + \text{Low}_j}{2} \quad \text{if} \quad u_{i,j} < \text{Low}_j. \tag{13}$$

Accompanying learning

To avoid wasting computing resources, the accompanying learning strategy is introduced. In the PSO learning stage, each particle selects an accompanying particle randomly. If the learning particle's fitness is better than the accompany particle's, the PSO learning process will be executed according to Eq. (14). Otherwise, the PSO learning is skipped in this iteration. With accompanying learning, the best particles win more computing resources to improve the algorithm's performance.

$$\text{Vel}_{i,d} = w \cdot \text{Vel}_{i,d} + c \cdot r_d \big(\text{Pbest}_{i,d} - \text{Pos}_{i,d}\big). \tag{14}$$

BFGS

In the late stage of PSO, the swarm diversity is insufficient to maintain high convergence speed. Hence employing local search in the late stage can improve search accuracy. BFGS is a popular quasi-Newton method whose core is to approximate the inverse Hessian matrix. The major steps of BFGS [32] are given in Algorithm 1.

Algorithm 1

```
Let $z_0$, $\varepsilon$ and $H_0$ denote the starting point, tolerance and initial Hessian matrix, respectively. $k$=0 is the iteration number，$\nabla f_k$ stands for the gradient of the objective function.
While $k<k_{max}$
    $p_k = -H_k \nabla f_k$
     $z_{k+1} = z_k + \alpha_k p_k$
    $s_k = z_{k+1} - z_k$
    $y_k = \nabla f_{k+1} - \nabla f_k$
    $H_{k+1} = (I - p_k s_k y_k^T) H_k (I - p_k s_k y_k^T) - p_k s_k s_k^t$
    $k$=$k$+1
End while
```

Methodologies

The major steps of HeDE-PSO are illustrated in Fig. 1. In each iteration, DE/rand/1 and DE/current-to-*p*best/1 work in parallel to generate complementary exemplars for PSO. Then the fitness value of a particle is compared with that of the accompanying particle to determine whether to execute the PSO learning stage. In the late stage of optimization, the promising area is located and BFGS local search is employed to improve search accuracy. Details for HeDE-PSO are as follows:

Step 1: Initialize the initial population and control parameters. The fitness of the initial population is evaluated.

Step 2: Generate the trail vector through two DE mutants with different characteristics. DE/rand/1 is employed for exploration, and it updates individuals according to Eq. (10). DE/current-to-*p*best/1 is employed for enhancing exploitation, and it generates new individuals according to Eq. (11).

Step 3: Conduct crossover according to Eq. (7). Handle out of bounds trial vectors according to Eqs. (12, 13).

Step 4: Evaluate and compare the fitness value of a trial vector with the relevant particle's Pbest. If the fitness of the trial vector is better than its Pbest, replace the Pbest with its trial vector. Update *FEs*.

Step 5: Randomly select an accompanying particle. If the learning particle's fitness value is no worse than the accompany particle's, execute PSO according to Eq. (14). Otherwise, skip the PSO learning part (update the index of particle and *Fes,* jump to step 2).

Step 6: Compare the fitness of new position with its Pbest. If the former is better, replace the Pbest with its new position.

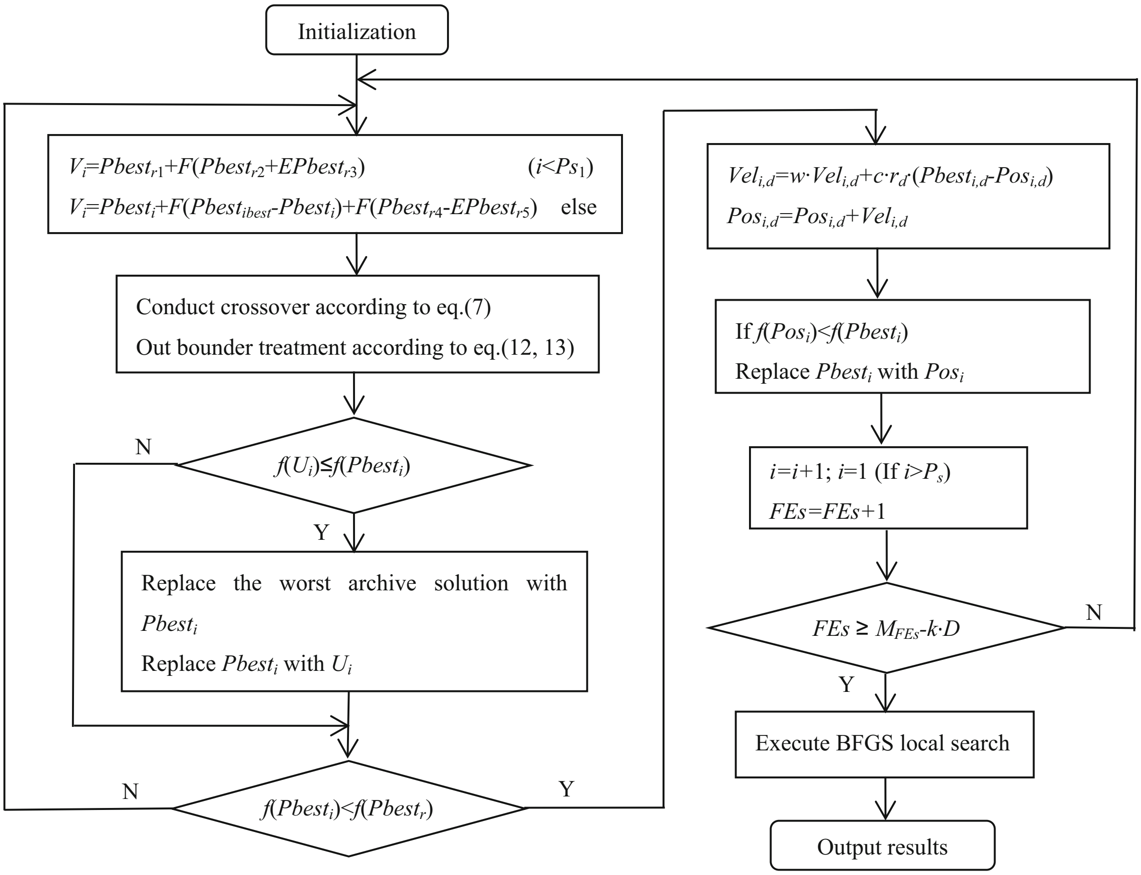

Initialization
V_i=$Pbest_{r1}$+F($Pbest_{r2}$+$EPbest_{r3}$) (i<Ps_1)
V_i=$Pbest_i$+F($Pbest_{ibest}$-$Pbest_i$)+F($Pbest_{r4}$-$EPbest_{r5}$) else
Conduct crossover according to eq.(7)
Out bounder treatment according to eq.(12, 13)
N
f(U_i)≤f($Pbest_i$)
Y
Replace the worst archive solution with $Pbest_i$
Replace $Pbest_i$ with U_i
N
f($Pbest_i$)<f($Pbest_r$)
Y
$Vel_{i,d}$=w·$Vel_{i,d}$+c·r_d·($Pbest_{i,d}$-$Pos_{i,d}$)
$Pos_{i,d}$=$Pos_{i,d}$+$Vel_{i,d}$
If f(Pos_i)<f($Pbest_i$)
Replace $Pbest_i$ with Pos_i
i=i+1; i=1 (If i>P_s)
FEs=FEs+1
FEs ≥ M_{FEs}-k·D
N
Y
Execute BFGS local search
Output results

Fig. 1 The flowchart of HeDE-PSO

Step 7: Update the index of particles and *FEs* (function of evolutions), if the index is bigger than the population, reset the index to 1.

Step 8: Compare *FEs* with PSO learning *FEs* (M_{FEs}-k·D, M_{FEs} is the maximum allowable *FEs*,*k* is an integer). If *FEs* is no bigger than PSO learning FEs, return to step 2, otherwise execute the following step.

Step 9: Conduct BFGS local search for the rest of *FEs* to improve accuracy.

The characteristics of HeDE-PSO are analyzed in "Ablation experiments".

Experimental works

To test the performance of HEDE-PSO, 29 CEC2017 functions [66] are adopted. f_2 is absent from this test. Six recent PSO variants, four meta-heuristics and JADE are employed for comparison tests. Among the comparison algorithms, biogeography-based learning particle swarm optimization (BLPSO) [31] updates each particle using the combination of its own personal best position and personal best positions of all other particles through the biogeography-based optimization (BBO) migration. Particle swarm optimization using dynamic tournament topology (DTT-PSO) [14] employs several better solutions choosing from the entire swarm to guide the evolution of each particle. Expanded particle swarm optimization based on multiple exemplars and forgetting ability (XPSO) [9] utilizes the locally best solution and the globally best solution to construct social learning exemplars and assigns different forgetting abilities to different particles. Fitness-based multi-role PSO (FMPSO) [60] divides the swarm into leaders, ramblers, and followers based on their fitness in each generation, and employs different learning strategies for different roles. Phasor particle swarm optimization (PPSO) [61] adjusts the control parameters of PSO based on phasor angle theory. Dynamic multiswarm differential learning particle swarm optimizer (DMSDL-PSO) [33] employs DE mutation to construct exploration enhanced exemplars and adopts BFGS local search in the late stage to improve accuracy. An adaptive particle swarm optimizer with decoupled exploration and exploitation (APSO-DEE) [29] employs two learning components to enhance exploration and exploitation, respectively. Multi-swarm strategy with adaptive sub-swarm size regulating is adopted to achieve better performance. The

Honey Badger Algorithm (HBA) [62] employs honey badgers' digging and honey finding approaches to construct the exploration phase and the exploitation phase, respectively. The salp swarm algorithm (SSA) [63] divides the swarm into leaders and follows according to the swarm behavior of salp chains. The leader guides the swarm, and the followers follow each other during the optimization process. The Archimedes optimization algorithm (AOA) [64] imitates the principle of buoyant force exerted upward on an object partially or fully immersed in fluid. AOA updates the density, volume and acceleration of every object to determine new positions in every generation. The dwarf mongoose optimization algorithm (DMO) [65] divides the swarm into the alpha group, babysitters, and the scout group. Each group contributes to compensatory behavioral adaptation, which leads to a semi-nomadic way of life in a territory large enough to support the entire group. Adaptive differential evolution with optional external archive (JADE) [42] improves the performance of DE by employing a new mutation strategy "DE/current-to-pbest/1" with optional external archive and adaptive updating control parameters. All the comparison algorithms adopt the authors recommended parameters configuration.

10-dimensional and 30-dimensional function experiments are conducted with the same parameter settings to evaluate the algorithms' scalability. Each function is run for 30 independent runs, and the Wilcoxon signed-rank test [67] with significance level of 0.05 is adopted for comparing test results. The parameter configurations of involved algorithms are given in Table 1. All the comparison algorithms adopt the authors' suggested parameter configurations.

