Hindawi
Computational Intelligence and Neuroscience
Volume 2021, Article ID 8930980, 20 pages
https://doi.org/10.1155/2021/8930980

Research Article

Enhanced Differential Evolution Algorithm with Local Search Based on Hadamard Matrix

Changshou Deng,[1] Xiaogang Dong ,[1,2] Yucheng Tan,[3] and Hu Peng[1]

[1]*School of Electronic Information Engineering, Jiujiang University, Jiujiang 332005, China*
[2]*College of Information Management, Jiangxi University of Finance and Ecomomics, Nanchang 330013, China*
[3]*College of Science, Jiujiang University, Jiujiang 332005, China*

Correspondence should be addressed to Xiaogang Dong; dxg110@aliyun.com

Received 12 August 2021; Accepted 17 September 2021; Published 29 October 2021

Academic Editor: Mario Versaci

Differential evolution (DE) is a robust algorithm of global optimization which has been used for solving many of the real-world applications since it was proposed. However, binomial crossover does not allow for a sufficiently effective search in local space. DE's local search performance is therefore relatively poor. In particular, DE is applied to solve the complex optimization problem. In this case, inefficiency in local research seriously limits its overall performance. To overcome this disadvantage, this paper introduces a new local search scheme based on Hadamard matrix (HLS). The HLS improves the probability of finding the optimal solution through producing multiple offspring in the local space built by the target individual and its descendants. The HLS has been implemented in four classical DE algorithms and jDE, a variant of DE. The experiments are carried out on a set of widely used benchmark functions. For 20 benchmark problems, the four DE schemes using HLS have better results than the corresponding DE schemes, accounting for 80%, 75%, 65%, and 65% respectively. Also, the performance of jDE with HLS is better than that of jDE on 50% test problems. The experimental results and statistical analysis have revealed that HLS could effectively improve the overall performance of DE and jDE.

1. Introduction

Differential evolution (DE), which was proposed by Storn for solving Chebyshev inequality in 1995 [1], is a well-known numerical optimization algorithm. Due to its simple structure, limited number of parameters, an easy implementation, and outstanding optimization performance, DE has drawn great attention of many researchers and engineers since it was proposed. Over the past two decades, DE has been successfully applied to a variety of fields, such as computer vision [2], dynamic economic dispatch [3], engineering design [4], project scheduling [5], artificial neural networks [6], and complex problems inherent to magnetorheological fluids of interest to the automotive industry, in the framework of extended irreversible thermodynamics [7, 8]. Unlike other population-based evolutionary algorithms, the mutation operator in DE utilizes differential information between individuals in the current population. The mechanism gives DE an obvious edge over other evolutionary algorithms. The binomial crossover, however, only produces one offspring in the space constructed by the target individual and its descendant. Therefore, the trial individual is just only one case of many potential solutions, and other potential solutions are ignored. Hence, it is clear that DE's search of the subspace is insufficient. This clearly affected the overall performance of DE.

To fill this gap, we introduced a new scheme of local search based on the Hadamard matrix (HLS) for the sake of improving the overall performance of DE.

The remainder of this paper is organized as follows. Section 2 introduces the basic elements of DE algorithm and reviews the related work. Section 3 presents the details about the Hadamard local search. The experimental results are reported in Section 4, while Section 5 concludes this paper.

2. Background

2.1. Differential Evolution. DE algorithm consists of the following four steps.

2.1.1. Initialization. Initialization is the first step of DE algorithm. It randomly generates a population which contains NP individuals in D-dimensional space. For the ith individual, the jth parameter was initialized by the following formula:

$$x_{ij}^{0} = L_j + \text{Rand}(0,1) \times \left(U_j - L_j\right), \tag{1}$$

where $\text{Rand}(0,1)$ is a uniformly distributed random number within the range $[0, 1]$ and L_j and U_j are the lower and upper bounds of the dimensional spaces, $i \in [1, \text{NP}]$, $j \in [1, D]$.

