The Journal of Supercomputing (2025) 81:696
https://doi.org/10.1007/s11227-024-06862-8

HHDE: a hyper-heuristic differential evolution with novel boundary repair technique for complex optimization

Rui Zhong[1] · Shilong Zhang[2] · Jun Yu[3] · Masaharu Munetomo[1]

Accepted: 19 December 2024

Abstract
Inspired by the architecture of the hyper-heuristic algorithm, we design a mutation operator archive, a crossover operator archive, and a boundary repair operator archive to propose a novel hyper-heuristic differential evolution (HHDE). The mutation operator and crossover operator archives contain multiple representative search operators derived from variants of DE. A learning-free probabilistic selection function serves as the high-level component of the HHDE and is employed to determine the optimization sequence during optimization automatically. Additionally, we focus on the boundary repair operator, an element often overlooked in the design of the evolutionary algorithm (EA). Based on the previous research, our designed boundary repair operator archive introduces two novel boundary repair techniques: optimum inheritance and iterative opposite-based mapping. Comprehensive numerical experiments on CEC2017, CEC2020, and CEC2022 benchmark functions are conducted to evaluate the performance of our proposed HHDE. A range of other state-of-the-art optimizers and advanced variants of DE are employed as competitor algorithms. The experimental results and statistical analysis confirm the competitiveness and efficiency of HHDE. The source code of HHDE can be found in https://github.com/RuiZhong961230/HHDE.

Keywords Hyper-heuristic (HH) · Differential evolution (DE) · Boundary repair technique · Complex optimization

1 Introduction

Since the conventional differential evolution (DE) algorithm was proposed by Storn and Price in 1996 [1], it has emerged as one of the most popular and efficient stochastic optimization techniques in the evolutionary computation (EC) community [2–4]. Thanks to its superior characteristics such as scalability, applicability, robustness, and easy implementation, DE-based optimization techniques have been widely applied in

Extended author information available on the last page of the article various domains, such as multi-objective optimization [5, 6], large-scale global optimization [7, 8], real-world engineering challenges [9–11], drug design [12, 13], and other applications [14, 15]. In the meantime, many remarkable variants of DE have been proposed, including self-adaptive DE (SaDE) [16], JADE [17], success-history-based parameter adaptation DE (SHADE) [18], and SHADE using linear population size reduction (L-SHADE) [19]. The rich history of DE will not be fully reviewed here, as it can be referred to [20–22].

The conventional DE employs a series of operations including mutation, crossover, and selection to search for improved solutions iteratively. These expert-designed search operators collectively form the optimization sequence. This raises an interesting question: Is it possible to optimize the optimization sequence itself? The response is in the affirmative, aligning with the concept of the hyper-heuristic (HH) framework [23]. In contrast to tailored meta-heuristic algorithms (MAs), which are specifically designed for particular problems, the HH algorithm represents a more generic, “off-the-peg” technique as opposed to “made-to-measure” [24]. As a high-level automatic methodology, the HH algorithm consists of two crucial components: the low-level and the high-level components. The low-level component represents the inherent attributes of the algorithm, whereas the high-level component functions as the intelligent decision maker for determining the optimization sequence. While many HH algorithms have been effectively applied to combinatorial optimization problems [25, 26], only a few have been focused on continuous problems [27, 28].

Furthermore, a critical yet frequently overlooked component in the design of the evolutionary algorithm (EA) is the boundary repair technique. In constructing the offspring through search operators, there is a probability that the resultant offspring individual may fall outside the feasible search domain. The traditional method to address this issue involves manually setting the value to the search boundary or randomly generating a value within the search domain. However, these simplistic methods often fail to make adequate use of the domain knowledge.

In this paper, we address the mentioned challenges by integrating the HH framework with DE, leading to the development of a novel hyper-heuristic differential evolution (HHDE) algorithm. HHDE consists of a mutation operator archive, a crossover operator archive, and a boundary repair strategy archive. As a primary approach, the learning-free stochastic function is employed as the high-level component of HHDE. Moreover, the comprehensive numerical experiments conducted using CEC2017, CEC2020, and CEC2022 benchmark functions have been compared against nine state-of-the-art competitor EAs. These rigorous comparisons further underscore the effectiveness and superior performance of our proposed HHDE.

The remaining paper is organized as follows. Section 2 introduces the related works, Sect. 3 introduces our proposed HHDE in detail, Sect. 4 describes experimental settings and results, we analyze the performance of HHDE in Sect. 5, and finally, Sect. 6 concludes our work.

2 Related works

2.1 Differential evolution (DE)

In this section, we briefly introduce the structure of the basic DE algorithm. Without the loss of generality, the minimization problem is considered in this paper, and the objective is to find an optimum, denoted as $\boldsymbol{x}^*$, that satisfies the following Eq. (1) [29, 30].

$$f(\boldsymbol{x}^*) = \min_{\vec{x}\in\Omega}(f(\vec{x})) \tag{1}$$

where $f(\cdot)$ is the objective function, $\vec{x} = \{x_1, x_2, ..., x_n\}$ is an n-dimensional trial solution, and Ω denotes the search domain.

As one of the population-based EAs, the first step of DE is to initialize the population through Eq. (2).

$$\begin{aligned} \vec{x}_{i,j} =& \mathrm{LB}_j + r \cdot (\mathrm{UB}_j - \mathrm{LB}_j) \\ X =& \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_{\mathrm{NP}}\} \end{aligned} \tag{2}$$

where UB_j and LB_j are the upper and lower bound of the *jth* dimension, respectively, r is a random number within the range (0, 1), NP is the population size, and X represents the population. Considering the simplest DE/rand/1 mutation strategy, the mutated individual v can be constructed using Eq. (3).

$$\vec{v}_i = \vec{x}_{r_1} + F \cdot (\vec{x}_{r_2} - \vec{x}_{r_3}) \tag{3}$$

where F is a scaling factor, $\vec{x}_{r_1}$, $\vec{x}_{r_2}$ and $\vec{x}_{r_3}$ are mutually different individuals which are randomly selected from the population X. Then, the binomial crossover between the trial vector v_i and parent individual x_i is formulated in Eq. (4).

$$\vec{u}_{i,j} = \begin{cases} \vec{v}_{i,j}, & \mathit{if}\ r < Cr\ \mathit{or}\ j = jrand \\ \vec{x}_{i,j}, & \text{otherwise} \end{cases} \tag{4}$$

where Cr denotes the crossover rate and *jrand* is a randomly selected dimension. Finally, the greedy selection strategy, which ensures the survival of better solutions, is expressed in Eq. (5).

$$\vec{x}_i = \begin{cases} \vec{u}_i, & \mathit{if}\ f(\vec{u}_i) < f(\vec{x}_i) \\ \vec{x}_i, & \text{otherwise} \end{cases} \tag{5}$$

The DE algorithm iteratively repeats processes involving the mutation operator, the crossover operator, and the selection, until the optimum is found or the computational budget is exhausted.

2.2 Hyper-heuristic (HH) framework

The concept of the HH algorithm can be traced to 1960 [31]. Represented as an advanced methodology, the HH algorithm focuses on optimizing the sequence of search operators based on existing knowledge-an approach often described as “heuristics to choose heuristics” [32, 33]. A representative architecture of the HH algorithm is illustrated in Fig. 1.

A general HH algorithm is composed of two key components: the low-level component and the high-level component. The low-level component contains the problem representation, the objective function(s), initial solutions, and a set of low-level heuristics (LLHs). These elements collectively consist of the intrinsic attributions of the HH algorithm. The high-level component is responsible for managing the LLHs and constructing the sequence of heuristics. It also employs a move acceptance principle to determine whether the generated offspring individual should be accepted or rejected. Additionally, feedback is harnessed as a form of reward to dynamically fine-tune the selection module of the LLHs, enhancing the overall efficiency and adaptability of the algorithm.

3 Our proposal: HHDE

This section introduces our proposed HHDE in detail. We begin by presenting an overview of HHDE, as illustrated in Fig. 2.

HHDE integrates the basic mutation-crossover-boundary repair-selection skeleton of the conventional DE, while the specific operator for each step is selected

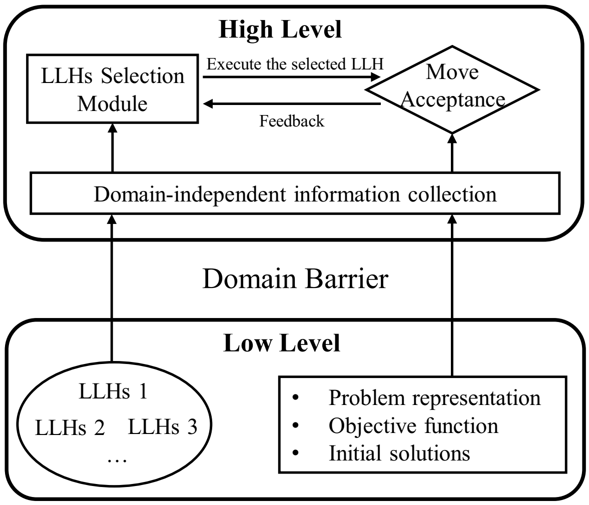

Fig. 1 The representative architecture of the HH algorithm [34]

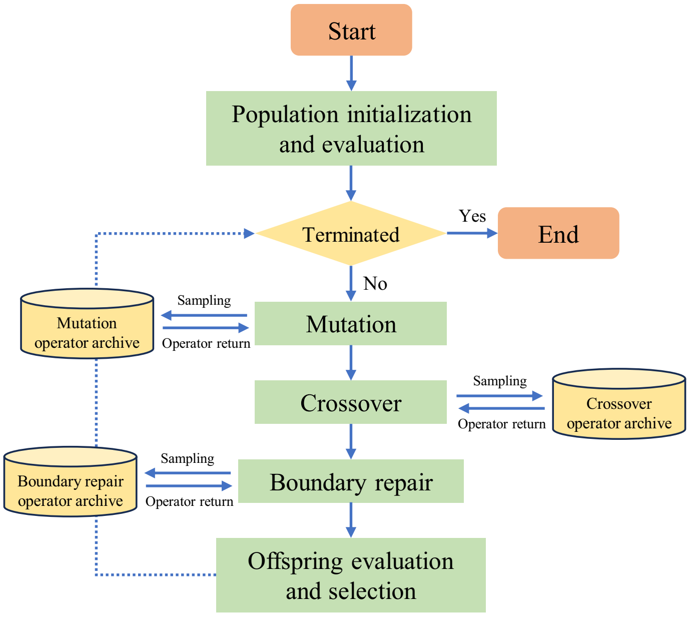
Start
Population initialization and evaluation
Terminated
Yes
End
No
Mutation
Sampling
Operator return
Mutation operator archive
Crossover
Sampling
Operator return
Crossover operator archive
Boundary repair
Sampling
Operator return
Boundary repair operator archive
Offspring evaluation and selection

Fig. 2 The flowchart of HHDE

from the designated archive. Moreover, we adopt the asynchronous update strategy in HHDE, which denotes that the search process for the next single individual only commences once the mutation-crossover-boundary repair-selection procedure for the preceding individual is completed. The benefit of this strategy is that, if a better solution is found, this enhanced information can be immediately used in the construction of the offspring for the subsequent individual, which can accelerate the optimization convergence. Next, we will introduce the mutation operator archive, the crossover operator archive, and the novel boundary repair operator archive.