Comparing with PSO algorithms

Table 2 shows the statistical results of HeDE-PSO and other PSO algorithms. The symbols ">", "≈ ", "<" denote that the performance of HeDE-PSO is significantly better than, tied with or significantly worse than the compared algorithms, according to the Wilcoxon signed-rank test, respectively. For example, in the first cell "> (7.62 ± 7.70)E + 02", ">" means the performance of HeDE-PSO is significantly better than BLPSO on f_1, and "7.62" and "7.70" are the mean error and standard deviation achieved by BLPSO on f_1, respectively. "E + 02" is the order of magnitude of scientific notation. In the last two rows, "Best" denotes for times of best mean performance achieved by the relevant algorithm, "w/t/l" stands for the times that HeDE-PSO performs "significantly better than", "tied with", or "significant worse than" the compared algorithm, respectively. In the 10D test, HeDE-PSO achieves the best performance on 16 functions, while DMSDL-PSO, BLPSO, DTT-PSO and FMPSO yield the best performance on 9, 5, 4 and 1 functions, respectively. XPSO, PPSO and APSO-DEE generate no best performances. HeDE-PSO performs significantly better than BLPSO, DTT-PSO, XPSO, FMPSO, PPSO, DMSDL-PSO and APSO-DEE on 16, 21, 23, 20, 27, 14 and 26 functions, respectively. HeDE-PSO performs worse than DMSDL-PSO on 7 functions. The overall performance of HEDE-PSO is better than DMSDL-PSO and other peer PSO algorithms. The average ranking of mean performance is given in Fig. 2, the lower average rank, the better. Figure 2 indicates that the average rank of HeDE-PSO is slightly lower than DMSDL-PSO, much better than other PSO algorithms. HeDE-PSO exhibits significant advantages except for ten composition functions (f_{21}–f_{30}). With DE to construct learning exemplars and BFGS local search, DMSDL-PSO yields high performance of composition functions.

Table 1 Parameter configuration of algorithms

Algorithm	Year	Parameters
HeDE-PSO	–	$Ps = 100$, $N_{ar} = 100$, $Ps_1 = 15$, $w = 0.7298$, $c = 1.4916$
BLPSO [44]	2017	$Ps = 40$, $w = 0.9$–0.2, $c = 1.496$, $I = 1, E = 1$
DTT-PSO [14]	2016	$Ps = 64$,$\phi = 4.1$, $P = 0.05$, $M = 6$, $k = 0.1$
XPSO [9]	2020	$Ps = 60$, $w = 0.9$–0.4, $c_1 = 1$, $c_2 = 0.5$, $c_3 = 0.5$
FMPSO [60]	2019	$Ps = 60$, $w = 0.9$–0.2
PPSO [61]	2019	$Ps = 40$, $\delta \in [0,2*\pi]$, $c_1 = \lvert\cos(\delta)\rvert^{2\sin(\delta)}$, $c_2 = \lvert\sin(\delta)\rvert^{2\cos(\delta)}$
DMSDL-PSO [33]	2018	$Ps = 40$, $w = 0.7298$, $c = 1.49618$, $R = 20$, $cr = 0.025$
APSO-DEE [29]	2021	$Ps = 40$, $\Phi = 0.3$
HBA [62]	2022	$N_P = 50$, $\beta = 6$, $c = 2$
SSA [63]	2017	$N_P = 30$, $c_1 = 2\exp(-(4*it/it_{\max})^2)$
AOA [64]	2021	$N_P = 30$, $c_1 = 2$, $c_2 = 6$, $c_3 = 2$, $c_4 = 0.5$, $u = 0.9$, $l = 0.1$
DMOA [65]	2022	$N_P = 50$, $N_b = 3$, $N_\alpha = 47$,
JADE [50]	2009	$N_P = 100$, $N_{ar} = 100$, $c = 0.1$, $p = 0.05$

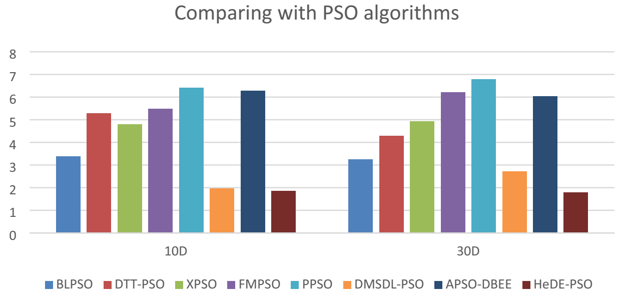


Fig. 2 Average ranking among PSO algorithms

Table 2 Comparison with PSO algorithms on 10D CEC2017 functions

Func	BLPSO	DTT-PSO	XPSO	FMPSO	PPSO	DMSDL-PSO	APSO-DEE	HeDE-PSO
f_1	> (7.62 ± 7.70)E + 02	> (2.54 ± 2.77)E + 02	> (1.72 ± 2.13)E + 03	> (4.32 ± 4.17)E + 03	> (1.85 ± 10.1)E + 03	> (2.94 ± 3.39)E + 01	> (1.70 ± 2.35)E + 03	**(0.00 ± 0.00)E + 00**
f_3	**≈ (0.00 ± 0.00)E + 00**	**≈ (0.00 ± 0.00)E + 00**	> (3.79 ± 3.11)E−14	**≈ (0.00 ± 0.00)E + 00**	> (1.10 ± 1.30)E−12	> (1.15 ± 1.79)E−12	> (2.36E ± 4.19 + 01)	**(0.00 ± 0.00)E + 00**
f_4	> (1.23 ± 0.13)E + 00	> (3.28 ± 0.56)E−01	> (2.05 ± 2.12)E−01	≈ (5.90 ± 18.8)E−06	> (5.02 ± 16.5)E + 00	≈ (2.99 ± 3.95)E−11	> (6.48 ± 0.70)E + 00	**(1.94 ± 3.74)E−11**
f_5	≈ (4.09 ± 1.45)E + 00	> (1.65 ± 0.52)E + 01	> (7.13 ± 3.35)E + 00	≈ (9.83 ± 6.33)E + 00	> (3.78 ± 1.36)E + 01	≈ (4.41 ± 1.03)E + 00	> (5.90 ± 2.89)E + 00	**(3.65 ± 0.97)E + 00**
f_6	**< (0.00 ± 0.00)E + 00**	< (1.01 ± 4.24)E−01	< (4.38 ± 12.1)E−07	> (1.05 ± 3.86)E−02	> (8.45 ± 9.46)E + 00	< (3.79 ± 20.8)E−15	≈ (6.15 ± 23.5)E−02	(4.89 ± 10.00)E−13
f_7	> (1.82 ± 0.22)E + 01	**< (1.31 ± 0.36)E + 01**	> (1.77 ± 0.36)E + 01	≈ (2.06 ± 0.72)E + 01	> (5.92 ± 1.97)E + 01	≈ (1.45 ± 0.18)E + 01	≈ (1.45 ± 0.21)E + 01	(1.49 ± 0.17)E + 01
f_8	> (4.95 ± 1.60)E + 00	> (1.05 ± 0.37)E + 01	> (6.42 ± 2.88)E + 00	≈ (9.99 ± 4.73)E + 00	> (2.89 ± 0.98)E + 01	≈ (4.58 ± 1.60)E + 00	≈ (5.47 ± 2.61)E + 00	**(3.78 ± 1.65)E + 00**
f_9	**≈ (0.00 ± 0.00)E + 00**	**≈ (0.00 ± 0.00)E + 00**	< (1.52 ± 3.94)E−14	≈ (7.63 ± 22.2)E−02	> (1.35 ± 1.29)E + 02	**≈ (0.00 ± 0.00)E + 00**	> (8.36 ± 21.1)E−02	**(0.00 ± 0.00)E + 00**
f_{10}	**< (6.68 ± 6.01)E + 01**	> (6.40 ± 1.85)E + 02	≈ (2.25 ± 1.46)E + 02	> (5.23 ± 2.37)E + 02	> (8.91 ± 2.99)E + 02	< (1.47 ± 0.84)E + 02	> (4.28 ± 2.53)E + 02	(1.82 ± 1.33)E + 02
f_{11}	> (1.77 ± 0.90)E + 00	> (1.30 ± 0.78)E + 01	> (2.80 ± 1.89)E + 00	> (5.58 ± 4.60)E + 00	> (3.31 ± 1.99)E + 01	> (1.39 ± 0.67)E + 00	> (1.25 ± 1.45)E + 01	**(6.00 ± 1.82)E−08**
f_{12}	> (1.33 ± 0.82)E + 04	> (6.39 ± 2.57)E + 03	> (1.53 ± 1.31)E + 04	> (2.77 ± 2.28)E + 04	> (5.58 ± 10.5)E + 03	> (4.40 ± 9.02)E + 02	> (9.84 ± 5.92)E + 03	**(9.95 ± 21.6)E + 00**
f_{13}	> (3.40 ± 3.71)E + 03	> (6.85 ± 2.15)E + 03	> (5.57 ± 4.25)E + 03	> (5.28 ± 7.74)E + 03	> (1.20 ± 1.54)E + 02	> (3.18 ± 2.61)E + 01	> (7.46 ± 5.98)E + 03	**(5.29 ± 1.51)E + 00**
f_{14}	> (2.01 ± 1.01)E + 01	> (1.86 ± 1.08)E + 03	> (3.62 ± 1.47)E + 01	> (1.22 ± 2.23)E + 02	> (4.05 ± 2.30)E + 01	> (1.59 ± 0.87)E + 00	> (3.62 ± 4.12)E + 03	**(1.54 ± 4.50)E−01**
f_{15}	> (2.25 ± 4.06)E + 01	> (2.11 ± 1.07)E + 03	> (3.79 ± 2.62)E + 01	> (3.82 ± 5.98)E + 02	> (8.59 ± 13.4)E + 01	> (1.77 ± 0.83)E + 00	> (2.84 ± 4.17)E + 03	**(1.05 ± 1.29)E−01**
f_{16}	> (1.06 ± 0.39)E + 00	> (2.93 ± 7.08)E + 02	> (9.39 ± 11.3)E + 01	> (4.23 ± 5.60)E + 01	> (1.57 ± 1.32)E + 02	> (1.04 ± 0.83)E + 00	> (8.71 ± 9.12)E + 01	**(4.99 ± 1.29)E−01**
f_{17}	≈ (6.92 ± 5.48)E + 00	> (4.30 ± 1.07)E + 01	> (2.78 ± 1.22)E + 01	> (3.13 ± 1.44)E + 01	> (5.14 ± 3.61)E + 01	**< (8.90 ± 5.44)E−01**	> (3.66 ± 2.29)E + 01	(7.21 ± 5.43)E + 00
f_{18}	> (1.31 ± 1.66)E + 03	> (6.84 ± 4.57)E + 03	> (6.91 ± 5.53)E + 03	> (8.72 ± 8.83)E + 03	> (1.27 ± 6.01)E + 03	> (1.88 ± 1.36)E + 01	> (9.54 ± 8.74)E + 03	**(7.29 ± 1.21)E−01**

Table 2 (continued)