2.1.2. Mutation Operator. Following initialization, the mutation operator was applied to each target individual x_i^G, thus generating a mutant v_i^G. In view of the important implications that the mutation operators have on the ability of DE's global search, many researchers focus on the work of improving them. Six efficient and widely used operators [9] are listed below.

(i) DE/rand/1/:

$$v_i^G = x_{r1}^G + F \times \left(x_{r2}^G - x_{r3}^G\right). \tag{2}$$

(ii) DE/best/1:

$$v_i^G = x_{best}^G + F \times \left(x_{r1}^G - x_{r2}^G\right). \tag{3}$$

(iii) DE/rand/2:

$$v_i^G = x_{r1}^G + F \times \left(x_{r2}^G - x_{r3}^G\right) + F \times \left(x_{r4}^G - x_{r5}^G\right). \tag{4}$$

(iv) DE/best/2:

$$v_i^G = x_{best}^G + F \times \left(x_{r1}^G - x_{r2}^G\right) + F \times \left(x_{r3}^G - x_{r4}^G\right). \tag{5}$$

(v) DE/rand-to-best/1:

$$v_i^G = x_{r1}^G + F \times \left(x_{best}^G - x_{r1}^G\right) + F \times \left(x_{r2}^G - x_{r3}^G\right). \tag{6}$$

(vi) DE/current-to-best/1:

$$v_i^G = x_i^G + F \times \left(x_{best}^G - x_i^G\right) + F \times \left(x_{r1}^G - x_{r2}^G\right). \tag{7}$$

2.1.3. Crossover Operator. Crossover operator randomly combines the genes of the target and its mutant to produce a new offspring. The binomial crossover is the most commonly used method. It is expressed as follows:

$$u_{ij}^{G+1} = \begin{cases} v_{ij}^G, & \text{if rand}(0,1) < \text{CR or } j = \text{rand}(i), \\ x_{ij}^G, & \text{otherwise,} \end{cases} \tag{8}$$

where $\text{rand}(0,1)$ is a uniform random number in $[0, 1]$ and CR is the crossover probability.

2.1.4. Selection Operator. The DE selection operator is a greedy strategy. From the target individual and its offspring, the one with the better fitness value will enter the next generation. The selection operator is shown in the following formula:

$$x_{i+1}^{G+1} = \begin{cases} u_i^{G+1}, & \text{if } f\left(u_i^{G+1}\right) < f\left(x_i^G\right), \\ x_i^G, & \text{otherwise.} \end{cases} \tag{9}$$

3. Related Work

Although DE has been successfully applied in many fields, it still needs to improve the performance of the algorithm in many other fields. Thus, several improved versions of DE were proposed by the researchers. These works can be divided into the following four categories.

3.1. Improvement of the Mutation Operator. Ramadas et al. proposed a ReDE algorithm introducing a revised mutation strategy for DE [10]. Mohamed and Almazyad et al. proposed an ANDE algorithm which introduced a new triangular mutation for DE [11]. Gong and Cai proposed a classification-based mutation strategy for DE [12], in which some of the parents are selected proportionally based on their classification in the current population. Peng et al. proposed an improvement in differential evolution, which was named RNDE. RNDE used a new mutation operator, DE/neighbor/1, to balance the exploration and exploitation ability of DE process [13].

3.2. The New Scheme of Self-Adapting Parameters. Brest et al. presented a new approach to the self-adaptive control parameter of DE [14]. Qin et al. proposed a self-adapting DE algorithm, in which the strategies for generating test vectors and their associated parameter values are progressively self-adapting, taking advantage of their previous experiences in the generation of promising solutions [15]. Zhu et al. proposed an adaptive population adaptation scheme (APTS) for DE to dynamically adjust the size of the population [16].