3.1 Mutation operator archive

The designed mutation operator archive contains five featured operators from different DE versions [35], which are presented in Eq. (6).

$$
\begin{aligned}
&rand/1: \vec{v}_i = \vec{x}_{r_1} + F \cdot (\vec{x}_{r_2} - \vec{x}_{r_3})\\
&best/1: \vec{v}_i = \vec{x}_{\text{best}} + F \cdot (\vec{x}_{r_2} - \vec{x}_{r_3})\\
&cur/1: \vec{v}_i = \vec{x}_i + F \cdot (\vec{x}_{r_2} - \vec{x}_{r_3})\\
&cur2best/1: \vec{v}_i = \vec{x}_i + F \cdot (\vec{x}_{\text{best}} - \vec{x}_i) + F \cdot (\vec{x}_{r_2} - \vec{x}_{r_3})\\
&cur2pbest/1: \vec{v}_i = \vec{x}_i + F \cdot (\vec{x}_{\text{pbest}} - \vec{x}_i) + F \cdot (\vec{x}_{r_2} - \vec{x}_{r_3})
\end{aligned}
\tag{6}
$$

Here, $\vec{x}_{\text{best}}$ denotes the optimum found so far, and $\vec{x}_{\text{pbest}}$ indicates the mean of the top-5% solutions as suggested in [17]. To minimize the number of the hyper-parameter, we set the scaling factor F following a normal distribution $N(0.5, 0.3)$ as recommended in [16].

3.2 Crossover operator archive

The crossover operator archive is composed of five crossover operators: binomial crossover with the parent individual and $\vec{x}_{\text{best}}$, exponential crossover with the parent individual and $\vec{x}_{\text{best}}$, and blending crossover.

The binomial crossover with the parent individual can be found in Eq. (4), and we simply replace the $\vec{x}_{i,j}$ to $\vec{x}_{\text{best,j}}$ to realize the binomial crossover with $\vec{x}_{\text{best}}$. A visual demonstration of the exponential crossover with the parent individual is provided in Fig. 3.

First, a random integer generated from $[1, n]$ determines the starting point of the crossover. Then, if a dimension-varying random number $rand()$ is smaller than the crossover rate Cr, the offspring individual replicates the corresponding value from the mutated individual $\vec{v}_i$; otherwise, it copies the corresponding value from the parent individual $\vec{x}_i$ and the exponential crossover terminated. Similarly, the parent individual $\vec{x}_i$ can be replaced by the current best solution $\vec{x}_{\text{best}}$ to realize the exponential crossover with $\vec{x}_{\text{best}}$.

Finally, our designed crossover archive absorbs the blending crossover proposed in [37], which is formulated in Eq. (7).

$$\vec{u}_{i,j} = \begin{cases} b \cdot \vec{x}_{i,j} + (1-b) \cdot \vec{v}_{i,j}, \ if\ r < Cr\ or\ j = jrand \\ \vec{x}_{i,j}, \text{ otherwise} \end{cases} \tag{7}$$

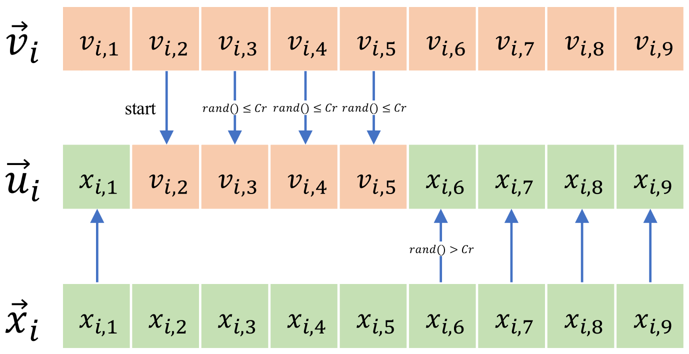
$\vec{v}_i$
$v_{i,1}$ $v_{i,2}$ $v_{i,3}$ $v_{i,4}$ $v_{i,5}$ $v_{i,6}$ $v_{i,7}$ $v_{i,8}$ $v_{i,9}$
start
$rand() \le Cr$ $rand() \le Cr$ $rand() \le Cr$
$\vec{u}_i$
$x_{i,1}$ $v_{i,2}$ $v_{i,3}$ $v_{i,4}$ $v_{i,5}$ $x_{i,6}$ $x_{i,7}$ $x_{i,8}$ $x_{i,9}$
$rand() > Cr$
$\vec{x}_i$
$x_{i,1}$ $x_{i,2}$ $x_{i,3}$ $x_{i,4}$ $x_{i,5}$ $x_{i,6}$ $x_{i,7}$ $x_{i,8}$ $x_{i,9}$

Fig. 3 A 9-dimensional demonstration of the exponential crossover with the parent individual [36]

b is a blending parameter that controls the proportion of each item derived from the parent individuals, which is randomly selected from $\{0.1, 0.5, 0.9\}$. As suggested in [16], Cr is set as a random number following $N(0.5, 0.3)$.

3.3 Boundary repair operator archive

The phenomenon where a generated offspring individual surpasses the defined search space is prevalent. However, the description of the boundary repair operator is neglected in many pieces of literature. To address this gap, we reviewed many source codes provided by corresponding authors and concluded the two most frequently used principles to implement the boundary repair process in Eqs. (8) and (9).

$$\begin{aligned} &if\ \vec{x}_{i,j} > \mathrm{UB}_j\ or\ \vec{x}_{i,j} < \mathrm{LB}_j \\ &then\ \vec{x}_{i,j} = \mathrm{LB}_j + r \cdot (\mathrm{UB}_j - \mathrm{LB}_j) \end{aligned} \tag{8}$$

$$\begin{aligned} &if\ \vec{x}_{i,j} > \mathrm{UB}_j\ then\ \vec{x}_{i,j} = \mathrm{UB}_j \\ &if\ \vec{x}_{i,j} < \mathrm{LB}_j\ then\ \vec{x}_{i,j} = \mathrm{LB}_j \end{aligned} \tag{9}$$

When the value in a specific dimension falls outside the search space, Eq. (8) adopts the random strategy to guarantee that the newly generated value remains within the domain, while Eq. (9) manually modifies any value exceeding the search space, aligning it with the boundary limits. These two simple principles are involved in our designed boundary repair operator archive. Moreover, another simple strategy described in Eq. (10) is also included, which inherits the value in the corresponding dimension from the current best solution x_{best}.

$$\begin{aligned} &if\ \vec{x}_{i,j} > \mathrm{UB}_j\ or\ \vec{x}_{i,j} < \mathrm{LB}_j \\ &then\ \vec{x}_{i,j} = \vec{x}_{\text{best, j}} \end{aligned} \tag{10}$$

Finally, inspired by the opposite-based learning [38], we recognize the potential contribution of this approach to the boundary repair process. The simplest formulation of the opposite-based number is defined in Eq. (11).

$$\vec{x}'_{i,j} = b_1 + b_2 - \vec{x}_{i,j} \tag{11}$$

where b_1 and b_2 are the mapping boundary. We apply this opposite-based mapping operator iteratively until the generated offspring individual is within the search space. Algorithm 1 describes this operator.

Algorithm 1 Iterative opposite-based mapping

```
Require: Solution individual: $\vec{x}_i$; Search space: $LB, UB$; Dimension size: $n$
Ensure: Corrected solution individual: $\vec{x}_i$
1: for $j = 0$ to $n$ do
2:     while $\vec{x}_{i,j} > UB_j$ or $\vec{x}_{i,j} < LB_j$ do
3:         if $\vec{x}_{i,j} > UB_j$ then
4:             $\vec{x}_{i,j} = 2 \cdot UB_j - \vec{x}_{i,j}$
5:         else
6:             $\vec{x}_{i,j} = 2 \cdot LB_j - \vec{x}_{i,j}$
7:         end if
8:     end while
9: end for
10: return $\vec{x}_i$
```

In summary, the pseudocode of our proposed HHDE is shown in Algorithm 2.

Algorithm 2 HHDE

```
Require: Population size $NP$; Search space: $LB, UB$; Dimension size: $n$;
    Maximum iteration: $T$
Ensure: Optimum: $\vec{x}_{best}$
1: Generating initial population $X$ by Eq. (2)
2: $t = 0$
3: while $t < T$ do
4:     for $i = 0$ to $NP$ do
5:         Sampling a mutation operator
6:         Implementing mutation to $\vec{x}_i$
7:         Sampling a crossover operator
8:         Implementing crossover to $\vec{x}_i$
9:         Sampling a boundary repair operator
10:        Implementing boundary repair to $\vec{x}_i$
11:        Evaluating and updating the $\vec{x}_{best}$
12:    end for
13:    $t = t + 1$
14: end while
15: return $\vec{x}_{best}$
```

Additionally, HHDE samples the mutation operator, crossover operator, and boundary repair operator from their respective archive, employing an unbiased probability approach. The benefit of this simple idea is computationally cheap, as it allows for a diverse range of strategies to be used in the evolutionary process, potentially enhancing the adaptability and robustness of the solution.

4 Numerical experiments

This section introduces the numerical experiments. In the following context, Sect. 4.1 details the experimental settings, and Sect. 4.2 presents the experimental results and statistical analyses on CEC2017, CEC2020, and CEC2022 benchmark functions.

4.1 Experiment settings

4.1.1 Benchmark functions

In this study, CEC2017,[1] CEC2020,[2] and CEC2022[3] test suites provided by [39] are employed as benchmark functions to evaluate the comprehensive performance of HHDE in diverse optimization tasks with various characteristics.

4.1.2 Compared methods and parameters

We compare our proposed HHDE with PSO [40], DE [1], covariance matrix adaptation evolution strategy (CMA-ES) [41], comprehensive learning PSO (CLPSO) [42], JADE [17], SHADE [18], L-SHADE [19], phasor PSO (PPSO) [43], improved multi-operator DE (IMODE) [44], and gene-targeting DE (GTDE) [45]; the detailed parameter settings are listed in Table 1. The population size for all competitor algorithms except L-SHADE is fixed at 100, and the maximum fitness evaluations (FEs) are $500 \times D$ (D = Dimension size). To ensure statistical robustness, each algorithm was executed independently 30 times on every single function.