Func	BLPSO	DTT-PSO	XPSO	FMPSO	PPSO	DMSDL-PSO	APSO-DEE	HeDE-PSO
f_{19}	> (7.46 ± 25.4)E + 01	> (4.36 ± 1.85)E + 03	> (3.45 ± 3.70)E + 01	> (2.66 ± 3.68)E + 03	> (8.65 ± 8.76)E + 00	> (6.18 ± 4.23)E−01	> (4.75 ± 5.09)E + 03	**(1.72 ± 1.96)E−02**
f_{20}	> (1.73 ± 5.07)E + 00	> (1.21 ± 0.49)E + 02	> (3.52 ± 4.71)E + 01	> (2.56 ± 1.07)E + 01	> (5.86 ± 3.91)E + 01	**< (1.04 ± 5.70)E−02**	> (3.83 ± 3.62)E + 01	(1.46 ± 2.13)E−01
f_{21}	≈ (1.87 ± 3.80)E + 02	> (2.09 ± 2.11)E + 02	> (1.86 ± 0.44)E + 02	≈ (1.94 ± 0.51)E + 02	≈ (1.36 ± 0.60)E + 02	**≈ (1.18 ± 0.41)E + 02**	> (2.00 ± 0.26)E + 02	(1.63 ± 0.52)E + 02
f_{22}	≈ (1.01 ± 0.00)E + 02	≈ (1.00 ± 0.00)E + 02	≈ (1.01 ± 0.01)E + 02	≈ (9.21 ± 2.66)E + 01	≈ (1.09 ± 0.03)E + 02	**≈ (8.56 ± 3.01)E + 01**	> (1.01 ± 0.01)E + 02	(1.00 ± 0.00)E + 02
f_{23}	**< (3.03 ± 0.02)E + 02**	> (3.20 ± 0.07)E + 02	> (3.08 ± 0.04)E + 02	> (3.10 ± 0.07)E + 02	> (3.29 ± 0.14)E + 02	> (3.07 ± 0.01)E + 02	> (3.12 ± 0.05)E + 02	(3.04 ± 0.02)E + 02
f_{24}	< (3.05 ± 0.65)E + 02	≈ (2.66 ± 1.11)E + 02	> (2.82 ± 1.02)E + 02	> (3.32 ± 0.49)E + 02	> (3.67 ± 0.55)E + 02	**< (1.57 ± 0.91)E + 02**	> (3.32 ± 0.44)E + 02	(3.32 ± 0.02)E + 02
f_{25}	≈ (4.11 ± 0.19)E + 02	> (4.33 ± 0.19)E + 02	> (4.20 ± 0.24)E + 02	> (4.19 ± 0.26)E + 02	> (4.32 ± 0.22)E + 02	**< (3.58 ± 1.03)E + 02**	> (4.31 ± 0.23)E + 02	(4.16 ± 0.22)E + 02
f_{26}	≈ (3.00 ± 0.00)E + 02	< (2.83 ± 0.38)E + 02	≈ (2.82 ± 0.62)E + 02	> (3.56 ± 1.83)E + 02	> (4.93 ± 3.64)E + 02	**< (1.17 ± 1.15)E + 02**	> (3.37 ± 0.86)E + 02	(3.00 ± 0.00)E + 02
f_{27}	> (3.91 ± 0.03)E + 02	**< (3.75 ± 0.02)E + 02**	> (3.99 ± 0.16)E + 02	> (3.97 ± 0.10)E + 02	> (4.02 ± 0.12)E + 02	> (3.90 ± 0.01)E + 02	> (4.03 ± 0.07)E + 02	(3.89 ± 0.01)E + 02
f_{28}	> (4.14 ± 1.32)E + 02	> (3.82 ± 0.16)E + 02	> (4.71 ± 1.53)E + 02	≈ (5.41 ± 1.53)E + 02	> (5.31 ± 2.23)E + 02	**≈ (2.66 ± 0.99)E + 02**	> (5.49 ± 1.12)E + 02	(3.30 ± 0.48)E + 02
f_{29}	> (2.57 ± 0.08)E + 02	> (2.98 ± 0.19)E + 02	> (2.58 ± 0.21)E + 02	> (2.94 ± 0.61)E + 02	> (3.19 ± 0.66)E + 02	> (2.52 ± 0.13)E + 02	> (2.85 ± 0.27)E + 02	**(2.38 ± 0.0.3)E + 02**
f_{30}	> (4.24 ± 1.93)E + 03	> (1.97 ± 0.95)E + 03	> (3.71 ± 7.69)E + 05	> (3.74 ± 5.49)E + 05	> (2.48 ± 5.16)E + 06	> (1.24 ± 0.88)E + 03	> (2.56 ± 4.86)E + 05	**(4.10 ± 0.18)E + 02**
Best	5	4	0	1	0	9	0	**16**
w/t/l	16/8/5	21/4/4	23/4/2	20/9/0	27/2/0	14/8/7	26/0/3	

The best test results are highlighted in bold

On 30D functions, Table 3 indicates that HeDE-PSO has the best performance on 17 functions, DTT-PSO, BLPSO and DMSDL-PSO have the best performance on 6, 4 and 4 functions, respectively. XPSO, FMPSO, PPSO and APSO-DEE yield no best performances. HeDE-PSO generates the best performance on all ten hybrid functions (f_{11}–f_{20}), while with seven simple multimodal functions (f_4–f_{10}), HeDE-PSO only wins the best performance on one function. HeDE-PSO outperforms BLPSO, DTT-PSO, XPSO, FMPSO, PPSO, DMSDL-PSO and APSO-DEE on 20, 22, 27, 28, 29, 21 and 29 functions, respectively. Figure 2 shows that the average rank of HeDE-PSO is lower than those of other PSO algorithms. The advantages of HeDE-PSO on 30D functions are more significant than on 10D functions. DMSDL-PSO and BLPSO perform better than the remaining PSO algorithms. With two DE mutations to construct exploration and exploitation learning exemplars, HeDE-PSO relieves premature convergence successfully and exhibits better adaptation than other PSO algorithms. HeDE-PSO outperforms APSO-DEE on both 10D and 30D CEC2017 test suite, indicates that employing two sub-swarms to enhance exploration and exploitation, respectively, performs better than utilizing two learning components to enhance exploration and exploitation in this study.

Comparison with other meta-heuristics

The comparison test results with other meta-heuristics on 30D functions in Table 4 show that HeDE-PSO has the best performance on 19 functions. JADE, DMOA and SSA yield the best performance on 7, 3 and 1 functions, respectively. HeDE-PSO outperforms HBA, SSA, AOA, DMOA and JADE on 29, 28, 29, 25 and 16 functions, respectively. The average rank in Fig. 3 shows that HeDE-PSO ranks the first, while JADE and HBA occupy the second and the third places, respectively. HeDE-PSO outperforms four recent meta-heuristics and JADE in this test.

Convergence analysis

To analyze the convergence speed of HeDE-PSO, the convergence curves of HeDE-PSO and other PSO algorithms on four different types of functions are given in Fig. 4. Figure 4a indicates that on the unimodal function f_1, HeDE-PSO achieves the highest accuracy, DMSDL-PSO ranks the second, and the rest of the PSO algorithms yield almost the same mean errors. Figure 4b shows that on the simple multimodal function f_6, DMSDL-PSO generates the lowest mean error, and HeDE-PSO and BLSPO rank second and the third, respectively. The rest of the PSO algorithms yield relatively bigger mean errors. On composition function f_{16}, Fig. 4c shows that HeDE-PSO yields the lowest mean error, BLPSO ranks second, and the other algorithms yield a relatively bigger mean error. Figure 4d reveals that DTT-PSO converges faster and yields the lowest mean error, and DMSDL-PSO ranks the second. The mean error of HeDE-PSO is relatively bigger. HeDE-PSO has significant advantages on unimodal functions and hybrid functions, while on simple multimodal functions and composition, HeDE-PSO yields moderate performance.

Ablation experiments

In this section, ablation experiments are carried out to show the effectiveness of each strategy employed by HeDE-PSO. Five contrast algorithms denoted by NoLS-PSO, Noapy-PSO, NoDE-PSO, DE^{r1}-PSO, and DE$^{p\text{best}}$-PSO are modified from HeDE-PSO. Each contrast algorithm is developed by removing one strategy from HeDE-PSO in "Methodologies". Details of five contrast algorithms are given in Table 5.

The test results are given in Table 6. The Wilcoxon signed-rank test shows that HeDE-PSO outperforms any contrast algorithm on at least twenty functions. HeDE-PSO outperforms NoLS-PSO on 23 functions, which indicates that the BFGS local search can improve search accuracy in the late stage of HeDE-PSO. HeDE-PSO performs better on 20 functions than Noapy-PSO, showing that the accompanying operation can allocate more computing resources to high quality particles to achieve high performance. HeDE-PSO defeats NoDE-PSO, DE^{r1}-PSO, DE$^{p\text{best}}$-PSO on 25, 24 and 21 functions, respectively, indicating that employing both DE/rand/1 and DE/current-to-*p*best/1 to generate learning exemplars (heterogeneous DE exemplars) can achieve better performance than employing a single DE mutant or not using DE mutation to construct learning exemplars. HeDE-PSO performs better than all the contrast algorithms and yields the best performance on 21 functions, indicating that the BFGS local search, the accompanying operation and the heterogeneous DE exemplar are indispensable for HeDE-PSO.

To analyze the proposed strategies in balancing exploitation and diversity, the convergence curves and diversity curves on unimodal f_1 and multimodal f_{17} are given in Fig. 5. The diversity is evaluated by the mean Euler distance between all the particles and their centroid.

Figure 5a shows that with both DE/rand/1 and DE/current-to-*p*best/1 to construct learning exemplars, HeDE-PSO outperforms NoDE-PSO, DE^{r1}-PSO and DE$^{p\text{best}}$-PSO. DE$^{p\text{best}}$-PSO converges fast in the early stage and performs better than NoDE-PSO and DE^{r1}-PSO, indicating that employing DE/current-to-*p*best/1 to generate learning exemplars (DE/current-to-*p*best/1 exemplars) can achieve high convergence speed on unimodal f_1. Figure 5b shows that HeDE-PSO can keep moderate diversity in the early stage and converge to the high quality area in the later stage.