3.3. Hybrid DE. Wang et al. proposed an orthogonal crossover (OX) operator, which is based on orthogonal design and can make a systematic and rational search in a region defined by the parent solutions [17]. Rahnamayan proposed an opposition-based learning DE (ODE). ODE employs opposition-based learning (OBL) for population initialization and for generation jumping [18]. Sun et al. proposed a new algorithm, named DE/EDA [19], which combined DE with estimation of distribution algorithm. Peng et al. proposed a novel DE variant with commensal learning and uniform local search, named CUDE. The biggest contribution of CUDE is to enhance the local space search performance of DE by using uniform experimental design [20].

3.4. The New Methods of Local Search Strategy. Local search can effectively improve the performance of evolutionary algorithms. For example, fittest individual refinement (FIR) was proposed by Noman et al. [21]. In the FIR, the search space around the best individual is explored greedily in each generation. Later, two implementations (DEfirDE and DEfirSPX) of FIR were proposed. The results of the experiments show that both schemes speed up DE for a set of well-known test functions, especially for high dimensions, and they are better than the other two well-known variants of DE. A crossover-based adaptive local search (LS) operator was proposed to enhance the performance of the standard DE algorithm [22]. The new algorithm mainly improved the local search by adaptively adjusting the length of the search using a hill-climbing heuristic. Trigonometric local search (TLS) and interpolated local search (ILS) were proposed in [23]. Combining these two local search strategies, two new variants of DE algorithms (DETLS and DEILS) were implemented. The new scheme improved the performance of DE in terms of the quality of solution without compromising on the convergence rate. A restart differential evolution algorithm with local search mutation (RDEL) was proposed in [24]. In RDEL, a novel local mutation rule based on the positions of the best and the worst individuals among the entire population of a particular generation is introduced. Also, it was combined with the basic mutation rule through a linear decreasing function. The new local mutation effectively enhanced the local search tendency of the basic DE and accelerated the convergence speed. An adaptive local search for dynamically balancing the degree of global search (GS) and local search (LS) was proposed in [25]. In this adaptive local search, if LS performs better than GS, it will increase its preference for utilization. If LS does not perform well, it will reduce its preferences for LS. The performance of the new algorithm for hybridization of the adaptive LS scheme is evaluated by using 10 benchmark problems, and the results prove the effectiveness of the algorithm. An enhanced differential evolution with random local search (named DERLS) was proposed in [26]. The advantage of using random local search in DERLS is to make a small random "jump" to a more promising area in the solution space, thus avoiding the local optimum. It is very simple, fast in calculation, and more efficient for multimode functions than classical DE. Peng et al. proposed a heterozygous differential evolution with Taguchi local search, which effectively enhances the local search performance of DE [27].

Inspired by local search methods, this paper uses the Hadamard matrix to construct the local search for DE.

4. Hadamard Local Search for Differential Evolution

4.1. Motivation. A crossover operator is a recombination operator that generates an offspring around the parents. Therefore, a local search strategy can be regarded as a moving operator [22]. In traditional DE, the binomial crossover operator (the most commonly used crossover operator) only generates and evaluates one single trial vector, which is a vertex of the hyper-rectangle defined by the mutant vector and the target vector [17]. That is to say, only one of many combinations is obtained. As a result, the search for space around parents is inadequate. On the other hand, if all vertices of the super-rectangle defined by the mutation vector and the target vector are checked, a lot of computation is needed. In this paper, a compromise method is used to construct a local search operator by using Hadamard matrix to search several vertices.

4.2. Local Search Based on Hadamard Matrix. A Hadamard matrix is a square matrix whose entries are either +1 or −1 and whose rows are mutually orthogonal. For example, a fourth-order Hadamard matrix (H4) is represented as follows:

$$H4 = \begin{cases} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{cases}. \tag{10}$$

In geometric terms, this means that each pair of rows in the Hadamard matrix represents two vertical vectors [28]. Except for the first line, half of the elements in each row contain +1 and the other half −1. This feature allows us to construct a new local search strategy based on Hadamard matrix (HLS).