4.2 Experimental results and statistical analyses

This section provides the experimental results and statistical analyses. To assess the significance of our proposed HHDE in comparison to other competing algorithms, we utilize the Mann–Whitney U test for each pair of algorithms. Subsequently, the Holm multiple comparison test is applied to modify the p-values obtained from the Mann–Whitney U test. The symbols $+$, $\approx$, and $-$ are employed to denote whether HHDE significantly outperforms, shows no significant difference, or significantly underperforms compared to the respective algorithm. Additionally, the average rank metric is computed, with the best-performing fitness value highlighted in bold.

[1] https://github.com/P-N-Suganthan/CEC2017-BoundContrained
[2] https://github.com/P-N-Suganthan/2020-Bound-Constrained-Opt-Benchmark
[3] https://github.com/P-N-Suganthan/2022-SO-BO

Table 1 Parameters of competitor algorithms

EAs	Parameters	Value
PSO	Inertia factor w	1
	Acceleration coefficients c_1 and c_2	2.05
	Max. and min. speed	2, − 2
DE	Mutation scheme	DE/cur-to-rand/1
	Scaling factor F	0.8
	Crossover rate Cr	0.9
CMA-ES	σ	1.3
CLPSO	local coefficient c_{local}	1.2
	max. and min. weight	0.9 and 0.4
JADE	μ_F and μ_{Cr}	0.5 and 0.5
	σ_F and σ_{Cr}	0.5 and 0.5
SHADE	μ_F and μ_{Cr}	0.5 and 0.5
	σ_F and σ_{Cr}	0.5 and 0.5
L-SHADE	Population size	$18 \times$ D
	μ_F and μ_{Cr}	0.5 and 0.5
	σ_F and σ_{Cr}	0.5 and 0.5
PPSO	Parameter-free	
IMODE	Mutation scheme	Hybridization
GTDE	μ_F and σ_F	0.7 and 0.5
	μ_{Cr} and σ_{Cr}	0.5 and 0.5
HHDE	μ_F and σ_F	0.5 and 0.3
	μ_{Cr} and σ_{Cr}	0.5 and 0.3

4.2.1 Optimization performance on CEC2017

Tables 2 and 3 list the results on CEC2017 benchmark functions. Due to the limitation of space, only the mean, statistical analysis, and average ranks are summarized.

4.2.2 Optimization performance on CEC2020

Tables 4 and 5 summarize the experimental results on CEC2020 benchmark functions, while Fig. 4 presents the convergence curves of optimizers on CEC2020 representative functions (i.e., f_1: Unimodal function; f_4: Multimodal function; f_6 and f_7: Hybrid functions; f_8 and f_{10}: Composite functions.).

4.2.3 Optimization performance on CEC2022

Tables 6 and 7 summarize the experimental results on CEC2022 benchmark functions, while Fig. 5 presents the convergence curves of optimizers on CEC2022

Table 2 Experimental results and statistical analyses on 30-D CEC2017 benchmark functions

Func	PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	PPSO	IMODE	GTDE	HHDE
f_1	8.970e+09 +	5.257e+10 +	3.444e+10 +	1.037e+09 +	7.381e+07 +	1.406e+07 +	2.460e+07 +	5.894e+09 +	5.955e+06 +	5.977e+05 ≈	**5.518e+05**
f_3	1.297e+05 +	2.608e+05 +	8.126e+04 +	1.080e+05 +	1.159e+05 +	9.641e+04 +	9.253e+04 +	7.370e+04 +	1.167e+05 +	4.846e+04 +	**2.429e+04**
f_4	3.784e+03 +	5.543e+03 +	1.253e+04 +	8.180e+02 +	5.724e+02 +	5.480e+02 ≈	5.479e+02 ≈	1.153e+03 +	**5.235e+02** ≈	5.560e+02 ≈	5.274e+02
f_5	8.478e+02 +	9.290e+02 +	8.701e+02 +	7.452e+02 +	6.875e+02 +	7.105e+02 +	7.017e+02 +	7.960e+02 +	7.085e+02 +	**6.164e+02** ≈	6.383e+02
f_6	6.715e+02 +	6.873e+02 +	6.863e+02 +	6.314e+02 ≈	6.166e+02 −	6.163e+02 −	6.188e+02 ≈	6.683e+02 +	**6.040e+02** −	6.085e+02 −	6.277e+02
f_7	1.187e+03 +	2.771e+03 +	1.319e+03 +	1.028e+03 +	9.274e+02 ≈	9.424e+02 ≈	9.286e+02 ≈	1.353e+03 +	9.452e+02 ≈	**8.875e+02** ≈	9.232e+02
f_8	1.124e+03 +	1.232e+03 +	1.117e+03 +	1.040e+03 +	9.811e+02 +	9.996e+02 +	9.932e+02 +	1.028e+03 +	1.007e+03 +	9.196e+02 ≈	**9.134e+02**
f_9	1.049e+04 +	1.991e+04 +	9.776e+03 +	7.091e+03 +	3.764e+03 +	4.205e+03 +	3.803e+03 +	7.479e+03 +	**1.122e+03** −	2.885e+03 ≈	3.089e+03
f_{10}	9.031e+03 +	8.649e+03 +	8.784e+03 +	7.544e+03 +	6.937e+03 +	7.222e+03 +	7.088e+03 +	7.076e+03 +	8.285e+03 +	6.496e+03 +	**4.938e+03**
f_{11}	5.000e+03 +	5.972e+03 +	7.065e+03 +	2.940e+03 +	2.362e+03 +	1.748e+03 +	1.628e+03 +	2.830e+03 +	1.558e+03 +	1.325e+03 +	**1.225e+03**
f_{12}	1.599e+09 +	2.968e+09 +	8.951e+09 +	8.798e+07 +	3.207e+07 +	1.135e+07 +	1.167e+07 +	2.453e+08 +	8.897e+06 +	2.381e+06 +	**4.012e+05**
f_{13}	3.791e+08 +	3.648e+08 +	8.106e+09 +	1.913e+07 +	1.628e+07 +	2.519e+06 +	2.101e+06 +	6.912e+06 +	1.256e+06 +	3.553e+04 +	**1.733e+04**
f_{14}	7.681e+05 +	2.040e+05 +	3.225e+06 +	6.925e+05 +	4.707e+05 +	1.618e+05 +	1.776e+05 +	1.366e+05 +	6.038e+04 +	**3.112e+03** −	6.162e+03
f_{15}	8.842e+06 +	1.052e+07 +	1.290e+08 +	2.851e+06 +	2.508e+06 +	4.857e+05 +	3.862e+05 +	9.087e+04 +	3.348e+05 +	9.630e+03 ≈	**8.579e+03**

Table 2 (continued)

Func	PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	PPSO	IMODE	GTDE	HHDE
f_{16}	4.325e+03 +	4.009e+03 +	6.692e+03 +	3.412e+03 +	3.125e+03 +	3.237e+03 +	3.065e+03 +	3.770e+03 +	3.369e+03 +	**2.692e+03** ≈	2.707e+03
f_{17}	3.094e+03 +	2.822e+03 +	3.497e+03 +	2.345e+03 +	2.318e+03 +	2.216e+03 ≈	2.189e+03 ≈	2.567e+03 +	2.306e+03 +	2.296e+03 +	**2.120e+03**
f_{18}	1.253e+07 +	9.897e+06 +	1.877e+07 +	2.153e+06 +	2.070e+06 +	9.921e+05 +	1.082e+06 +	2.542e+06 +	2.284e+06 +	1.809e+05 +	**1.098e+05**
f_{19}	2.887e+07 +	4.079e+07 +	2.342e+08 +	3.380e+06 +	2.360e+06 +	4.089e+05 +	3.650e+05 +	1.708e+06 +	3.538e+05 +	1.478e+04 ≈	**1.421e+04**
f_{20}	3.000e+03 +	2.917e+03 +	3.034e+03 +	2.802e+03 +	2.671e+03 ≈	2.585e+03 ≈	2.600e+03 ≈	2.881e+03 +	2.748e+03 +	**2.513e+03** ≈	2.546e+03
f_{21}	2.627e+03 +	2.690e+03 +	2.704e+03 +	2.537e+03 +	2.474e+03 +	2.499e+03 +	2.501e+03 +	2.607e+03 +	2.496e+03 +	2.423e+03 ≈	**2.421e+03**
f_{22}	5.196e+03 +	1.035e+04 +	9.672e+03 +	4.717e+03 +	3.764e+03 +	3.271e+03 +	3.336e+03 +	7.311e+03 +	3.868e+03 +	7.343e+03 +	**2.754e+03**
f_{23}	3.138e+03 +	3.032e+03 +	3.840e+03 +	2.921e+03 +	2.844e+03 +	2.859e+03 +	2.854e+03 +	3.093e+03 +	2.855e+03 +	**2.781e+03** −	2.824e+03
f_{24}	3.304e+03 +	3.163e+03 +	4.041e+03 +	3.081e+03 +	3.019e+03 ≈	3.044e+03 ≈	3.029e+03 ≈	3.303e+03 +	3.017e+03 ≈	**2.970e+03** −	3.027e+03
f_{25}	3.366e+03 +	7.632e+03 +	4.460e+03 +	3.085e+03 +	2.933e+03 ≈	2.919e+03 ≈	2.936e+03 ≈	3.253e+03 +	**2.895e+03** −	2.897e+03 ≈	2.913e+03
f_{26}	8.528e+03 +	8.375e+03 +	1.034e+04 +	6.408e+03 +	5.520e+03 ≈	5.817e+03 ≈	5.715e+03 ≈	8.079e+03 +	5.692e+03 ≈	**5.298e+03** ≈	5.498e+03
f_{27}	3.649e+03 +	3.341e+03 +	5.300e+03 +	3.304e+03 +	3.248e+03 −	3.255e+03 −	3.255e+03 −	3.503e+03 +	**3.236e+03** −	3.250e+03 −	3.285e+03
f_{28}	4.618e+03 +	5.662e+03 +	6.462e+03 +	3.682e+03 +	3.340e+03 +	3.309e+03 +	3.318e+03 +	3.907e+03 +	**3.262e+03** ≈	3.297e+03 ≈	3.270e+03

Table 2 (continued)

Func	PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	PPSO	IMODE	GTDE	HHDE
f_{29}	5.493e+03 +	4.683e+03 +	7.219e+03 +	4.548e+03 +	4.020e+03 ≈	4.169e+03 +	4.165e+03 +	5.271e+03 +	4.284e+03 +	4.026e+03 ≈	**3.972e+03**
f_{30}	3.715e+07 +	4.041e+07 +	1.179e+09 +	1.093e+07 +	1.763e+06 +	1.672e+06 +	1.438e+06 +	4.796e+06 +	9.418e+05 +	5.213e+04 +	**1.771e+04**
+/≈/−	29/0/0	29/0/0	29/0/0	28/1/0	21/6/2	20/7/2	20/8/1	29/0/0	20/5/4	9/15/5	–
Avg. rank	9.4	9.6	10.3	7.2	4.7	4.6	4.4	7.3	3.9	2.4	**1.8**

f_1: Unimodal function; $f_3 - f_9$: Simple multimodal functions; $f_{10} - f_{19}$: Hybrid functions; $f_{20} - f_{30}$: Composite functions