Table 3 Comparison with PSO algorithms on 30D CEC2017 functions

Func	BLPSO	DTT-PSO	XPSO	FMPSO	PPSO	DMSDL-PSO	APSO-DEE	HeDE-PSO
f_1	> (7.62 ± 7.70)E + 02	> (2.54 ± 2.77)E + 02	> (1.72 ± 2.13)E + 03	> (4.32 ± 4.17)E + 03	> (1.85 ± 10.1)E + 03	> (2.94 ± 3.39)E + 01	> (4.83 ± 4.94)E + 03	**(0.00 ± 0.00)E + 00**
f_3	**≈ (0.00 ± 0.00)E + 00**	**≈ (0.00 ± 0.00)E + 00**	> (3.79 ± 3.11)E−14	**≈ (0.00 ± 0.00)E + 00**	> (1.10 ± 1.30)E−12	> (1.15 ± 1.79)E−12	> (7.04 ± 2.30)E + 04	**(0.00 ± 0.00)E + 00**
f_4	> (1.23 ± 0.13)E + 00	> (3.28 ± 0.56)E−01	> (2.05 ± 2.12)E−01	≈ (5.90 ± 18.8)E−06	> (5.02 ± 16.5)E + 00	≈ (2.99 ± 3.95)E−11	> (1.17 ± 0.28)E + 02	**(1.94 ± 3.74)E−11**
f_5	≈ (4.09 ± 1.45)E + 00	> (1.65 ± 0.52)E + 01	> (7.13 ± 3.35)E + 00	≈ (9.83 ± 6.33)E + 00	> (3.78 ± 1.36)E + 01	≈ (4.41 ± 1.03)E + 00	> (4.01 ± 0.86)E + 01	**(3.65 ± 0.97)E + 00**
f_6	**< (0.00 ± 0.00)E + 00**	< (1.01 ± 4.24)E−01	< (4.38 ± 12.1)E−07	> (1.05 ± 3.86)E−02	> (8.45 ± 9.46)E + 00	< (3.79 ± 20.8)E−15	> (7.57 ± 6.00)E−01	(4.89 ± 10.00)E−13
f_7	> (1.82 ± 0.22)E + 01	**< (1.31 ± 0.36)E + 01**	> (1.77 ± 0.36)E + 01	≈ (2.06 ± 0.72)E + 01	> (5.92 ± 1.97)E + 01	≈ (1.45 ± 0.18)E + 01	> (6.76 ± 1.05)E + 01	(1.49 ± 0.17)E + 01
f_8	> (4.95 ± 1.60)E + 00	> (1.05 ± 0.37)E + 01	> (6.42 ± 2.88)E + 00	≈ (9.99 ± 4.73)E + 00	> (2.89 ± 0.98)E + 01	≈ (4.58 ± 1.60)E + 00	> (3.97 ± 0.95)E + 01	**(3.78 ± 1.65)E + 00**
f_9	**≈ (0.00 ± 0.00)E + 00**	**≈ (0.00 ± 0.00)E + 00**	< (1.52 ± 3.94)E−14	≈ (7.63 ± 22.2)E−02	> (1.35 ± 1.29)E + 02	**≈ (0.00 ± 0.00)E + 00**	> (1.11 ± 1.02)E + 02	**(0.00 ± 0.00)E + 00**
f_{10}	**< (6.68 ± 6.01)E + 01**	> (6.40 ± 1.85)E + 02	≈ (2.25 ± 1.46)E + 02	> (5.23 ± 2.37)E + 02	> (8.91 ± 2.99)E + 02	< (1.47 ± 0.84)E + 02	> (2.36 ± 0.41)E + 03	(1.82 ± 1.33)E + 02
f_{11}	> (1.77 ± 0.90)E + 00	> (1.30 ± 0.78)E + 01	> (2.80 ± 1.89)E + 00	> (5.58 ± 4.60)E + 00	> (3.31 ± 1.99)E + 01	> (1.39 ± 0.67)E + 00	> (1.71 ± 0.72)E + 02	**(6.00 ± 1.82)E−08**
f_{12}	> (1.33 ± 0.82)E + 04	> (6.39 ± 2.57)E + 03	> (1.53 ± 1.31)E + 04	> (2.77 ± 2.28)E + 04	> (5.58 ± 10.5)E + 03	> (4.40 ± 9.02)E + 02	> (9.64 ± 9.30)E + 05	**(9.95 ± 21.6)E + 00**
f_{13}	> (3.40 ± 3.71)E + 03	> (6.85 ± 2.15)E + 03	> (5.57 ± 4.25)E + 03	> (5.28 ± 7.74)E + 03	> (1.20 ± 1.54)E + 02	> (3.18 ± 2.61)E + 01	> (1.50 ± 0.93)E + 04	**(5.29 ± 1.51)E + 00**
f_{14}	> (2.01 ± 1.01)E + 01	> (1.86 ± 1.08)E + 03	> (3.62 ± 1.47)E + 01	> (1.22 ± 2.23)E + 02	> (4.05 ± 2.30)E + 01	> (1.59 ± 0.87)E + 00	> (1.45 ± 1.56)E + 05	**(1.54 ± 4.50)E−01**
f_{15}	> (2.25 ± 4.06)E + 01	> (2.11 ± 1.07)E + 03	> (3.79 ± 2.62)E + 01	> (3.82 ± 5.98)E + 02	> (8.59 ± 13.4)E + 01	> (1.77 ± 0.83)E + 00	> (3.17 ± 4.43)E + 03	**(1.05 ± 1.29)E−01**
f_{16}	> (1.06 ± 0.39)E + 00	> (2.93 ± 7.08)E + 02	> (9.39 ± 11.3)E + 01	> (4.23 ± 5.60)E + 01	> (1.57 ± 1.32)E + 02	> (1.04 ± 0.83)E + 00	> (6.48 ± 2.78)E + 02	**(4.99 ± 1.29)E−01**
f_{17}	≈ (6.92 ± 5.48)E + 00	> (4.30 ± 1.07)E + 01	> (2.78 ± 1.22)E + 01	> (3.13 ± 1.44)E + 01	> (5.14 ± 3.61)E + 01	**< (8.90 ± 5.44)E−01**	> (2.23 ± 1.16)E + 02	(7.21 ± 5.43)E + 00
f_{18}	> (1.31 ± 1.66)E + 03	> (6.84 ± 4.57)E + 03	> (6.91 ± 5.53)E + 03	> (8.72 ± 8.83)E + 03	> (1.27 ± 6.01)E + 03	> (1.88 ± 1.36)E + 01	> (5.95 ± 5.96)E + 05	**(7.29 ± 1.21)E−01**

Table 3 (continued)

Func	BLPSO	DTT-PSO	XPSO	FMPSO	PPSO	DMSDL-PSO	APSO-DEE	HeDE-PSO
f_{19}	> (7.46 ± 25.4)E + 01	> (4.36 ± 1.85)E + 03	> (3.45 ± 3.70)E + 01	> (2.66 ± 3.68)E + 03	> (8.65 ± 8.76)E + 00	> (6.18 ± 4.23)E−01	> (6.64 ± 6.24)E + 03	**(1.72 ± 1.96)E−02**
f_{20}	> (1.73 ± 5.07)E + 00	> (1.21 ± 0.49)E + 02	> (3.52 ± 4.71)E + 01	> (2.56 ± 1.07)E + 01	> (5.86 ± 3.91)E + 01	**< (1.04 ± 5.70)E−02**	> (3.05 ± 1.58)E + 02	(1.46 ± 2.13)E−01
f_{21}	≈ (1.87 ± 3.80)E + 02	> (2.09 ± 2.11)E + 02	> (1.86 ± 0.44)E + 02	≈ (1.94 ± 0.51)E + 02	≈ (1.36 ± 0.60)E + 02	**≈ (1.18 ± 0.41)E + 02**	> (2.43 ± 0.12)E + 02	(1.63 ± 0.52)E + 02
f_{22}	≈ (1.01 ± 0.00)E + 02	≈ (1.00 ± 0.00)E + 02	≈ (1.01 ± 0.01)E + 02	≈ (9.21 ± 2.66)E + 01	≈ (1.09 ± 0.03)E + 02	**≈ (8.56 ± 3.01)E + 01**	> (1.06 ± 0.15)E + 02	(1.00 ± 0.00)E + 02
f_{23}	**< (3.03 ± 0.02)E + 02**	> (3.20 ± 0.07)E + 02	> (3.08 ± 0.04)E + 02	> (3.10 ± 0.07)E + 02	> (3.29 ± 0.14)E + 02	> (3.07 ± 0.01)E + 02	> (4.24 ± 0.27)E + 02	(3.04 ± 0.02)E + 02
f_{24}	< (3.05 ± 0.65)E + 02	≈ (2.66 ± 1.11)E + 02	> (2.82 ± 1.02)E + 02	> (3.32 ± 0.49)E + 02	> (3.67 ± 0.55)E + 02	**< (1.57 ± 0.91)E + 02**	> (4.80 ± 0.26)E + 02	(3.32 ± 0.02)E + 02
f_{25}	≈ (4.11 ± 0.19)E + 02	> (4.33 ± 0.19)E + 02	> (4.20 ± 0.24)E + 02	> (4.19 ± 0.26)E + 02	> (4.32 ± 0.22)E + 02	**< (3.58 ± 1.03)E + 02**	> (4.17 ± 0.22)E + 02	(4.16 ± 0.22)E + 02
f_{26}	≈ (3.00 ± 0.00)E + 02	< (2.83 ± 0.38)E + 02	≈ (2.82 ± 0.62)E + 02	> (3.56 ± 1.83)E + 02	> (4.93 ± 3.64)E + 02	**< (1.17 ± 1.15)E + 02**	> (1.70 ± 0.58)E + 03	(3.00 ± 0.00)E + 02
f_{27}	> (3.91 ± 0.03)E + 02	**< (3.75 ± 0.02)E + 02**	> (3.99 ± 0.16)E + 02	> (3.97 ± 0.10)E + 02	> (4.02 ± 0.12)E + 02	> (3.90 ± 0.01)E + 02	> (5.42 ± 0.20)E + 02	(3.89 ± 0.01)E + 02
f_{28}	> (4.14 ± 1.32)E + 02	> (3.82 ± 0.16)E + 02	> (4.71 ± 1.53)E + 02	≈ (5.41 ± 1.53)E + 02	> (5.31 ± 2.23)E + 02	**≈ (2.66 ± 0.99)E + 02**	> (4.76 ± 0.27)E + 02	(3.30 ± 0.48)E + 02
f_{29}	> (2.57 ± 0.08)E + 02	> (2.98 ± 0.19)E + 02	> (2.58 ± 0.21)E + 02	> (2.94 ± 0.61)E + 02	> (3.19 ± 0.66)E + 02	> (2.52 ± 0.13)E + 02	> (7.78 ± 1.74)E + 02	**(2.38 ± 0.0.3)E + 02**
f_{30}	> (4.24 ± 1.93)E + 03	> (1.97 ± 0.95)E + 03	> (3.71 ± 7.69)E + 05	> (3.74 ± 5.49)E + 05	> (2.48 ± 5.16)E + 06	> (1.24 ± 0.88)E + 03	> (1.51 ± 1.08)E + 04	**(4.10 ± 0.18)E + 02**
Best	5	4	0	1	0	9	0	**16**
w/t/l	16/8/5	21/4/4	23/4/2	20/9/0	27/2/0	14/8/7	29/0/0	