Based on the above characteristics of Hadamard matrix, we propose a new local search operator. We take the fourth-order Hadamard matrix H4 as an example. Since the scale of the optimization problem d is generally greater than 4, it is impossible to create the crossover operator directly on H4. To use H4, the d-dimensional space of the optimization problem needs to be divided into several subspaces. For example, if the dimension size of the optimization problem is 10, the interval [1, 10] should be randomly divided into four subintervals, and each interval corresponds to an element of H4. Figure 1 shows an example of HLS.

Algorithm 1 presents the steps of HLS. With HLS, v1 and x1 will produce four offspring, among which v1 is the mutant individual of x1. Due to the characteristics of Hadamard matrix, these four offspring are four random combinations of v1 and x1. Compared with the traditional crossover operator, HLS can search more completely in local space and find better solution more easily. Therefore, HLS will improve the search performance when classical crossover cannot find a better solution.

4.3. New Framework of DE with HLS. There are two common ways to use the local search operator in DE algorithm. One is to replace the original crossover operator with local search operator, just as OXDE [17] did. Another method is to select an individual to perform a local search independently during evolution. In essence, for local search, the goal is to find better offspring than the target individual. Therefore, when the crossover operator can produce better offspring, there is no need for local search. In the process of evolution, the success

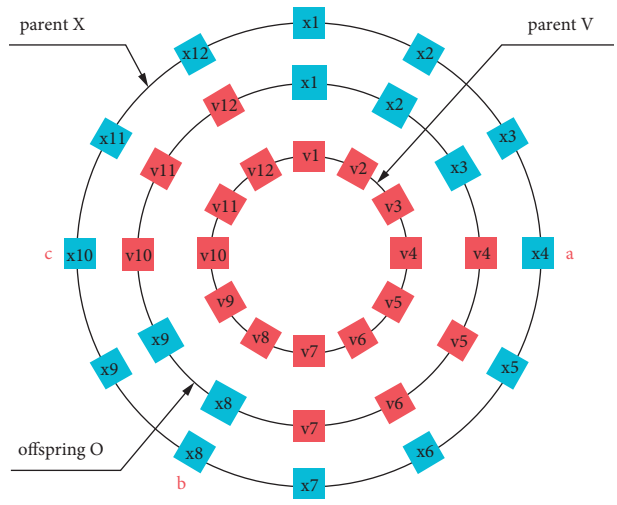

Figure 1: HLS example.

Input: $\mathbf{v}_1, \mathbf{x}_1$
(1) Divide $\mathbf{v}_1$, $\mathbf{x}_1$ into four subvectors randomly
(2) Read the information of H4
(3) for $i = 1:4$
(4) for $j = 1:4$
(5) if $H4[i][j] == 1$
(6) $offspring[i][j] = v_1[i][j]$
(7) else
(8) $offspring[i][j] = \mathbf{x}_1[i][j]$
(9) end
(10) end
(11) end
Output: offspring

Algorithm 1: The steps of HLS.

rate of individual renewal is relatively fast in the early stage of evolution, while in the late stage of evolution, the success rate of individual renewal is very slow and even tends to zero.

In the following, we will use three representative functions: Quartic with Noise, Penalized1, and Shift Ackley, to give the convergence process. In the experiment, the population size was set to 30. Figure 2 shows a successful single update in solving these three functions. As can be seen from Figure 2, almost all individuals cannot be successfully updated at the later stage.

To improve the success rate of DE during the later stage, a new framework of DE with HLS was proposed. HLS operator will not affect the performance of the algorithm in the early stage of evolution, but it can effectively avoid premature convergence in the late stage of evolution. The new framework is presented in Algorithm 2. To avoid consuming too much evaluation resources, the framework uses HLS with the specified probability p (see Algorithm 2, Step 13). During our experiments, P is set to 0.1. In addition, to make full use of the information of the evolution process, HLS uses a mutation vector to construct the local search (see Algorithm 2, Step 14).