Table 3 Experimental results and statistical analyses on 50-D CEC2017 benchmark functions

Func	PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	PPSO	IMODE	GTDE	HHDE
f_1	3.218e+10 +	1.315e+11 +	8.494e+10 +	1.105e+09 +	2.430e+07 −	3.419e+06 −	1.023e+07 −	1.864e+10 +	**1.689e+06** −	1.095e+08 −	3.408e+08
f_3	2.636e+05 +	4.954e+05 +	1.758e+05 +	2.196e+05 +	2.139e+05 +	2.067e+05 +	1.954e+05 +	1.842e+05 +	2.488e+05 +	1.505e+05 +	**1.050e+05**
f_4	1.117e+04 +	2.444e+04 +	2.453e+04 +	9.839e+02 +	6.850e+02 ≈	6.049e+02 −	6.490e+02 −	2.839e+03 +	**5.988e+02** −	6.549e+02 ≈	7.233e+02
f_5	1.159e+03 +	1.347e+03 +	1.149e+03 +	9.473e+02 +	8.602e+02 +	8.799e+02 +	8.834e+02 +	1.001e+03 +	8.922e+02 +	**7.710e+02** ≈	7.761e+02
f_6	6.850e+02 +	7.046e+02 +	6.995e+02 +	6.304e+02 ≈	6.194e+02 −	6.120e+02 −	6.137e+02 −	6.846e+02 +	**6.022e+02** −	6.189e+02 −	6.396e+02
f_7	1.610e+03 +	5.495e+03 +	2.024e+03 +	1.269e+03 ≈	1.121e+03 −	1.135e+03 −	1.136e+03 −	1.980e+03 +	1.148e+03 −	**1.101e+03** −	1.321e+03
f_8	1.440e+03 +	1.633e+03 +	1.458e+03 +	1.266e+03 +	1.150e+03 +	1.170e+03 +	1.173e+03 +	1.326e+03 +	1.186e+03 +	**1.031e+03** ≈	1.082e+03
f_9	3.821e+04 +	5.899e+04 +	3.588e+04 +	2.304e+04 +	1.374e+04 +	1.205e+04 ≈	1.303e+04 +	2.707e+04 +	**1.121e+03** −	1.063e+04 ≈	1.115e+04
f_{10}	1.534e+04 +	1.513e+04 +	1.528e+04 +	1.281e+04 +	1.115e+04 +	1.136e+04 +	1.153e+04 +	1.207e+04 +	1.455e+04 +	9.204e+03 ≈	**8.415e+03**
f_{11}	1.770e+04 +	3.243e+04 +	2.173e+04 +	6.391e+03 +	9.078e+03 +	3.834e+03 +	3.730e+03 +	5.971e+03 +	3.195e+03 +	1.516e+03 ≈	**1.473e+03**
f_{12}	9.940e+09 +	2.047e+10 +	5.305e+10 +	2.393e+08 +	1.659e+08 +	2.070e+07 +	2.102e+07 +	1.500e+09 +	1.597e+07 +	1.608e+07 +	**1.177e+07**
f_{13}	7.807e+08 +	4.080e+09 +	3.045e+10 +	1.588e+07 +	1.392e+07 +	6.673e+05 +	5.526e+05 +	1.198e+08 +	3.935e+05 +	2.189e+04 ≈	**1.470e+04**
f_{14}	1.102e+07 +	3.010e+06 +	1.309e+08 +	2.746e+06 +	2.230e+06 +	9.150e+05 +	6.156e+05 +	2.427e+06 +	5.659e+05 +	**4.830e+04** ≈	1.051e+05
f_{15}	5.093e+07 +	1.286e+08 +	3.886e+09 +	1.632e+06 +	4.665e+06 +	3.124e+05 +	1.671e+05 +	4.871e+06 +	3.381e+05 +	2.590e+04 +	**9.917e+03**

Table 3 (continued)

Func	PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	PPSO	IMODE	GTDE	HHDE
f_{16}	6.733e+03 +	6.631e+03 +	8.800e+03 +	4.316e+03 +	4.489e+03 +	4.314e+03 +	4.137e+03 +	4.917e+03 +	4.794e+03 +	**3.345e+03** ≈	3.600e+03
f_{17}	4.727e+03 +	5.761e+03 +	5.667e+03 +	3.672e+03 +	3.640e+03 +	3.524e+03 +	3.322e+03 +	4.174e+03 +	3.779e+03 +	3.314e+03 +	**3.176e+03**
f_{18}	7.980e+07 +	3.592e+07 +	1.059e+08 +	7.787e+06 +	8.449e+06 +	5.048e+06 +	4.930e+06 +	8.261e+06 +	7.017e+06 +	**5.329e+05** ≈	8.410e+05
f_{19}	1.164e+08 +	1.318e+08 +	1.521e+09 +	6.206e+05 +	8.443e+05 +	1.353e+05 +	1.039e+05 +	3.839e+06 +	1.366e+05 +	2.729e+04 ≈	**1.433e+04**
f_{20}	4.321e+03 +	4.536e+03 +	4.126e+03 +	3.739e+03 +	3.618e+03 +	3.544e+03 +	3.529e+03 +	3.670e+03 +	3.996e+03 +	3.196e+03 ≈	**3.173e+03**
f_{21}	2.991e+03 +	3.135e+03 +	3.164e+03 +	2.738e+03 +	2.659e+03 +	2.651e+03 +	2.659e+03 +	2.925e+03 +	2.686e+03 +	**2.553e+03** ≈	2.600e+03
f_{22}	1.693e+04 +	1.683e+04 +	1.692e+04 +	1.341e+04 +	1.235e+04 +	1.232e+04 +	1.287e+04 +	1.412e+04 +	1.636e+04 +	1.133e+04 ≈	**9.844e+03**
f_{23}	3.775e+03 +	3.589e+03 +	4.912e+03 +	3.220e+03 ≈	3.105e+03 ≈	3.115e+03 ≈	3.107e+03 ≈	3.837e+03 +	3.116e+03 ≈	**3.088e+03** −	3.187e+03
f_{24}	4.067e+03 +	3.650e+03 +	5.403e+03 +	3.404e+03 ≈	3.277e+03 ≈	3.307e+03 ≈	3.274e+03 ≈	3.951e+03 +	3.289e+03 ≈	**3.218e+03** −	3.319e+03
f_{25}	6.752e+03 +	2.483e+04 +	1.277e+04 +	3.440e+03 +	3.143e+03 ≈	3.157e+03 ≈	3.138e+03 ≈	5.008e+03 +	**3.095e+03** −	3.102e+03 ≈	3.170e+03
f_{26}	1.395e+04 +	1.332e+04 +	1.640e+04 +	8.597e+03 ≈	7.316e+03 −	7.550e+03 −	7.449e+03 −	1.471e+04 +	7.499e+03 −	**7.300e+03** −	8.309e+03
f_{27}	5.298e+03 +	3.813e+03 ≈	8.696e+03 +	3.751e+03 ≈	3.575e+03 −	3.531e+03 −	3.554e+03 −	4.442e+03 +	**3.411e+03** −	3.503e+03 −	3.802e+03
f_{28}	8.216e+03 +	9.751e+03 +	1.082e+04 +	4.431e+03 +	3.478e+03 ≈	3.464e+03 ≈	3.513e+03 ≈	5.718e+03 +	**3.376e+03** −	4.008e+03 ≈	3.559e+03

Table 3 (continued)

Func	PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	PPSO	IMODE	GTDE	HHDE
f_{29}	9.173e+03 +	7.162e+03 +	3.554e+04 +	5.399e+03 ≈	4.818e+03 −	4.876e+03 −	**4.731e+03** −	7.283e+03 +	5.211e+03 ≈	5.078e+03 ≈	5.287e+03
f_{30}	4.743e+08 +	7.313e+08 +	3.468e+09 +	5.350e+07 +	2.500e+07 +	2.033e+07 +	1.957e+07 +	9.836e+07 +	2.545e+07 +	**3.982e+06** ≈	4.380e+06
+/≈/−	29/0/0	28/1/0	29/0/0	22/7/0	18/5/6	17/5/7	18/4/7	29/0/0	16/3/9	4/18/7	–
Avg. rank	9.4	9.8	10.2	6.7	4.7	3.8	3.6	7.8	4.3	**2.4**	3.1

Table 4 Experimental results and statistical analyses on 30-D CEC2020 benchmark functions