The best test results are highlighted in bold

Table 4 Comparison with other meta-heuristics on 30D CEC2017 functions

Func	HBA	SSA	AOA	DMOA	JADE	HeDE-PSO
f_1	> (4.35 ± 4.34)E + 03	> (5.38 ± 5.84)E + 03	> (3.17 ± 12.7)E + 08	> (4.59 ± 4.57)E + 03	> (1.36 ± 0.28)E−14	**(0.00 ± 0.00)E + 00**
f_3	≈ (1.82 ± 6.33)E−04	< **(3.43 ± 0.89)E−08**	> (3.11 ± 2.61)E + 03	> (2.33 ± 0.69)E + 05	< (6.86 ± 16.2)E + 03	(4.32 ± 8.02)E−05
f_4	> (5.49 ± 3.32)E + 01	> (8.90 ± 2.18)E + 01	> (4.94 ± 0.29)E + 02	> (3.05 ± 1.73)E + 01	> (4.88 ± 2.61)E + 01	**(1.09 ± 1.78)E + 00**
f_5	> (1.11 ± 0.25)E + 02	> (1.12 ± 0.37)E + 02	> (7.06 ± 0.26)E + 02	> (1.94 ± 0.14)E + 02	> (2.64 ± 0.37)E + 01	**(2.36 ± 0.77)E + 01**
f_6	> (4.03 ± 3.48)E + 00	> (2.88 ± 0.91)E + 01	> (6.44 ± 0.12)E + 02	< (6.10 ± 14.6)E−13	≈ **(1.55 ± 0.55)E−13**	(2.43 ± 8.80)E−07
f_7	> (1.73 ± 0.43)E + 02	> (1.55 ± 0.50)E + 02	> (1.18 ± 0.13)E + 03	> (2.25 ± 0.11)E + 02	> (5.49 ± 0.37)E + 01	**(5.08 ± 0.75)E + 01**
f_8	> (1.01 ± 0.24)E + 02	> (1.20 ± 0.38)E + 02	> (9.33 ± 0.20)E + 02	> (1.92 ± 0.12)E + 02	≈ **(2.54 ± 0.31)E + 01**	(2.59 ± 0.54)E + 01
f_9	> (1.24 ± 0.78)E + 03	> (2.02 ± 1.57)E + 03	> (4.21 ± 0.84)E + 03	≈ **(1.89 ± 4.32)E−14**	≈ (7.16 ± 0.00)E−03	(8.79 ± 33.0)E−02
f_{10}	> (4.22 ± 1.12)E + 03	> (3.81 ± 0.73)E + 03	> (4.54 ± 0.64)E + 03	> (7.77 ± 0.31)E + 03	< **(1.87 ± 0.26)E + 03**	(2.73 ± 0.50)E + 03
f_{11}	> (1.19 ± 0.54)E + 02	> (1.59 ± 0.56)E + 02	> (1.23 ± 0.07)E + 03	> (8.42 ± 0.97)E + 01	> (3.49 ± 2.52)E + 01	**(7.80 ± 0.32)E + 00**
f_{12}	> (3.79 ± 1.84)E + 04	> (1.34 ± 0.81)E + 06	> (9.93 ± 33.0)E + 06	> (1.86 ± 1.75)E + 07	≈ **(1.20 ± 0.39)E + 03**	(1.57 ± 1.68)E + 03
f_{13}	> (1.79 ± 8.24)E + 05	> (1.04 ± 0.81)E + 05	> (7.13 ± 5.56)E + 04	> (8.78 ± 7.48)E + 03	< **(4.30 ± 2.55)E + 01**	(5.68 ± 2.64)E + 01
f_{14}	> (1.84 ± 1.24)E + 03	> (3.44 ± 2.89)E + 03	> (1.15 ± 0.84)E + 04	> (4.77 ± 2.72)E + 04	> (3.71 ± 6.54)E + 03	**(3.09 ± 0.66)E + 01**
f_{15}	> (7.05 ± 8.94)E + 03	> (5.08 ± 3.19)E + 04	> (7.93 ± 6.12)E + 03	> (1.02 ± 1.13)E + 03	> (2.54 ± 4.43)E + 03	**(1.21 ± 0.75)E + 01**
f_{16}	> (8.82 ± 2.81)E + 02	> (8.77 ± 3.26)E + 02	> (2.87 ± 0.30)E + 03	> (1.76 ± 0.17)E + 03	> (3.72 ± 1.38)E + 02	**(1.09 ± 1.39)E + 02**
f_{17}	> (4.28 ± 2.16)E + 02	> (3.39 ± 1.87)E + 02	> (2.45 ± 0.26)E + 03	> (7.36 ± 1.24)E + 02	> (8.58 ± 4.30)E + 01	**(3.32 ± 0.76)E + 01**
f_{18}	> (9.86 ± 7.35)E + 04	> (1.76 ± 1.15)E + 05	> (1.76 ± 1.50)E + 05	> (3.42 ± 1.70)E + 06	> (1.86 ± 4.43)E + 04	**(2.78 ± 0.39)E + 01**
f_{19}	> (1.20 ± 1.51)E + 04	> (2.99 ± 1.28)E + 05	> (1.93 ± 3.06)E + 04	> (1.72 ± 2.05)E + 03	> (1.55 ± 3.22)E + 03	**(1.04 ± 0.33)E + 01**
f_{20}	> (4.81 ± 2.43)E + 02	> (3.93 ± 1.14)E + 02	> (2.37 ± 0.16)E + 03	> (7.49 ± 1.85)E + 02	> (1.21 ± 0.61)E + 02	**(2.37 ± 2.16)E + 01**
f_{21}	> (3.02 ± 0.27)E + 02	> (3.11 ± 0.28)E + 02	> (2.47 ± 0.03)E + 03	> (3.99 ± 0.08)E + 02	≈ (2.27 ± 0.04)E + 02	**(2.23 ± 0.06)E + 02**
f_{22}	> (1.26 ± 2.02)E + 03	> (2.23 ± 1.85)E + 03	> (5.32 ± 1.72)E + 03	> (7.71 ± 0.38)E + 03	≈ **(1.00 ± 0.00)E + 02**	**(1.00 ± 0.00)E + 02**
f_{23}	> (4.68 ± 0.37)E + 02	> (4.64 ± 0.25)E + 02	> (3.18 ± 0.26)E + 03	> (5.43 ± 0.09)E + 02	≈ (3.74 ± 0.05)E + 02	**(3.71 ± 0.08)E + 02**
f_{24}	> (6.07 ± 1.91)E + 02	> (5.19 ± 0.30)E + 02	> (3.10 ± 0.20)E + 03	> (6.28 ± 0.10)E + 02	≈ **(4.39 ± 0.06)E + 02**	(4.39 ± 0.06)E + 02
f_{25}	> (3.92 ± 0.14)E + 02	> (3.99 ± 0.21)E + 02	> (2.93 ± 0.03)E + 03	< **(3.79 ± 0.00)E + 02**	> (3.87 ± 0.00)E + 02	(3.80 ± 0.00)E + 02
f_{26}	> (2.17 ± 0.81)E + 03	> (1.98 ± 0.76)E + 03	> (4.68 ± 1.73)E + 03	> (2.66 ± 0.08)E + 03	> (1.20 ± 0.08)E + 03	**(1.13 ± 0.10)E + 03**

Table 4 (continued)

Func	HBA	SSA	AOA	DMOA	JADE	HeDE-PSO
f_{27}	> (6.20 ± 1.02)E + 02	> (5.31 ± 0.13)E + 02	> (3.21 ± 0.05)E + 03	> (5.00 ± 0.00)E + 02	> (5.04 ± 0.08)E + 02	**(4.92 ± 0.12)E + 02**
f_{28}	> (3.40 ± 0.64)E + 02	> (3.96 ± 0.42)E + 02	> (3.30 ± 0.03)E + 03	> (4.99 ± 0.03)E + 02	≈ (3.48 ± 0.62)E + 02	**(3.37 ± 0.55)E + 02**
f_{29}	> (1.11 ± 0.35)E + 03	> (1.05 ± 0.27)E + 03	> (3.94 ± 0.39)E + 03	> (1.29 ± 0.23)E + 03	> (4.76 ± 0.34)E + 02	**(4.39 ± 0.57)E + 02**
f_{30}	> (1.31 ± 0.89)E + 04	> (1.19 ± 0.72)E + 06	> (7.30 ± 31.2)E + 04	< **(7.42 ± 4.73)E + 02**	> (2.16 ± 0.15)E + 03	(2.05 ± 0.10)E + 03
Best	0	1	0	3	7	19
w/t/l	29/1/0	28/0/1	29/0/0	25/1/3	16/9/3	

The best test results are highlighted in bold

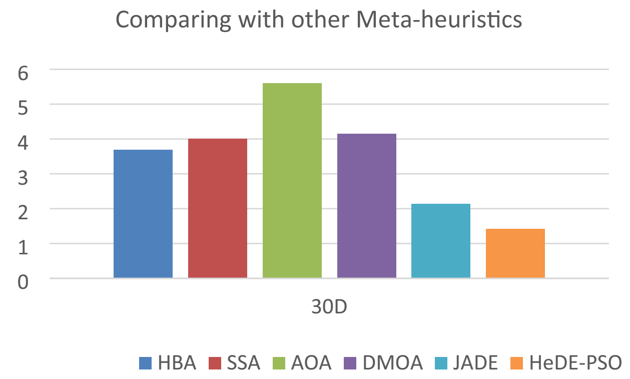


Fig. 3 Average ranking among meta-heuristics

The diversity of HeDE-PSO is bigger than that of DE^{pbest}-PSO but smaller than that of DE^{r1}-PSO. Without DE to construct learning exemplars, NoDE-PSO cannot converge to a high quality solution. Figure 5c indicates that on multimodal functions f_{17}, HeDE-PSO ranks the first, and DE^{r1}-PSO, DE^{pbest}-PSO and NoDE-PSO rank second, third and fourth, respectively. HeDE-PSO employing heterogeneous DE exemplars performs better than employing only either the DE/rand/1 exemplar or the DE/current-to-*p*best/1 exemplar. Adopting the DE/rand/1 exemplar performs better than employing DE/current-to-*p*best/1 exemplar. Figure 5d shows that on multimodal f_{17}, only HeDE-PSO converges to a high-quality solution, while other contrast algorithms do not converge sufficiently to achieve high accuracy. With heterogeneous DE exemplar, HeDE-PSO can keep proper swarm diversity to avoid local optima. HeDE-PSO generates exemplars by exploration enhanced DE mutants and exploration/exploitation balanced DE mutants, which helps to balance convergence and diversity. HeDE-PSO is more adaptive to different kinds of optimization problems than employing a single DE mutant learning exemplar.

Parameter analysis

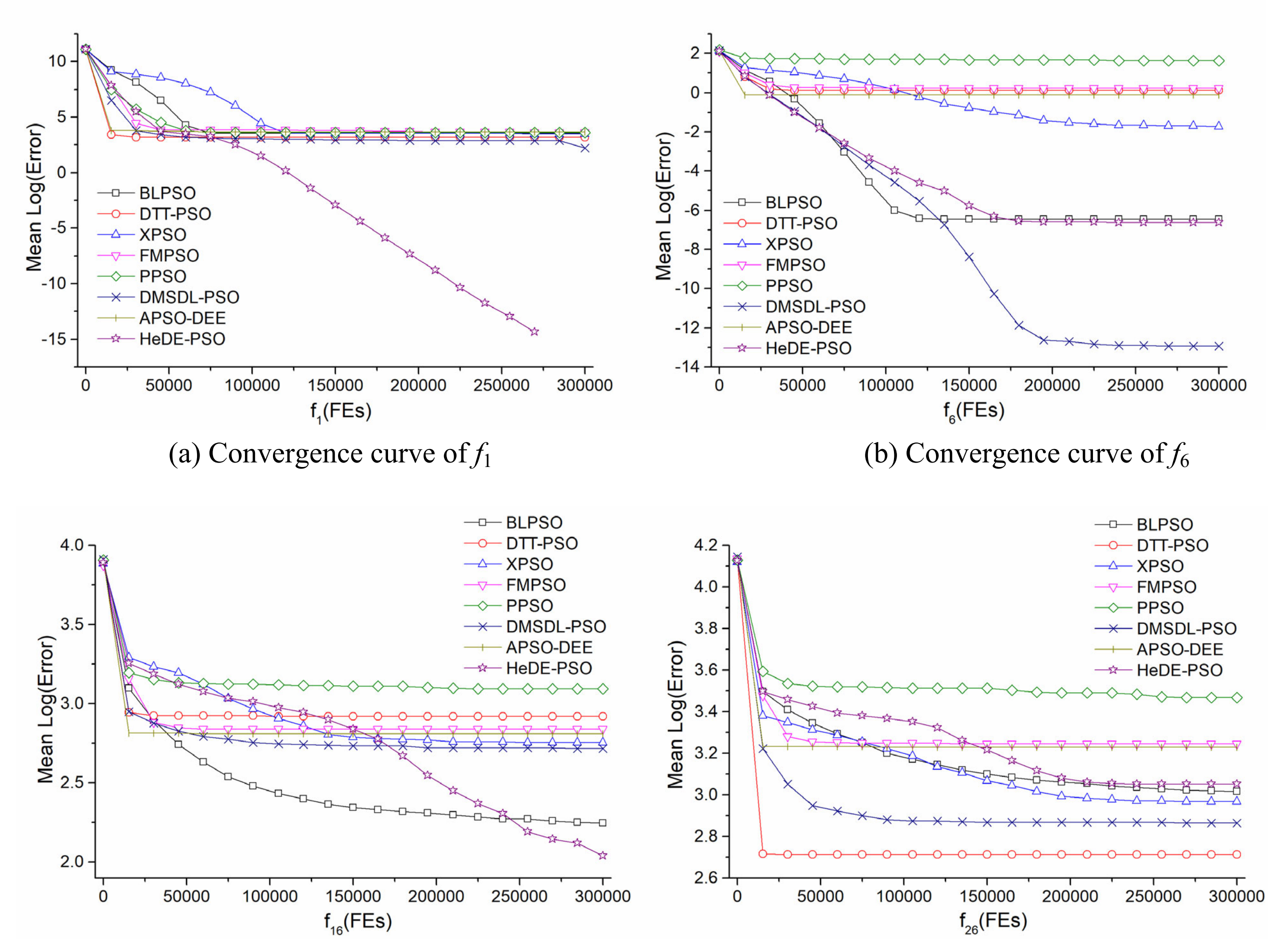
In this section, three parameters of HeDE-PSO are analyzed, namely, Ps_1, $p\%$ and FE$_{\mathrm{BFGS}}$ (function evolution of BFGS). In each test, only one parameter is tested for different values, while other parameters are set according to Table 1. f_1, f_7, f_{11},f_{17},f_{21} andf_{27} are employed as representative functions.