4.4. Computational Complexity. For DE with HLS, computational complexity is determined by the number of times the three operators of DE and HLS are executed. Also, its execution time is proportional to the search space dimension. Consequently, DE with HLS has a worst case time complexity on the order of, where is size of population, is the maximum iteration number, and P is the user-defined probability to execute HLS. It is easy to deduce that the time complexity of DE with HLS is. In [9], the time complexity of DE is $O(D \times NP \times GMAX)$. Therefore, the time complexity of DE with HLS is the same as that of DE.

5. Experimental Study

5.1. Test Suit. In our experiments, twenty widely used test functions [29, 30] are used to evaluate DEHLS. These functions include 7 unimodal functions (f1~f7), 6 multimodal functions (f8~f13) and 7 shifted functions (f14~f20). The information of each function is listed in Table 1.

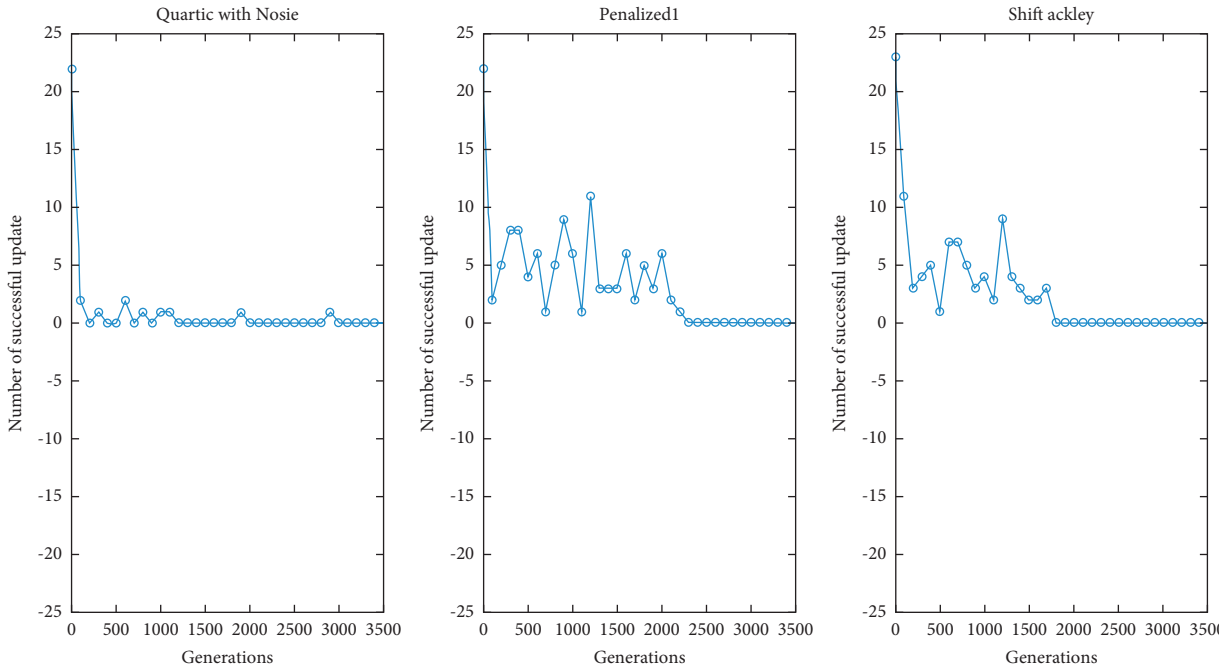

Figure 2: Successful update of population with generations.