Func.		PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	PPSO	IMODE	GTDE	HHDE
f_1	mean	8.970e+09	5.257e+10	9.571e+09	1.037e+09	**2.047e+05**	1.406e+07	2.109e+05	5.894e+09	5.955e+06	3.731e+08	3.658e+05
		+	+	+	+	≈	+	≈	+	+	+	
	std	2.967e+09	3.750e+09	2.081e+09	1.527e+08	8.746e+04	3.916e+06	9.009e+04	2.292e+09	1.735e+06	7.946e+08	4.126e+05
f_2	mean	1.678e+12	4.699e+12	9.116e+11	1.248e+11	2.424e+07	2.336e+09	**1.953e+07**	6.901e+11	8.354e+08	1.930e+09	3.435e+07
		+	+	+	+	≈	+	≈	+	+	+	
	std	7.625e+11	4.486e+11	1.522e+11	2.094e+10	8.099e+06	6.979e+08	6.304e+06	2.151e+11	2.538e+08	2.750e+09	2.619e+07
f_3	mean	4.529e+11	1.543e+12	2.970e+11	4.468e+10	**9.931e+06**	7.556e+08	1.257e+07	2.109e+11	4.125e+08	3.808e+08	2.245e+07
		+	+	+	+	≈	+	≈	+	+	+	
	std	2.315e+11	1.521e+11	5.701e+10	1.091e+10	3.819e+06	1.736e+08	5.905e+06	9.112e+10	1.888e+08	4.950e+08	2.264e+07
f_4	mean	9.772e+04	2.604e+05	4.065e+03	2.015e+03	1.917e+03	1.920e+03	**1.916e+03**	7.767e+03	1.919e+03	1.927e+03	1.925e+03
		+	+	+	+	≈	≈	≈	+	≈	≈	
	std	1.676e+05	1.299e+05	1.437e+03	5.164e+01	1.021e+00	1.495e+00	1.301e+00	4.453e+03	1.114e+00	7.272e+00	9.150e+00
f_5	mean	2.523e+07	2.135e+07	2.046e+06	1.106e+06	3.597e+05	1.070e+06	1.396e+06	3.736e+06	3.443e+06	1.517e+06	**1.375e+05**
		+	+	+	+	≈	+	+	+	+	+	
	std	7.585e+06	4.696e+06	8.228e+05	3.634e+05	3.625e+05	3.173e+05	4.398e+05	3.313e+06	9.876e+05	9.282e+05	4.215e+04
f_6	mean	1.414e+06	3.930e+05	3.291e+04	8.979e+04	1.375e+04	2.522e+04	1.854e+04	5.185e+04	4.258e+04	3.962e+04	**6.478e+03**
		+	+	+	+	+	+	+	+	+	+	
	std	2.934e+06	2.244e+05	5.767e+03	3.163e+04	5.863e+03	7.023e+03	7.310e+03	3.212e+04	1.974e+04	2.112e+04	4.412e+03
f_7	mean	6.420e+07	8.318e+07	2.912e+06	2.927e+06	4.677e+05	1.681e+06	1.007e+06	5.287e+06	2.694e+06	1.449e+06	**2.520e+05**
		+	+	+	+	+	+	+	+	+	+	
	std	5.167e+07	3.061e+07	8.421e+05	1.131e+06	3.166e+05	5.634e+05	2.077e+05	4.419e+06	8.082e+05	7.016e+05	2.615e+05
f_8	mean	2.844e+03	2.753e+03	2.505e+03	2.441e+03	2.375e+03	2.389e+03	**2.371e+03**	2.886e+03	2.380e+03	2.412e+03	2.406e+03
		+	+	+	+	−	≈	−	+	≈	≈	
	std	1.949e+02	5.277e+01	1.199e+01	8.380e+00	3.475e+00	3.482e+00	3.003e+00	2.555e+02	1.429e+00	1.925e+01	1.663e+01
f_9	mean	1.220e+04	1.436e+04	8.835e+03	5.419e+03	2.634e+03	2.908e+03	**2.623e+03**	8.948e+03	2.773e+03	5.770e+03	2.631e+03
		+	+	+	+	≈	+	≈	+	+	+	
	std	2.398e+03	5.304e+02	3.637e+02	2.124e+02	3.079e+00	2.580e+01	3.024e+00	1.323e+03	2.183e+01	2.406e+03	4.819e+01

Table 4 (continued)

Func.		PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	PPSO	IMODE	GTDE	HHDE
f_{10}	mean	4.326e+03	6.315e+03	3.422e+03	3.099e+03	**2.922e+03**	2.943e+03	2.922e+03	3.496e+03	2.926e+03	2.951e+03	2.944e+03
		+	+	+	+	−	≈	−	+	≈	≈	
	std	1.134e+03	6.492e+02	1.058e+02	3.351e+01	8.018e−02	8.797e+00	1.528e−01	1.854e+02	1.905e+00	4.225e+01	2.514e+01
+/≈/−		10/0/0	10/0/0	10/0/0	10/0/0	2/6/2	7/3/0	3/5/2	10/0/0	7/3/0	7/3/0	–
Avg. rank		10.1	10.6	7.9	6.9	**1.8**	4.6	2.1	8.8	4.7	5.6	2.9

f_1: Unimodal function; $f_2 - f_4$: Multimodal functions; $f_5 - f_7$: Hybrid functions; $f_8 - f_{10}$: Composite functions

Table 5 Experimental results and statistical analyses on 50-D CEC2020 benchmark functions

Func.		PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	PPSO	IMODE	GTDE	HHDE
f_1	mean	3.218e+10	1.315e+11	2.507e+10	1.105e+09	**5.999e+03**	3.419e+06	2.257e+04	1.864e+10	1.689e+06	2.854e+08	2.250e+08
		+	+	+	+	−	−	−	+	−	+	
	std	7.352e+09	1.303e+10	5.676e+09	1.930e+08	3.968e+03	1.424e+06	1.347e+04	5.420e+09	3.336e+05	5.142e+08	9.981e+07
f_2	mean	3.488e+12	1.422e+13	3.055e+12	1.258e+11	**5.833e+05**	4.132e+08	3.438e+06	2.306e+12	1.811e+08	1.503e+11	3.966e+10
		+	+	+	+	−	−	−	+	−	+	
	std	1.133e+12	1.004e+12	6.845e+11	1.587e+10	2.916e+05	1.491e+08	1.916e+06	8.122e+11	8.570e+07	1.861e+11	4.652e+10
f_3	mean	1.400e+12	4.976e+12	1.017e+12	3.395e+10	**2.193e+05**	1.248e+08	7.866e+05	6.077e+11	6.693e+07	1.786e+10	1.070e+10
		+	+	+	+	−	−	−	+	−	≈	
	std	7.373e+11	5.674e+11	2.023e+11	4.727e+09	1.772e+05	4.500e+07	9.264e+05	1.721e+11	5.269e+07	2.808e+10	1.378e+10
f_4	mean	5.110e+05	2.439e+06	6.757e+04	2.072e+03	1.931e+03	1.937e+03	**1.930e+03**	2.130e+04	1.936e+03	1.950e+03	2.017e+03
		+	+	+	+	−	−	−	+	−	≈	
	std	6.369e+05	9.536e+05	5.509e+04	5.268e+01	1.704e+00	1.852e+00	1.523e+00	9.602e+03	2.413e+00	1.187e+01	5.423e+01
f_5	mean	1.207e+08	8.681e+07	8.232e+06	8.268e+06	**1.308e+05**	4.471e+06	1.426e+06	2.368e+07	9.572e+06	6.312e+06	1.253e+06
		+	+	+	+	−	+	≈	+	+	+	
	std	3.951e+07	2.813e+07	1.446e+06	1.705e+06	5.561e+04	9.626e+05	7.847e+05	1.532e+07	2.918e+06	3.767e+06	1.109e+06
f_6	mean	7.501e+07	3.483e+08	4.659e+05	2.182e+05	**3.103e+03**	2.275e+04	8.969e+03	2.490e+06	2.289e+04	2.489e+04	9.137e+03
		+	+	+	+	−	+	≈	+	+	+	
	std	1.396e+08	1.102e+08	1.771e+05	7.685e+04	8.194e+02	7.014e+03	4.206e+03	3.071e+06	7.035e+03	1.276e+04	7.247e+03
f_7	mean	9.309e+08	1.141e+09	5.488e+07	1.822e+07	**1.959e+05**	8.831e+06	2.807e+06	5.787e+07	1.312e+07	1.022e+07	2.834e+06
		+	+	+	+	−	+	≈	+	+	+	
	std	5.060e+08	3.944e+08	1.517e+07	3.849e+06	1.239e+05	2.205e+06	1.527e+06	4.841e+07	5.217e+06	6.133e+06	2.873e+06
f_8	mean	4.495e+03	3.574e+03	2.762e+03	2.532e+03	2.413e+03	2.419e+03	**2.405e+03**	5.935e+03	2.413e+03	2.557e+03	2.682e+03
		+	+	+	−	−	−	−	+	−	−	
	std	8.203e+02	2.490e+02	5.833e+01	1.522e+01	6.127e+00	4.089e+00	3.873e+00	1.957e+03	6.708e+00	2.408e+01	1.484e+02
f_9	mean	3.240e+04	3.129e+04	1.550e+04	5.851e+03	**2.604e+03**	2.815e+03	2.607e+03	2.291e+04	2.693e+03	1.026e+04	4.131e+03
		+	+	+	+	−	−	−	+	−	+	
	std	7.528e+03	2.869e+03	1.099e+03	3.661e+02	1.006e+00	2.154e+02	2.230e+00	1.024e+04	1.369e+01	1.789e+03	6.729e+02

Table 5 (continued)

Func.		PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	PPSO	IMODE	GTDE	HHDE
f_{10}	mean	1.167e+04	1.742e+04	6.231e+03	4.011e+03	**3.289e+03**	3.385e+03	3.303e+03	6.217e+03	3.302e+03	3.456e+03	3.718e+03
		+	+	+	+	−	−	−	+	−	−	
	std	3.317e+03	1.721e+03	6.959e+02	4.671e+01	3.509e+01	4.664e+01	2.432e+01	1.143e+03	2.224e+01	1.486e+02	1.997e+02
+/≈/−		10/0/0	10/0/0	10/0/0	9/0/1	0/0/10	3/0/7	0/3/7	10/0/0	3/0/7	6/2/2	–
Avg. rank		10.2	10.6	8.3	6.6	**1.3**	4.0	2.0	8.7	3.8	5.8	4.7

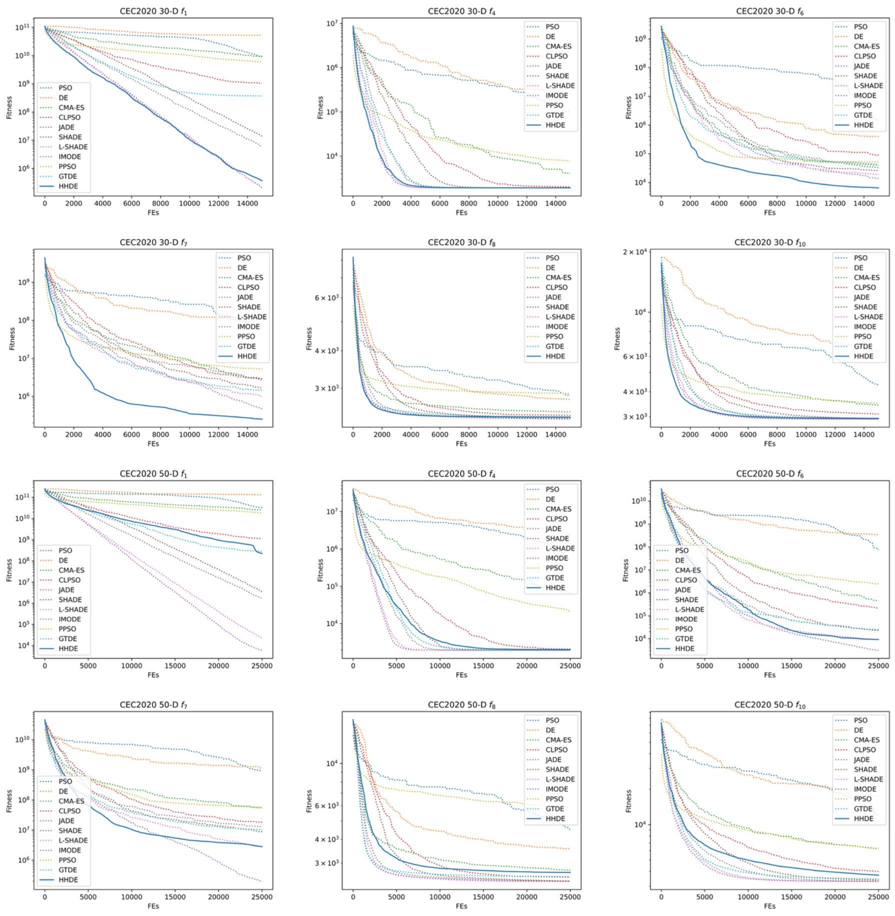
Fig. 4 Convergence curves of optimizers on CEC2020 representative functions

representative functions (i.e., f_1: Unimodal function; f_3 and f_4: Basic functions; f_6: Hybrid function; f_9 and f_{12}: Composite functions.).