Ps_1 stands for the subswarm size of DE/current-to-pbest/1, and the remaining particles employ DE/rand/1 to construct learning exemplars. Increasing Ps_1 means enlarging the exploitation enhanced subswarm and reducing the exploration enhanced subswarm. Table 7 indicates that $Ps_1 = 15$ obtains the best performance of f_1,f_7,f_{11} and f_{21}; hence, $Ps_1 = 15$ is the best choice of HeDE-PSO.$p\%$ denotes that the top $p\%$ particles are selected for DE/current-to-*p*best/1 mutation in Eq. (11). Reducing $p\%$ means selecting a small portion of high-quality particles for DE/current-to-*p*best/1 mutation to enhance exploitation, while increasing$p\%$ means more elite particles can be employed for DE/current-to-*p*best/1 mutation to enhance exploration. Table 8 shows that $p\% = 20\%$ yields relatively better performance than other values; therefore $p\% = 20\%$ is adopted by HeDE-PSO.

FE$_{\mathrm{BFGS}}$ denotes the total function evolution consumed by BFGS local search in the later stage of HeDE-PSO. BFGS local search converges faster than population-based algorithms, while it is apt to fall into local optima. The initial point is very important for BFGS local search. Table 9 indicates that FE$_{\mathrm{BFGS}} = 100{*}D$ performs better than other parameter settings.

Application to industrial refrigeration system design

To test the performance of HeDE-PSO on real-life applications, the industrial refrigeration system design problem is used. The industrial refrigeration system design problem provided in CEC2020 is from a real-world single-objective



(a) Convergence curve of f_1

(b) Convergence curve of f_6

(c) Convergence curve of f_{16}

(d) Convergence curve of f_{26}

Fig. 4 The curves of mean Log(Error)

constrained optimization competition [68], and its mathematical model is expressed in Eqs. (19–34). The industrial refrigeration system design problem has fourteen variables, one objective and fifteen inequality constraints. The global optimum is 3.2213E−2 and the violation is 0. For fair comparison, ε-constraint [69, 70] is adopted as a constraint handling technique, and the BFGS local search [32] of DMSDL-PSO and HeDE-PSO is removed. The violation function Φ(x) can be expressed as

$$\phi(x) = \sum_j ||\max\{0,\ g_j(x)||^p + \sum_j ||\max\{0,\ h_j(x)||^p, \tag{15}$$

where $g_j(x) \le 0$ and $h_j(x) = 0$ denote inequality constraints and equality constraints, respectively. Let f_A, Φ_A, and f_B, Φ_B, denote for fitness values and constraint violation values at points A and B, respectively. ε level comparison is defined as below:

$$(f_A, \phi_A) <_\varepsilon (f_A, \phi_A) = \begin{cases} f_A < f_B, & If\,\phi_A, \phi_B \le \varepsilon \\ f_A < f_B, & If\,\phi_A = \phi_B \\ \phi_A < \phi_B\ otherwise & \end{cases}, \tag{16}$$

$$\varepsilon_0 = \phi(\boldsymbol{x}_\theta), \tag{17}$$

$$\varepsilon(t) = \begin{cases} \varepsilon(0)(1 - T_\lambda/T_c)^{cp}, & 0 < t \le T_c \\ 0 & t > T_c \end{cases}. \tag{18}$$

In this study, $\theta = 0.5$, $T_\lambda = 0.95 \times T_c$, $cp = 3$, $T_c = 0.2 \times M_{\text{FEs}}$. M_{FEs} is the maximum allowable function evolutions.

The test results of 30 independent runs are given in Table 10. *MF*, *MV*, *FR*, and *SR* denote mean fitness, mean constraint violation, feasibility rate and success rate, respectively. Success rate is the ratio that one algorithm obtains a feasible solution and the error is not bigger than 1e−8 ($f(\overline{x}) - f(\overline{x}^*) \le 10^{-8}$). Table 10 indicates that HeDE-PSO achieves 100% *FR* and 100% *SR*. BLPSO achieves 100% *FR*

Table 5 Modification of contrast algorithm

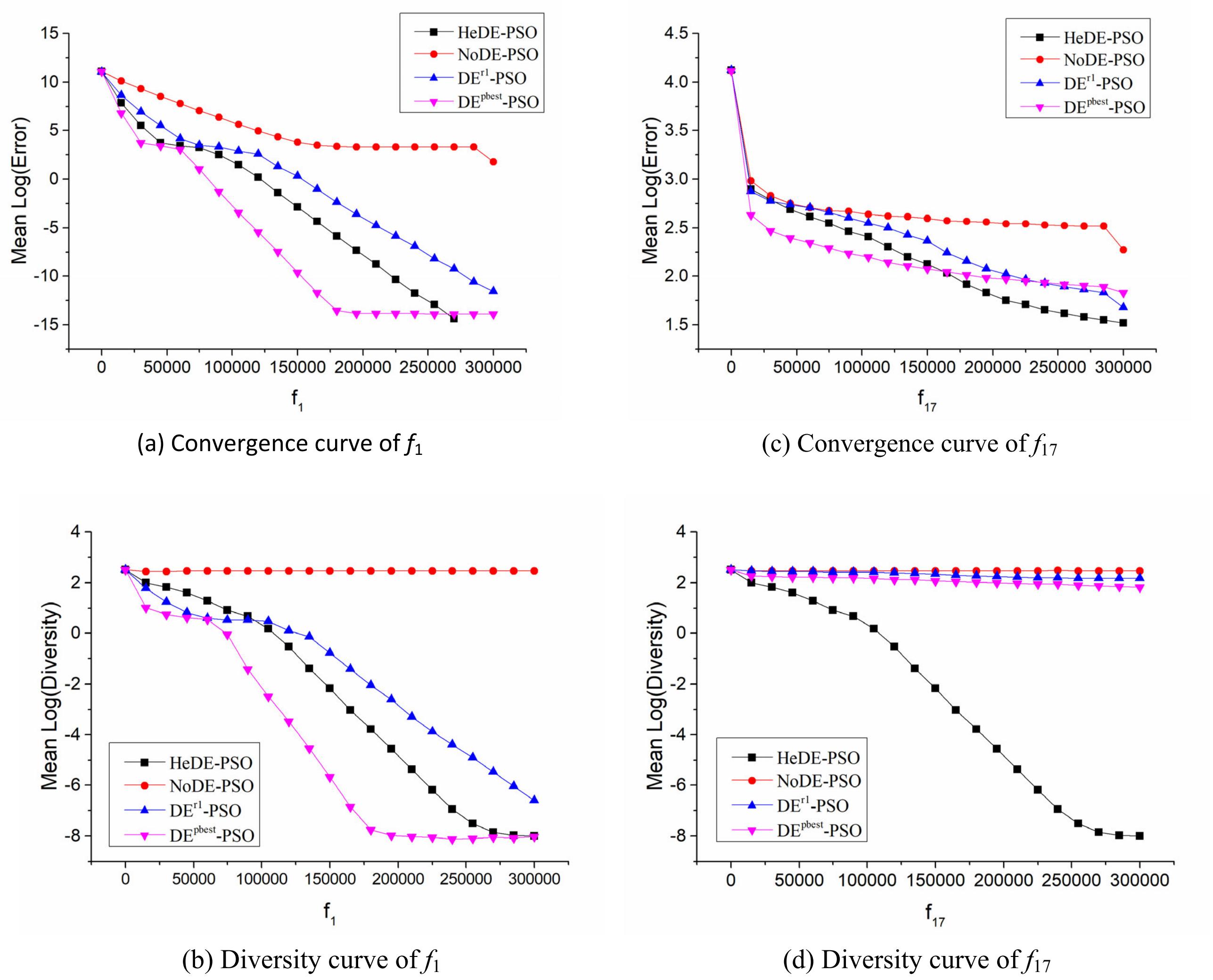
Contrast algorithm	Modification	Detail modification in "Methodologies"
NoLS-PSO	No BFGS local search	Removes step 9 (BFGS local search)
Noapy-PSO	No accompany operation	Executes PSO operation in step 5 for every particle without comparing fitness value of accompanying particle
NoDE-PSO	No DE mutation	Removes steps 2, 3, 4 (DE mutation)
DE^{r1}-PSO	DE/rand/1 mutation only	Generates the trial vector by DE/rand/1 mutation only in step 2
$\text{DE}^{p\text{best}}$-PSO	DE/current-to-*p*best/1 only	Generates the trial vector by DE/current-to-*p*best/1 mutation only in step 2