5 Discussion

5.1 Computational complexity analysis

Supposing the population size is N, the dimension size is D, and the maximum iteration is T, the computational complexity of HHDE is methodically analyzed according to its procedural flowchart.

The first process is population initialization, and the computational complexity is $O(N \cdot D)$. Then, HHDE enters the iteration. For a single individual, the computational complexity of the mutation operator sampling can be ignored, since the number

Table 6 Experimental results and statistical analyses on 10-D CEC2022 benchmark functions

Func.		PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	PPSO	IMODE	GTDE	HHDE
f_1	mean	3.495e+03	1.115e+04	1.708e+03	4.719e+03	1.018e+03	1.816e+03	1.946e+03	7.223e+02	1.781e+03	6.033e+02	**3.000e+02**
		+	+	+	+	+	+	+	+	+	+	
	std	1.422e+03	2.556e+03	2.908e+02	1.320e+03	2.638e+02	6.173e+02	4.772e+02	3.839e+02	4.588e+02	1.851e+02	2.293e−10
f_2	mean	5.650e+02	5.054e+02	4.409e+02	4.901e+02	4.135e+02	4.329e+02	4.390e+02	4.555e+02	4.148e+02	**4.106e+02**	4.194e+02
		+	+	+	+	≈	+	+	+	≈	≈	
	std	5.115e+01	2.539e+01	1.226e+01	2.462e+01	3.474e+00	7.497e+00	9.974e+00	2.886e+01	4.778e+00	4.252e+00	2.579e+01
f_3	mean	6.000e+02	6.001e+02	6.000e+02	6.000e+02	6.000e+02	6.000e+02	6.000e+02	6.000e+02	6.000e+02	6.000e+02	**6.000e+02**
		+	+	+	+	+	+	+	+	+	+	
	std	2.565e−02	2.926e−02	3.177e−03	8.259e−03	1.192e−04	6.933e−04	1.272e−03	2.464e−03	1.707e−04	7.530e−06	5.084e−14
f_4	mean	8.012e+02	8.013e+02	8.011e+02	8.009e+02	8.010e+02	8.006e+02	8.005e+02	8.004e+02	8.011e+02	8.008e+02	**8.003e+02**
		+	+	+	+	+	+	+	+	+	+	
	std	1.961e−01	2.521e−01	1.999e−01	2.146e−01	2.447e−01	1.518e−01	1.180e−01	2.191e−01	2.120e−01	1.629e−01	1.621e−01
f_5	mean	9.011e+02	9.035e+02	9.012e+02	9.013e+02	**9.001e+02**	9.006e+02	9.006e+02	9.023e+02	9.002e+02	9.002e+02	9.002e+02
		+	+	+	+	≈	+	+	+	≈	≈	
	std	6.681e−01	7.869e−01	3.756e−01	5.543e−01	4.459e−02	1.136e−01	1.234e−01	1.777e+00	4.770e−02	1.861e−01	4.048e−01
f_6	mean	4.393e+06	4.167e+05	4.492e+04	1.308e+06	9.140e+04	2.671e+05	2.076e+05	3.716e+04	1.294e+05	5.902e+04	**4.424e+03**
		+	+	+	+	+	+	+	+	+	+	
	std	5.181e+06	1.853e+05	1.912e+04	1.131e+06	3.974e+04	1.544e+05	1.362e+05	1.608e+04	7.656e+04	1.473e+04	3.489e+03
f_7	mean	2.185e+03	2.155e+03	2.128e+03	2.103e+03	2.068e+03	2.054e+03	2.060e+03	2.088e+03	2.065e+03	2.052e+03	**2.025e+03**
		+	+	+	+	+	+	+	+	+	+	
	std	7.953e+01	2.990e+01	2.801e+01	2.377e+01	8.594e+00	9.575e+00	1.289e+01	3.588e+01	1.251e+01	9.680e+00	4.080e+00
f_8	mean	3.351e+03	2.387e+03	2.243e+03	2.496e+03	2.261e+03	2.267e+03	2.280e+03	2.766e+03	2.254e+03	2.295e+03	**2.222e+03**
		+	+	+	+	+	+	+	+	+	+	
	std	1.286e+03	6.011e+01	7.625e+00	1.028e+02	1.462e+01	1.924e+01	1.755e+01	6.466e+02	1.402e+01	1.206e+02	7.765e+00
f_9	mean	2.734e+03	2.695e+03	2.626e+03	2.681e+03	2.529e+03	**2.395e+03**	2.432e+03	2.742e+03	2.472e+03	2.668e+03	2.516e+03
		+	+	≈	+	≈	≈	≈	+	≈	+	
	std	5.158e+01	1.532e+01	4.544e+01	1.073e+02	1.215e+02	3.337e+01	6.575e+01	1.559e+02	9.024e+01	5.352e+00	1.768e+02

Table 6 (continued)

Func.		PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	PPSO	IMODE	GTDE	HHDE
f_{10}	mean	2.623e+03	2.646e+03	2.616e+03	2.638e+03	2.607e+03	2.618e+03	2.616e+03	2.641e+03	**2.607e+03**	2.646e+03	2.628e+03
		≈	+	≈	+	−	≈	−	+	−	≈	
	std	7.574e+00	9.781e+00	1.451e+00	1.140e+01	3.093e+00	5.581e+00	4.472e+00	6.667e+01	1.646e+00	6.877e+01	5.403e+01
f_{11}	mean	2.643e+03	2.652e+03	2.726e+03	2.645e+03	2.604e+03	2.612e+03	2.614e+03	2.897e+03	2.608e+03	2.789e+03	**2.600e+03**
		+	+	+	+	+	+	+	+	+	+	
	std	1.528e+01	1.205e+01	2.554e+02	1.284e+01	2.488e+00	3.744e+00	2.327e+00	4.268e+02	3.113e+00	3.437e+02	1.596e−06
f_{12}	mean	2.898e+03	2.871e+03	2.869e+03	2.893e+03	2.867e+03	2.872e+03	2.872e+03	2.884e+03	2.868e+03	**2.867e+03**	2.872e+03
		+	≈	≈	+	≈	+	+	+	≈	≈	
	std	8.245e+00	1.192e+00	6.909e−01	6.397e+00	4.264e−01	1.690e+00	1.823e+00	2.331e+01	4.088e−01	1.464e+00	9.656e+00
+/≈/−		11/1/0	11/1/0	9/3/0	12/0/0	7/4/1	10/2/0	10/1/1	12/0/0	7/4/1	8/4/0	–
Avg. rank		9.4	9.4	6.1	8.6	3.5	5.0	5.3	7.4	3.9	4.6	**2.6**

f_1: Unimodal function; $f_2 - f_5$: Basic functions; $f_6 - f_8$: Hybrid functions; $f_9 - f_{12}$: Composite functions

Table 7 Experimental results and statistical analyses on 20-D CEC2022 benchmark functions

Func.		PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	PPSO	IMODE	GTDE	HHDE
f_1	mean	1.732e+04	4.718e+04	7.479e+03	1.343e+04	1.262e+03	5.660e+03	5.927e+03	6.752e+03	4.921e+03	2.753e+03	**3.000e+02**
		+	+	+	+	+	+	+	+	+	+	
	std	2.848e+03	6.334e+03	1.086e+03	2.680e+03	3.009e+02	9.396e+02	1.139e+03	2.280e+03	9.475e+02	8.211e+02	2.868e−02
f_2	mean	9.184e+02	1.426e+03	6.339e+02	6.072e+02	**4.496e+02**	4.837e+02	4.900e+02	6.408e+02	4.585e+02	4.617e+02	4.537e+02
		+	+	+	+	≈	+	+	+	+	+	
	std	2.042e+02	1.976e+02	3.486e+01	2.606e+01	1.063e−01	1.454e+01	1.220e+01	9.083e+01	1.052e+01	2.148e+01	9.468e+00
f_3	mean	6.004e+02	6.006e+02	6.001e+02	6.000e+02	6.000e+02	6.000e+02	6.000e+02	6.001e+02	6.000e+02	6.000e+02	**6.000e+02**
		+	+	+	+	+	+	+	+	+	+	
	std	1.237e−01	1.214e−01	1.339e−02	9.727e−03	1.875e−05	3.766e−04	3.883e−04	5.816e−02	6.337e−05	7.616e−05	8.952e−10
f_4	mean	8.036e+02	8.041e+02	8.042e+02	8.029e+02	8.038e+02	8.022e+02	8.022e+02	8.017e+02	8.037e+02	8.027e+02	**8.006e+02**
		+	+	+	+	+	+	+	+	+	+	
	std	3.837e−01	5.351e−01	2.607e−01	2.685e−01	3.195e−01	3.625e−01	3.043e−01	7.082e−01	3.882e−01	7.628e−01	3.460e−01
f_5	mean	9.089e+02	9.188e+02	9.056e+02	9.037e+02	**9.001e+02**	9.018e+02	9.020e+02	9.054e+02	9.003e+02	9.031e+02	9.016e+02
		+	+	+	+	−	≈	≈	+	−	≈	
	std	3.150e+00	2.324e+00	9.892e−01	6.986e−01	2.944e−02	3.515e−01	4.228e−01	2.751e+00	9.467e−02	2.175e+00	9.545e−01
f_6	mean	9.035e+08	4.585e+08	2.196e+07	4.845e+07	8.157e+06	1.292e+07	9.342e+06	2.095e+06	2.966e+07	1.366e+06	**5.231e+04**
		+	+	+	+	+	+	+	+	+	+	
	std	6.779e+08	1.529e+08	9.031e+06	2.698e+07	3.709e+06	4.477e+06	5.664e+06	2.794e+06	7.766e+06	1.301e+06	2.297e+04
f_7	mean	3.364e+03	3.188e+03	2.480e+03	2.501e+03	2.155e+03	2.202e+03	2.208e+03	2.865e+03	2.261e+03	2.212e+03	**2.057e+03**
		+	+	+	+	+	+	+	+	+	+	
	std	7.845e+02	3.014e+02	1.006e+02	1.132e+02	3.832e+01	4.481e+01	5.501e+01	3.579e+02	6.473e+01	7.805e+01	2.461e+01
f_8	mean	3.967e+07	1.326e+06	3.895e+03	2.512e+04	5.280e+03	3.327e+03	3.495e+03	5.079e+03	6.348e+03	5.751e+03	**2.311e+03**
		+	+	+	+	+	+	+	+	+	+	
	std	9.410e+07	2.025e+06	5.004e+02	5.231e+04	9.903e+02	4.422e+02	5.209e+02	1.138e+03	2.731e+03	1.677e+03	1.123e+02
f_9	mean	3.272e+03	2.930e+03	2.706e+03	2.811e+03	2.647e+03	2.695e+03	2.709e+03	3.046e+03	2.664e+03	2.642e+03	**2.639e+03**
		+	+	+	+	+	+	+	+	+	+	
	std	2.269e+02	1.003e+02	2.057e+01	3.486e+01	3.304e+00	1.313e+01	1.133e+01	3.365e+02	7.981e+00	6.005e+00	3.641e+00

Table 7 (continued)

Func.		PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	PPSO	IMODE	GTDE	HHDE
f_{10}	mean	2.886e+03	4.630e+03	3.587e+03	3.066e+03	**2.799e+03**	2.857e+03	2.892e+03	4.709e+03	2.831e+03	5.101e+03	2.955e+03
		≈	+	+	+	−	≈	≈	+	≈	+	
	std	7.902e+01	1.805e+03	1.396e+03	2.305e+02	3.752e+01	4.411e+01	1.135e+02	1.268e+03	7.917e+01	1.929e+03	5.539e+02
f_{11}	mean	2.937e+03	2.990e+03	2.643e+03	2.667e+03	2.601e+03	2.609e+03	2.612e+03	2.798e+03	2.603e+03	3.557e+03	**2.600e+03**
		+	+	+	+	+	+	+	+	+	+	
	std	1.900e+02	4.455e+02	7.767e+00	1.710e+01	1.187e−01	1.748e+00	1.719e+00	4.448e+02	2.470e+00	6.597e+02	7.983e−02
f_{12}	mean	3.228e+03	2.982e+03	2.973e+03	3.052e+03	**2.947e+03**	2.971e+03	2.970e+03	3.095e+03	2.951e+03	2.968e+03	3.031e+03
		+	≈	≈	≈	−	≈	≈	≈	−	−	
	std	6.376e+01	1.043e+01	5.749e+00	1.668e+01	8.600e+00	7.856e+00	7.879e+00	9.169e+01	5.937e+00	2.762e+01	6.081e+01
+/≈/−		11/1/0	11/1/0	11/1/0	11/1/0	8/1/3	9/3/0	9/3/0	11/1/0	9/1/2	10/1/1	–
Avg. rank		9.5	9.9	7.3	7.8	2.8	4.2	5.0	7.5	4.5	5.2	**2.2**

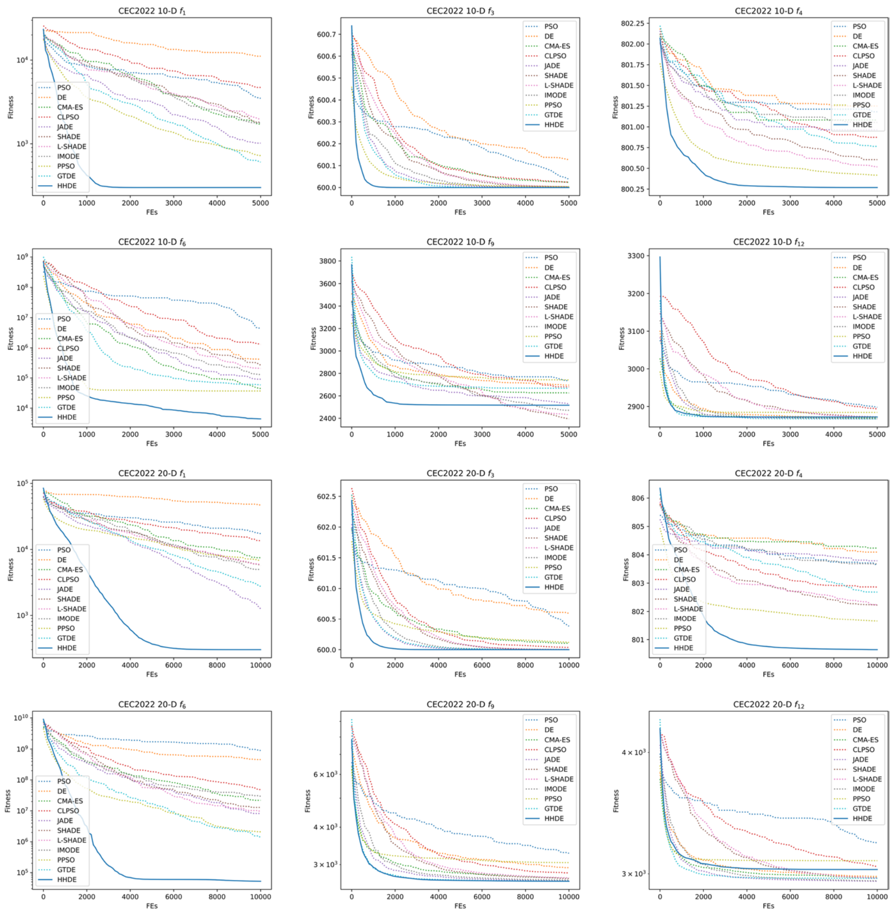

Fig. 5 Convergence curves of optimizers on CEC2022 representative functions

of mutation operators is a constant. The computational complexity of *rand*/1, *best*/1, *cur*/1, *cur*2*best*/1, and *cur*2*pbest*/1 is $O(D)$. Although *cur*2*pbest*/1 computes the mean of the p-best individuals, this operator increases the practical complexity but does not affect the theoretical analysis. Subsequently, both the crossover operator and the boundary repair operator exhibit computational complexities that are analogous to those of the mutation module. In summary, the total computational complexity of HHDE is expressed in Eq. (12).

$$\begin{aligned} &O(N \cdot D + T \cdot (N \cdot D + N \cdot D + N \cdot D)) \\ =&O(N \cdot D + T \cdot (3N \cdot D)) \\ :=&O(T \cdot N \cdot D) \end{aligned} \tag{12}$$

5.2 Performance analysis on CEC2017, CEC2020, and CEC2022

This section delves into analyzing our proposed HHDE performance on the standardized CEC2017, CEC2020, and CEC2022 benchmark functions. These benchmark suites encompass a diverse range of test functions with distinct characteristics, including unimodal, multimodal, hybrid, and composite structures. Therefore, optimizing across these test functions enables a comprehensive evaluation of the efficacy of optimizers, facilitating thorough investigations into the features of the involved algorithms.

At the outset, the unimodal functions f_1 within the CEC2017, CEC2020, and CEC2022 serve as initial benchmarks, allowing for an evaluation of the optimization process in terms of exploitative capabilities. Notably, the HHDE demonstrates marked superiority in 10-D, 20-D, and 30-D scenarios compared to state-of-the-art optimizers. However, its performance undergoes significant degradation in 50-D dimensions when compared with competitor algorithms such as JADE, SHADE, L-SHADE, IMODE, and GTDE. Through this phenomenon, we infer that a meticulously designed parameter adaptation scheme may contribute more influence on optimization accuracy in high-dimensional search spaces, rather than the diversity of search operators.

Following this, f_3 to f_9 within CEC2017, f_2 to f_4 within CEC2020, and f_2 to f_5 within CEC2022 are unimodal functions. These functions exhibit multiple local optima, offering an opportunity to assess optimizers' abilities in both escaping local optima and achieving global convergence. The competitiveness of our proposed HHDE is evident through experimental results and statistical analyses detailed in Sects. 4.2.1, 4.2.2, and 4.2.3. While benchmarked against state-of-the-art optimizers and advanced variants such as JADE, SHADE, L-SHADE, IMODE, and GTDE, HHDE may display inferior performance in certain instances, such as the 30-D f_f function in CEC2017. Nonetheless, the exceptional performance of HHDE should not be overlooked, as its capacity to escape from local optima and achieve global convergence is empirically validated through the results.

Subsequently, the remaining functions encompass hybrid and composite structures, characterized by intricate fitness landscapes and numerous optima. These properties present significant challenges to optimizers, requiring a delicate balance between exploitation and exploration, avoidance of premature convergence, and attainment of global optimization. Upon reviewing the average rank summary, it is clear that HHDE consistently exhibits superior performance across numerous test functions within this category. This underscores its efficiency and effectiveness in complex optimization environments.

Although our proposed HHDE demonstrates satisfactory performance across diverse optimization tasks, it is essential to address a significant issue: HHDE consistently performs worse than JADE across all instances of the 50-D CEC2020 benchmark functions. We can turn to the No Free Lunch Theorem [46, 47] to explain this phenomenon. This theorem suggests that if an algorithm excels in a specific class of problems, it inherently trades off by performing less effectively in other problem classes. Essentially, it implies that any pair of algorithms will exhibit identical average performance across all conceivable problem domains. Consequently, our future research efforts will be directed toward the further advancement of HHDE, focusing on developing problem-specific variants to address this challenge.

6 Conclusion

This paper proposes a novel hyper-heuristic differential evolution (HHDE) for numerical optimization. Inspired by the architecture of the HH algorithm, our approach involves the creation of distinct archives for mutation operators, crossover operators, and boundary repair operators. As a primary study, the learning-free selection function with unbiased probability is employed to sample the search operator. Moreover, we focus on the boundary repair module, which is usually ignored in the EA description. Based on the previous research on boundary repair, we embed two strategies in our designed boundary repair archive: optimum inheritance and iterative opposite-based mapping. Comprehensive numerical experiments on CEC2017, CEC2020, and CEC2022 benchmark functions adequately confirm our proposed HHDE's competitiveness with state-of-the-art optimizers.

In future research, we are committed to the ongoing development and refinement of the proposed HHDE. A promising topic is to intelligently determine the optimization sequence, where machine learning and reinforcement learning techniques may further significantly enhance the performance of the HHDE.

Acknowledgements This work was supported by JSPS KAKENHI Grant Number 21A402 and 24K15098 and JST SPRING Grant Number JPMJSP2119.

Author Contributions Rui Zhong helped in conceptualization, methodology, writing—original draft, writing—review & editing, and funding acquisition. Shilong Zhang was involved in validation, formal analysis, and writing—review & editing. Jun Yu contributed to methodology, investigation, and writing—review & editing. Masaharu Munetomo helped in writing—review & editing and project administration.