Table 6 Test results of contrast algorithms

Func	HeDE-PSO	NoLS-PSO	Noapy-PSO	NoDE-PSO	DE^{r1}-PSO	$\text{DE}^{p\text{best}}$-PSO
f_1	**0.00E + 00**	= 0.00E + 00	> 2.10E−11	> 6.22E + 01	> 1.37E−14	> 2.89E−12
f_3	4.32E−05	> 1.34E−01	< 2.87E−06	> 1.70E−01	**< 5.31E−14**	< 3.15E−06
f_4	1.09E + 00	> 4.84E + 01	= 1.10E + 00	= 2.79E + 00	= 1.86E + 00	**= 8.38E−01**
f_5	**2.36E + 01**	> 1.02E + 02	> 3.19E + 01	> 8.72E + 01	> 3.42E + 01	> 3.19E + 01
f_6	2.43E−07	< 4.56E−09	> 5.06E−09	> 1.26E + 00	> 2.85E−02	> **3.54E−11**
f_7	**5.08E + 01**	> 1.58E + 02	> 5.88E + 01	> 9.50E + 01	> 6.61E + 01	> 6.56E + 01
f_8	**2.59E + 01**	> 1.02E + 02	> 3.21E + 01	> 7.52E + 01	> 3.39E + 01	> 3.36E + 01
f_9	8.79E−02	< 1.81E−02	< 1.51E−02	> 1.60E + 02	> 2.10E + 00	**< 0.00E + 00**
f_{10}	**2.73E + 03**	> 6.23E + 03	> 3.19E + 03	> 3.32E + 03	> 3.11E + 03	> 3.28E + 03
f_{11}	**7.80E + 00**	> 1.74E + 01	> 1.34E + 01	> 6.02E + 01	> 5.77E + 01	> 1.90E + 01
f_{12}	1.57E + 03	> 1.28E + 04	= 1.76E + 03	> 3.14E + 03	= 1.67E + 03	**= 1.41E + 03**
f_{13}	**5.68E + 01**	> 6.94E + 01	= 5.98E + 01	> 2.17E + 03	= 6.10E + 01	= 6.06E + 01
f_{14}	**3.09E + 01**	> 4.18E + 01	> 4.41E + 01	> 1.18E + 03	> 4.81E + 01	> 5.49E + 01
f_{15}	**1.21E + 01**	> 2.15E + 01	> 2.52E + 01	> 8.14E + 02	> 3.98E + 01	> 2.74E + 01
f_{16}	**1.09E + 02**	> 2.56E + 02	> 3.76E + 02	> 6.91E + 02	> 2.84E + 02	> 2.47E + 02
f_{17}	**3.32E + 01**	> 6.10E + 01	> 4.64E + 01	> 1.88E + 02	> 6.77E + 01	> 4.78E + 01
f_{18}	**2.78E + 01**	> 3.27E + 01	= 3.05E + 01	> 3.10E + 03	> 3.49E + 01	> 3.11E + 01
f_{19}	**1.04E + 01**	= 1.24E + 01	> 1.84E + 01	> 1.68E + 03	> 2.68E + 01	> 1.64E + 01
f_{20}	**2.37E + 01**	= 3.27E + 01	> 3.37E + 01	> 3.81E + 02	> 5.80E + 01	= 2.83E + 01
f_{21}	**2.23E + 02**	> 2.96E + 02	> 2.30E + 02	> 2.60E + 02	> 2.29E + 02	> 2.29E + 02
f_{22}	**1.00E + 02**	> 7.09E + 02	> 1.00E + 02	> 3.01E + 02	> 1.83E + 02	> 4.50E + 02
f_{23}	**3.71E + 02**	> 4.31E + 02	> 3.77E + 02	> 4.57E + 02	> 3.82E + 02	> 3.81E + 02
f_{24}	**4.39E + 02**	> 4.96E + 02	> 4.47E + 02	> 4.91E + 02	> 4.52E + 02	> 4.49E + 02
f_{25}	**3.80E + 02**	> 3.87E + 02	**= 3.80E + 02**	< 3.81E + 02	= 3.81E + 02	= **3.80E + 02**
f_{26}	1.13E + 03	> 1.60E + 03	> 1.22E + 03	**< 1.02E + 03**	> 1.33E + 03	> 1.23E + 03
f_{27}	**4.92E + 02**	> 4.99E + 02	= 4.95E + 02	= 5.07E + 02	> 5.02E + 02	= **4.92E + 02**
f_{28}	3.37E + 02	> 3.26E + 02	**> 3.20E + 02**	> 3.21E + 02	> 3.47E + 02	> 3.30E + 02
f_{29}	**4.39E + 02**	> 5.46E + 02	> 5.50E + 02	> 8.37E + 02	> 5.07E + 02	> 5.07E + 02
f_{30}	2.05E + 03	= 2.04E + 03	**= 2.04E + 03**	> 1.43E + 04	> 2.19E + 03	= 2.05E + 03
Best	21	0	3	1	1	6
w/t/l	/	23/4/2	20/7/2	25/2/2	24/4/1	21/7/1

The best test results are highlighted in bold

while *SR* is 0%. APSO-DEE, XPSO, FMPSO and DMSDL-PSO yield high *FR*, while their *SR* are not high. The SR of DTT-PSO and XPSO are 3.33% and 16.7%, respectively. The SR of BLPSO, FMPSO and PPSO are 0%. HeDE-PSO has significant advantages over other PSO algorithms.



(a) Convergence curve of f_1

(c) Convergence curve of f_{17}

(b) Diversity curve of f_1

(d) Diversity curve of f_{17}

Fig. 5 Convergence and diversity curves of the proposed strategies

Table 7 The effects of Ps_1

Func	5	10	15	20	25	30
f_1	5.21E−15	4.74E−16	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
f_7	5.96E + 01	5.62E + 01	**5.08E + 01**	5.38E + 01	5.79E + 01	6.18E + 01
f_{11}	1.34E + 01	1.17E + 01	**8.77E + 00**	1.11E + 01	1.16E + 01	1.17E + 01
f_{17}	4.51E + 01	4.66E + 01	4.09E + 01	3.99E + 01	**3.72E + 01**	3.83E + 01
f_{21}	**2.23E + 02**	2.28E + 02	**2.23E + 02**	2.26E + 02	2.29E + 02	2.27E + 02
f_{27}	4.94E + 02	4.94E + 02	4.94E + 02	**4.93E + 02**	**4.93E + 02**	4.95E + 02

The best test results are highlighted in bold

Minimize:

$$\begin{aligned} f(\overline{x}) = {} & 63098.88x_2x_4x_{12} + 5441.5x_2^2x_{12} + 115055.5x_2^{1.664}x_6 \\ & + 6172.27x_2^2x_6 + 63098.88x_1x_3x_{11} \\ & + 5441.5x_1^2x_{11} + 115055.5x_1^{1.664}x_5 \\ & + 6172.27x_1^2x_5 + 140.53x_1x_{11} + 281.29x_3x_{11} \\ & + 70.26x_1^2 + 281.29x_1x_3 + 281.29x_3^2 \\ & + 14437x_8^{1.8812}x_{12}^{0.3424}x_{10}x_{14}^{-1}x_1^2x_7x_9^{-1}, \end{aligned} \tag{19}$$

subject to

$$g_1(\overline{x}) = 1.524x_7^{-1} \leq 1, \tag{20}$$

Table 8 The effects of $p\%$

Func	5%	10%	15%	20%	25%	30%
f_1	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
f_7	5.84E + 01	5.46E + 01	5.42E + 01	**5.08E + 01**	5.52E + 01	5.41E + 01
f_{11}	1.00E + 01	9.70E + 00	9.42E + 00	**8.77E + 00**	1.10E + 01	1.08E + 01
f_{17}	4.24E + 01	4.83E + 01	4.57E + 01	4.09E + 01	4.03E + 01	**3.98E + 01**
f_{21}	2.29E + 02	**2.23E + 02**	2.24E + 02	**2.23E + 02**	2.25E + 02	2.25E + 02
f_{27}	4.96E + 02	4.92E + 02	4.98E + 02	4.94E + 02	**4.93E + 02**	4.95E + 02

The best test results are highlighted in bold

Table 9 The effects of $\mathrm{FE}_{\mathrm{BFGS}}$

Func	25*D	50*D	75*D	100*D	125*D	150*D
f_1	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
f_7	5.39E + 01	5.36E + 01	5.48E + 01	**5.08E + 01**	5.67E + 01	5.58E + 01
f_{11}	1.73E + 01	1.07E + 01	1.41E + 01	**8.77E + 00**	1.07E + 01	9.00E + 00
f_{17}	4.22E + 01	4.28E + 01	**4.03E + 01**	4.09E + 01	4.05E + 01	4.53E + 01
f_{21}	2.24E + 02	2.23E + 02	2.26E + 02	**2.23E + 02**	2.23E + 02	2.24E + 02
f_{27}	**4.91E + 02**	4.99E + 02	4.95E + 02	4.94E + 02	4.96E + 02	4.92E + 02

The best test results are highlighted in bold

Table 10 Test results on industrial refrigeration system design

Statistics	BLPSO	DTT-PSO	XPSO	FMPSO	PPSO	DMSDL-PSO	APSO-DBEE	HeDE-PSO
MF	3.753E−02	6.907E−03	4.360E−01	8.646E−01	7.485E + 01	6.713E + 02	8.894E−2	3.221E−02
MV	0.000E + 00	1.119E + 03	1.745E + 01	2.583E−01	1.661E + 01	1.615E−01	1.314E−1	0.000E + 00
FR	100.00%	10.00%	70.00%	73.33%	53.33%	83.33%	86.67%	100.00%
SR	0.00%	3.33%	16.67%	0.00%	0.00%	0.00%	0.00%	100.00%

$$g_2(\overline{x}) = 1.524x_8^{-1} \leq 1, \tag{21}$$

$$g_3(\overline{x}) = 0.07789x_1 - 2x_7^{-1}x_9 - 1 \leq 0, \tag{22}$$

$$g_4(\overline{x}) = 7.05305x_9^{-1}x_1^2x_{10}x_8^{-1}x_2^{-1}x_{14}^{-1} - 1 \leq 0, \tag{23}$$

$$g_5(\overline{x}) = 0.0833x_{13}^{-1}x_{14} - 1 \leq 0, \tag{24}$$

$$\begin{aligned} g_6(\overline{x}) = {} & 47.136x_2^{0.333}x_{10}^{-1}x_{12} - 1.333x_8x_{13}^{2.1195} \\ & + 62.08x_{13}^{2.1195}x_{12}^{-1}x_8^{0.2}x_{10}^{-1} - 1 \leq 0, \end{aligned} \tag{25}$$

$$g_7(\overline{x}) = 0.04471x_{10}x_8^{1.8812}x_{12}^{0.3424} - 1 \leq 0, \tag{26}$$

$$g_8(\overline{x}) = 0.0488x_9x_7^{1.893}x_{11}^{0.316} - 1 \leq 0, \tag{27}$$

$$g_9(\overline{x}) = 0.0099x_1x_3^{-1} - 1 \leq 0, \tag{28}$$

$$g_{10}(\overline{x}) = 0.0193x_2x_4^{-1} - 1 \leq 0, \tag{29}$$

$$g_{11}(\overline{x}) = 0.0298x_1x_5^{-1} - 1 \leq 0, \tag{30}$$

$$g_{12}(\overline{x}) = 0.056x_2x_6^{-1} - 1 \leq 0, \tag{31}$$

$$g_{13}(\overline{x}) = 2x_9^{-1} - 1 \leq 0, \tag{32}$$

$$g_{14}(\overline{x}) = 2x_{10}^{-1} - 1 \leq 0, \tag{33}$$

$$g_{15}(\overline{x}) = x_{12}x_{11}^{-1} - 1 \leq 0. \tag{34}$$

With bounds

$$0.001 \leq x_i \leq 5, \quad I = 1, 2, 3, \ldots, 4.$$

Conclusions

This study proposes a heterogeneous differential evolution particle swarm optimization (HeDE-PSO) method. HeDE-PSO adopts two DE mutants to construct learning exemplars for PSO to improve adaptability and employs BFGS local search to increase search accuracy in the late stage. DE/rand/1 is employed for enhancing exploration and DE/current-to-*p*Best/1 is employed for enhancing exploitation. The test results on 10-dimensional and 30-dimensional functions show that HeDE-PSO defeats the comparison algorithms on most of the tested functions. For 30-dimensional functions, HeDE-PSO outperforms other PSO algorithms on at least 20 functions out of 29 functions. HeDE-PSO obtains high performance on 10-dimensional and 30-dimensional CEC2017 functions without parameter turning. On the industrial refrigeration system design problem, HeDE-PSO is the sole algorithm generating 100% feasibility rate and 100% success rate. The test results indicate that adopting heterogeneous DE learning exemplars can improve the performance and adaptability of HeDE-PSO. HeDE-PSO is applicable to both benchmark functions and real-life application problems.

Acknowledgements The authors thank the reviewers for their constructive comments. This work is supported by the Scientific Research Fund of Hunan Provincial Education Department under Grant 20A460, the Applied Characteristic Disciplines of Electronic Science and Technology of Xiangnan University (XNXY20221210), the Scientific Research Start-up Fund for High-level Talents in Xiangnan University, Chenzhou Low Carbon Intelligent Manufacturing Technology Research and Development Center, Hunan Provincial Innovation and Entrepreneurship Training Program for College Students (2020-3667).

Data availability All data generated or analyzed during this study are presented in this published article.

Declarations

Conflict of interest The authors declare that there is no competing financial interests or personal relationships influence the work reported in this paper.