Data Availability The source code of this research code can be downloaded from https://github.com/RuiZhong961230/HHDE.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Storn R (1996) On the usage of differential evolution for function optimization. In: Proceedings of North American Fuzzy Information Processing, pp 519–523 . https://doi.org/10.1109/NAFIPS.1996.534789
2. Huang C, Zhou X, Ran X, Wang J, Chen H, Deng W (2023) Adaptive cylinder vector particle swarm optimization with differential evolution for UAV path planning. Eng Appl Artif Intell 121:105942. https://doi.org/10.1016/j.engappai.2023.105942

3. Li C, Sun G, Deng L, Qiao L, Yang G (2023) A population state evaluation-based improvement framework for differential evolution. Inf Sci 629:15–38. https://doi.org/10.1016/j.ins.2023.01.120
4. Zhang Y, Li S, Wang Y, Yan Y, Zhao J, Gao Z (2024) Self-adaptive enhanced learning differential evolution with surprisingly efficient decomposition approach for parameter identification of photovoltaic models. Energy Convers Manage 308:118387. https://doi.org/10.1016/j.enconman.2024.118387
5. He L, Cao Y, Li W, Cao J, Zhong L (2022) Optimization of energy-efficient open shop scheduling with an adaptive multi-objective differential evolution algorithm. Appl Soft Comput 118:108459. https://doi.org/10.1016/j.asoc.2022.108459
6. Rauf HT, Gao J, Almadhor A, Haider A, Zhang Y-D, Al-Turjman F (2023) Multi population-based chaotic differential evolution for multi-modal and multi-objective optimization problems. Appl Soft Comput 132:109909. https://doi.org/10.1016/j.asoc.2022.109909
7. Ma X, Huang Z, Li X, Wang L, Qi Y, Zhu Z (2022) Merged differential grouping for large-scale global optimization. IEEE Trans Evol Comput 26(6):1439–1451. https://doi.org/10.1109/TEVC.2022.3144684
8. Rui Zhong, Zhang E, Munetomo M (2023) Cooperative coevolutionary surrogate ensemble-assisted differential evolution with efficient dual differential grouping for large-scale expensive optimization problems. Complex Intell Syst 2023:1–21. https://doi.org/10.1007/s40747-023-01262-6
9. Debnath S, Baishya S, Sen D, Arif W (2021) A hybrid memory-based dragonfly algorithm with differential evolution for engineering application. Eng Comput 37:2775–2802. https://doi.org/10.1007/s00366-020-00958-4
10. Zhong R, Yu J (2024) DEA2H2: differential evolution architecture based adaptive hyper-heuristic algorithm for continuous optimization. Clust Comput 8:1–28. https://doi.org/10.1007/s10586-024-04587-0
11. Tiwari P, Mishra V, Parouha R (2024) Developments and design of differential evolution algorithm for non-linear/non-convex engineering optimization. Arch Comput Methods Eng. https://doi.org/10.1007/s11831-023-10036-9
12. Kaur M, Singh D, Chahar V (2020) Drug synergy prediction using dynamic mutation based differential evolution. Curr Pharm Des. https://doi.org/10.2174/1381612826666201106090938
13. Luukkonen S, van den Maagdenberg HW, Emmerich MTM, van Westen GJP (2023) Artificial intelligence in multi-objective drug design. Curr Opin Struct Biol 79:102537. https://doi.org/10.1016/j.sbi.2023.102537
14. Liu M, Yao X, Li Y (2020) Hybrid whale optimization algorithm enhanced with lévy flight and differential evolution for job shop scheduling problems. Appl Soft Comput 87:105954. https://doi.org/10.1016/j.asoc.2019.105954
15. Zhao H, Tang L, Li JR, Liu J (2023) Strengthening evolution-based differential evolution with prediction strategy for multimodal optimization and its application in multi-robot task allocation. Appl Soft Comput 139:110218. https://doi.org/10.1016/j.asoc.2023.110218
16. Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: 2005 IEEE Congress on Evolutionary Computation, vol 2, pp 1785–17912 . https://doi.org/10.1109/CEC.2005.1554904
17. Zhang J, Sanderson AC (2009) Jade: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958. https://doi.org/10.1109/TEVC.2009.2014613
18. Tanabe R, Fukunaga A (2013) Success-history based parameter adaptation for differential evolution. In: 2013 IEEE Congress on Evolutionary Computation, pp 71–78 . https://doi.org/10.1109/CEC.2013.6557555
19. Tanabe R, Fukunaga AS (2014) Improving the search performance of shade using linear population size reduction. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp 1658–1665 . https://doi.org/10.1109/CEC.2014.6900380
20. Bilal Pant M, Zaheer H, Garcia-Hernandez L, Abraham A (2020) Differential evolution: a review of more than two decades of research. Eng Appl Artif Intell 90:103479. https://doi.org/10.1016/j.engappai.2020.103479
21. 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(5):3831–3872. https://doi.org/10.1016/j.aej.2021.09.013
22. Chakraborty S, Saha AK, Ezugwu A, Agushaka O, Zitar R, Abualigah L (2022) Differential evolution and its applications in image processing problems: a comprehensive review. Arch Comput Methods Eng 30:1–56. https://doi.org/10.1007/s11831-022-09825-5

23. Gölcük İ, Ozsoydan FB (2021) Q-learning and hyper-heuristic based algorithm recommendation for changing environments. Eng Appl Artif Intell 102:104284. https://doi.org/10.1016/j.engappai.2021.104284
24. Dowsland KA (1998) Off-the-peg or made-to-measure? timetabling and scheduling with SA and TS. In: Burke E, Carter M (eds) Practice and Theory of Automated Timetabling II. Springer, Berlin, Heidelberg, pp 37–52. https://doi.org/10.1007/BFb0055880
25. Jackson WG, Özcan E, Drake JH (2013) Late acceptance-based selection hyper-heuristics for cross-domain heuristic search. In: 2013 13th UK Workshop on Computational Intelligence (UKCI), pp 228–235. https://doi.org/10.1109/UKCI.2013.6651310
26. Kheiri A, Keedwell E (2022) Selection hyper-heuristics. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. GECCO '22, pp 983–996. Association for Computing Machinery, New York, NY, USA . https://doi.org/10.1145/3520304.3533655
27. Cruz-Duarte JM, Amaya I, Ortiz-Bayliss JC, Conant-Pablos SE, Terashima-Marín H, Shi Y (2021) Hyper-heuristics to customise metaheuristics for continuous optimisation. Swarm Evol Comput 66:100935. https://doi.org/10.1016/j.swevo.2021.100935
28. Zhong R, Yu J, Zhang C, Munetomo M (2023) Surrogate ensemble-assisted hyper-heuristic algorithm for expensive optimization problems. Int J Comput Intell Syst 16:169. https://doi.org/10.1007/s44196-023-00346-y
29. Zhang Y-J, Wang Y-F, Yan Y-X, Zhao J, Gao Z-M (2024) Self-adaptive hybrid mutation slime mould algorithm: case studies on UAV path planning, engineering problems, photovoltaic models and infinite impulse response. Alex Eng J 98:364–389. https://doi.org/10.1016/j.aej.2024.04.075
30. Zhong R, Fan Q, Zhang C, Yu J (2024) Hybrid remora crayfish optimization for engineering and wireless sensor network coverage optimization. Clust Comput 4:1–28. https://doi.org/10.1007/s10586-024-04508-1
31. Fisher H (1963) Probabilistic learning combinations of local job-shop scheduling rules. Ind Sched 1963:225–251
32. Cowling P, Kendall G, Soubeiga E (2001) A hyperheuristic approach to scheduling a sales summit. In: Burke E, Erben W (eds) Practice and Theory of Automated Timetabling III. Springer, Berlin, Heidelberg, pp 176–190. https://doi.org/10.1007/3-540-44629-X_11
33. Yan F, Di K (2023) Solving the multi-robot task allocation with functional tasks based on a hyper-heuristic algorithm. Appl Soft Comput 146:110628. https://doi.org/10.1016/j.asoc.2023.110628
34. Zhong R, Yu J (2024) A novel evolutionary status guided hyper-heuristic algorithm for continuous optimization. Clust Comput 8:1–30. https://doi.org/10.1007/s10586-024-04593-2
35. Opara K, Arabas J (2018) Comparison of mutation strategies in differential evolution—a probabilistic perspective. Swarm Evol Comput 39:53–69. https://doi.org/10.1016/j.swevo.2017.12.007
36. Meng Z, Chen Y (2023) Differential evolution with exponential crossover can be also competitive on numerical optimization. Appl Soft Comput 146:110750. https://doi.org/10.1016/j.asoc.2023.110750
37. Ghosh A, Das S, Mallipeddi R, Das AK, Dash SS (2017) A modified differential evolution with distance-based selection for continuous optimization in presence of noise. IEEE Access 5:26944–26964. https://doi.org/10.1109/ACCESS.2017.2773825
38. Zhang Y-J, Wang Y-F, Yan Y-X, Zhao J, Gao Z-M (2022) Lmraoa: an improved arithmetic optimization algorithm with multi-leader and high-speed jumping based on opposition-based learning solving engineering and numerical problems. Alex Eng J 61:12367–12403. https://doi.org/10.1016/j.aej.2022.06.017
39. Van Thieu Nguyen (2020) Opfunu: an open-source python library for optimization benchmark functions. Zenodo. https://doi.org/10.5281/zenodo.3620960
40. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, vol 4, pp 1942–19484 . https://doi.org/10.1109/ICNN.1995.488968
41. Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evol Comput 9(2):159–195. https://doi.org/10.1162/106365601750190398
42. Liang JJ, Qin AK, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 10(3):281–295. https://doi.org/10.1109/TEVC.2005.857610
43. 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(19):9701–9718. https://doi.org/10.1007/s00500-018-3536-8

44. Sallam KM, Elsayed SM, Chakrabortty RK, Ryan MJ (2020) Improved multi-operator differential evolution algorithm for solving unconstrained problems. In: 2020 IEEE Congress on Evolutionary Computation (CEC), pp 1–8 . https://doi.org/10.1109/CEC48606.2020.9185577
45. Wang Z-J, Jian J-R, Zhan Z-H, Li Y, Kwong S, Zhang J (2023) Gene targeting differential evolution: a simple and efficient method for large-scale optimization. IEEE Trans Evol Comput 27(4):964–979. https://doi.org/10.1109/TEVC.2022.3185665
46. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82. https://doi.org/10.1109/4235.585893
47. Ho Y-C, Pepyne D (2002) Simple explanation of the no-free-lunch theorem and its implications. J Optim Theory Appl 115:549–570. https://doi.org/10.1023/A:1021251113462

Authors and Affiliations

Rui Zhong[1] · Shilong Zhang[2] · Jun Yu[3] · Masaharu Munetomo[1]

✉ Rui Zhong
zhongrui@iic.hokudai.ac.jp

Shilong Zhang
f24c130g@mail.cc.niigata-u.ac.jp

Jun Yu
yujun@ie.niigata-u.ac.jp

Masaharu Munetomo
munetomo@iic.hokudai.ac.jp

1 Information Initiative Center, Hokkaido University, Sapporo, Japan

2 Graduate School of Science and Technology, Niigata University, Niigata, Japan

3 Institute of Science and Technology, Niigata University, Niigata, Japan