References

1. Holland JH (1973) Erratum: genetic algorithms and the optimal allocation of trials. SIAM J Comput 2:88–105
2. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the 6th international symposium on micro machine and human science, pp 39–43
3. Ratnaweera A, Halgamuge SK, Watson HC (2004) Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans Evol Comput 8:240–255
4. Liu W, Wang Z, Yuan Y, Zeng N, Hone K, Liu X (2019) A novel sigmoid-function-based adaptive weighted particle swarm optimizer. IEEE Trans Cybern 51:1085–1093
5. Shi Y, Eberhart RC (2001) Fuzzy adaptive particle swarm optimization. In: IEEE congress on evolutionary computation, pp 101–106
6. Xia X, Song H, Zhang Y, Gui L, Xu X, Li K et al (2022) A particle swarm optimization with adaptive learning weights tuned by a multiple-input multiple-output fuzzy logic controller. IEEE Trans Fuzzy Syst, pp 1–15
7. Alatas B, Akin E, Ozer AB (2009) Chaos embedded particle swarm optimization algorithms. Chaos Solitons Fractals 40:1715–1734
8. Zhan Z-H, Zhang J, Li Y, Chung HS-H (2009) Adaptive particle swarm optimization. IEEE Trans Syst Man Cybern Part B Cybern 39:1362–1381
9. Xia X, Gui L, He G, Wei B, Zhang Y, Yu F et al (2020) An expanded particle swarm optimization based on multi-exemplar and forgetting ability. Inf Sci 508:105–120
10. Mendes R, Kennedy J, Neves J (2004) The fully informed particle swarm: simpler, maybe better. IEEE Trans Evol Comput 8:204–210
11. Liang JJ, Suganthan PN (2005) Dynamic multi-swarm particle swarm optimizer. IEEE Swarm Intell Symp 2005:124–129
12. Li T, Shi J, Deng W, Hu Z (2022) Pyramid particle swarm optimization with novel strategies of competition and cooperation. Appl Soft Comput 121:108731
13. Lim WH, Isa NAM (2014) Particle swarm optimization with increasing topology connectivity. Eng Appl Artif Intell 27:80–102
14. Wang L, Yang B, Orchard J (2016) Particle swarm optimization using dynamic tournament topology. Appl Soft Comput 48:584–596
15. Lin A, Li S, Liu R (2022) Mutual learning differential particle swarm optimization. Egypt Inform J 23:469–481
16. Javidrad F, Nazari M (2017) A new hybrid particle swarm and simulated annealing stochastic optimization method. Appl Soft Comput 60:634–654
17. Gong YJ, Li JJ, Zhou Y, Li Y, Chung HSH, Shi YH et al (2016) Genetic learning particle swarm optimization. IEEE Trans Cybern 46:2277–2290
18. Adamu A, Abdullahi M, Junaidu SB, Hassan IH (2021) An hybrid particle swarm optimization with crow search algorithm for feature selection. Mach Learn Appl 6:100108
19. Yang R, Liu Y, Yu Y, He X, Li H (2021) Hybrid improved particle swarm optimization-cuckoo search optimized fuzzy PID controller for micro gas turbine. Energy Rep 7:5446–5454
20. Gümüşçü A, Kaya S, Tenekeci ME, Karaçizmeli İH, Aydilek İB (2021) The impact of local search strategies on chaotic hybrid firefly particle swarm optimization algorithm in flow-shop scheduling. J King Saud Univ Comput Inf Sci 34:6432–6440
21. Bashath S, Ismail AR (2019) Improved particle swarm optimization by fast simulated annealing algorithm. In: 2019 international conference of artificial intelligence and information technology (ICAIIT), pp 297–301
22. Zhen L, Liu Y, Dongsheng W, Wei Z (2020) Parameter estimation of software reliability model and prediction based on hybrid wolf pack algorithm and particle swarm optimization. IEEE Access 8:29354–29369

23. Liang JJ, Qin AK, Member S, Suganthan PN, Member S, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 10:281–295
24. Qin QD, Cheng S, Zhang QY, Li L, Shi YH (2016) Particle swarm optimization with inter swarm interactive learning strategy. IEEE Trans Cybern 46:2238–2251
25. Cheng R, Jin Y (2015) A social learning particle swarm optimization algorithm for scalable optimization. Inf Sci 291:43–60
26. Xu G, Cui Q, Shi X, Ge H, Zhan Z-H, Lee HP et al (2019) Particle swarm optimization based on dimensional learning strategy. Swarm Evol Comput 45:33–51
27. Wang F, Zhang H, Li KS, Lin ZY, Yang J, Shen XL (2018) A hybrid particle swarm optimization algorithm using adaptive learning strategy. Inf Sci 436:162–177
28. Kaucic M, Piccotto F (2022) A level-based learning swarm optimizer with a hybrid constraint-handling technique for large-scale portfolio selection problems. In: 2022 IEEE congress on evolutionary computation (CEC), pp 1–8
29. Li D, Guo W, Lerch A, Li Y, Wang L, Wu Q (2021) An adaptive particle swarm optimizer with decoupled exploration and exploitation for large scale optimization. Swarm Evol Comput 60:100789
30. Sheng M, Wang Z, Liu W, Wang X, Chen S, Liu X (2022) A particle swarm optimizer with multi-level population sampling and dynamic p-learning mechanisms for large-scale optimization. Knowl Based Syst 242:108382
31. Liang JJ, Suganthan PN (2005) Dynamic multi-swarm particle swarm optimizer with local search. In: IEEE congress on evolutionary computation, pp 522–528
32. Wu G, Qiu D, Yu Y, Pedrycz W, Ma M, Li H (2014) Superior solution guided particle swarm optimization combined with local search techniques. Expert Syst Appl 41:7536–7548
33. Chen Y, Li L, Peng H, Xiao J, Wu QT (2017) Dynamic multi-swarm differential learning particle swarm optimizer. Swarm Evol Comput 39:209–221
34. Hu M, Wu T, Weir JD (2013) An adaptive particle swarm optimization with multiple adaptive methods. IEEE Trans Evol Comput 17:705–720
35. Cao Y, Zhang H, Li W, Zhou M, Zhang Y, Chaovalitwongse WA (2019) Comprehensive learning particle swarm optimization algorithm with local search for multimodal functions. IEEE Trans Evol Comput 23:718–731
36. Yu F, Tong L, Xia X (2022) Adjustable driving force based particle swarm optimization algorithm. Inf Sci 609:60–78
37. Chen Y, Li L, Peng H, Xiao J, Yang Y, Shi Y (2017) Particle swarm optimizer with two differential mutation. Appl Soft Comput 61:314–330
38. Lynn N, Suganthan PN (2017) Ensemble particle swarm optimizer. Appl Soft Comput 55:533–548
39. Wang H, Liang M, Sun C, Zhang G, Xie L (2021) Multiple-strategy learning particle swarm optimization for large-scale optimization problems. Complex Intell Syst 7:1–16
40. Ning Y, Peng Z, Dai Y, Bi D, Wang J (2018) Enhanced particle swarm optimization with multi-swarm and multi-velocity for optimizing high-dimensional problems. Appl Intell 49:335–351
41. Lynn N, Suganthan PN (2015) Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation. Swarm Evol Comput 24:11–24
42. Shi YH, Eberhart R (1998) A modified particle swarm optimizer. IEEE, New York
43. Lin A, Sun W, Yu H, Wu G, Tang H (2018) Global genetic learning particle swarm optimization with diversity enhancement by ring topology. Swarm Evol Comput 44:571–583
44. Chen X, Tianfield H, Mei CL, Du WL, Liu GH (2017) Biogeography-based learning particle swarm optimization. Soft Comput 21:7519–7541
45. Lu L, Zheng H, Jie J, Zhang M, Dai R (2021) Reinforcement learning-based particle swarm optimization for sewage treatment control. Complex Intell Syst 7:2199–2210
46. Zhang X, Sun W, Xue M, Lin A (2021) Probability-optimal leader comprehensive learning particle swarm optimization with Bayesian iteration. Appl Soft Comput 103:107132
47. Zhang D, Ma G, Deng Z, Wang Q, Zhang G, Zhou W (2022) A self-adaptive gradient-based particle swarm optimization algorithm with dynamic population topology. Appl Soft Comput 130:109660
48. Cheng S, Lu H, Lei X, Shi Y (2018) A quarter century of particle swarm optimization. Complex Intell Syst 4:227–239
49. Das S, Mullick SS, Suganthan PN (2016) Recent advances in differential evolution—an updated survey. Swarm Evol Comput 27:1–30
50. Zhang JQ, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13:945–958
51. Wu G, Shen X, Li H, Chen H, Lin A (2017) Ensemble of differential evolution variants. Inf Sci 423:172–186
52. Ahmad MF, Isa NAM, Lim WH, Ang KM (2022) Differential evolution: a recent review based on state-of-the-art works. Alex Eng J 61:3831–3872
53. Melton RG (2018) Differential evolution/particle swarm optimizer for constrained slew maneuvers. Acta Astronaut 148:246–259
54. Song M, Chen K, Wang J (2018) Three-dimensional wind turbine positioning using Gaussian particle swarm optimization with differential evolution. J Wind Eng Ind Aerodyn 172:317–324
55. Wang S, Li Y, Yang H (2019) Self-adaptive mutation differential evolution algorithm based on particle swarm optimization. Appl Soft Comput 81:105496
56. Mohammadi B, Guan Y, Moazenzadeh R, Safari MJS (2021) Implementation of hybrid particle swarm optimization-differential evolution algorithms coupled with multi-layer perceptron for suspended sediment load estimation. CATENA 198:105024
57. Moharam A, El-Hosseini MA, Ali HA (2016) Design of optimal PID controller using hybrid differential evolution and particle swarm optimization with an aging leader and challengers. Appl Soft Comput 38:727–737
58. Deng L, Lu G, Shao Y, Fei M, Hu H (2016) A novel camera calibration technique based on differential evolution particle swarm optimization algorithm. Neurocomputing 174:456–465
59. Wang C, Xu M, Zhang Q, Feng J, Jiang R, Wei Y et al (2021) Parameters identification of Thevenin model for lithium-ion batteries using self-adaptive particle swarm optimization differential evolution algorithm to estimate state of charge. J Energy Storage 44:103244
60. Xia XW, Xing Y, Wei B, Zhang YL, Li X, Deng XL et al (2019) A fitness-based multi-role particle swarm optimization. Swarm Evol Comput 44:349–364
61. Ghasemi M, Akbari E, Rahimnejad A, Razavi SE, Ghavidel S, Li L (2019) Phasor particle swarm optimization: a simple and efficient variant of PSO. Soft Comput 23:9701–9718
62. Hashim FA, Houssein EH, Hussain K, Mabrouk MS, Al-Atabany W (2022) Honey badger algorithm: new metaheuristic algorithm for solving optimization problems. Math Comput Simul 192:84–1104
63. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191
64. Hashim FA, Hussain K, Houssein EH, Mabrouk MS, Al-Atabany W (2021) Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems. Appl Intell 51:1531–1551
65. Agushaka JO, Ezugwu AE, Abualigah L (2022) Dwarf mongoose optimization algorithm. Comput Methods Appl Mech Eng 391:114570

66. Awad NH, Ali MZ, Liang JJ, Qu BY, Suganthan PN (2016) Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective bound constrained real-parameter numerical optimization. http://www.ntu.edu.sg/home/epnsugan/
67. 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 1:3–18
68. Kumar A, Wu G, Ali MZ, Mallipeddi R, Suganthan PN, Das S (2020) A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. Swarm Evol Comput 56:100693
69. Takahama T, Sakai S (2006) Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites. In: 2006 IEEE international conference on evolutionary computation, pp 1–8
70. Wu G, Wen X, Wang L, Pedrycz W, Suganthan PN (2021) A voting-mechanism based ensemble framework for constraint handling techniques. IEEE Trans Evol Comput 26:646–660