Input: D, NP, F, CR, P, MaxFEs
Randomly initialize population *pop*
Evaluate the *pop* by objective function obj_func, get *fit*
FEs = NP
while FEs < MaxFEs do
  for $i = 1$: NP do
    Execute the mutation operator to generate a mutation vector v_i
    Execute the crossover operator to generate a trial vector u_i
    Evaluate the trial vector u_i to get fit_u_i
    FEs = FEs + 1
    if fit_u_i < fit(i)
      pop(i,:) = u_i
      fit(i) = fit_u_i
    else
      if rand < P
        offspring = HLS(v_i, pop(i, :))
        ovalue = obj_func(offspring)
        FEs = FEs + 4;
        [min_value, min_index] = min(ovalue)
        if min_value < fit(i)
          pop(i, :) = offspring(min_index)
          fit(i) = min_value
        end
      end
    end
  end
end
Output: optimal solution

Algorithm 2: New framework of DE with HLS.

5.2. Experimental Setting. In this paper, three sets of experiments were conducted. The first set of experiments combined the HLS with four classical operators of DE algorithm to verify the validity of the HLS. The second set of experiments analysed the influence of the size growth of Hadamard matrix. The third set of experiments tested the

Table 1: Test suit.

Type	Function	Name	Dimension	$F(x^*)$	Range
Unimodal functions	f1	Sphere	30	0	[−100, 100]
	f2	Schwefel 2.22	30	0	[−10, 10]
	f3	Schwefel 1.2	30	0	[−100, 100]
	f4	Schwefel 2.21	30	0	[−100, 100]
	f5	Rosenbrock	30	0	[−30, 30]
	f6	Step	30	0	[−100, 100]
	f7	Quartic with Noise	30	0	[−1.28, 1.28]
Multimodal functions	f8	Schwefel 2.26	30	−12569.5	[−500, 500]
	f9	Rastrigin	30	0	[−5.12, 5.12]
	f10	Ackley	30	0	[−32, 32]
	f11	Griewank	30	0	[−600, 600]
	f12	Penalized1	30	0	[−50, 50]
	f13	Penalized2	30	0	[−50, 50]
Shifted functions	f14	Shift sphere	30	0	[−100, 100]
	f15	Shift schwefel1.2	30	0	[−100, 100]
	f16	Shift schwefel1.2 with noise	30	0	[−100, 100]
	f17	Shift Griewank	30	0	[−600, 600]
	f18	Shift Ackley	30	0	[−32, 32]
	f19	Shift penalized1	30	0	[−50, 50]
	f20	Shift penalized2	30	0	[−50, 50]

performance of jDE [14] with HLS and compared it with four other state-of-the-art DE variants (jDE, Sade, ODE, and OXDE).

In all experiments, the dimension size (D) of the test problem is set to 30, the termination criterion is $D * 10000$, and 30 independent runs were conducted. In the first set of experiments, the parameters F and CR are set to 0.9, and the parameter NP is set to NP = D. In the third set of experiments, the parameters of the four compared algorithms are set according to the original literature.

5.3. Quality of the HLS. In this section, four classical schemes of DE, namely, DE/rand/1, DE/best/1, DE/rand-to-best/1, and DE/current-to-best/1, are used in the experiment to evaluate the quality of HLS. To distinguish them, these four diagrams are, respectively, named DE1, DE2, DE3, and DE4. The HLS operator has been integrated into each of the four schemes above, under the names DE1HLS, DE2HLS, DE3HLS, and DE4HLS. To guarantee the fairness of the experiment, the same parameters are defined for all the algorithms.

Table 2 presents the results of the experiment. "Average error" and "standard error" represent, respectively, the average value of error of the function and the standard deviation obtained by all the algorithms. The results of the Wilcoxon rank sum test are marked "-," "+," and "≈" in the table to indicate that the performance of DE without HLS is lower, better, and similar to that of DE with HLS. In addition, Figures 3–10 show the evolutionary processes of the two competitors.

From the results of Table 2, we can see that the HLS can greatly improve the performance of the four classic DE schemes. For the 20 benchmark problems, the number of functions with better results from the four DE schemes with HLS than the DE schemes was 16, 15, 13, and 13, respectively. This improvement suggests that HLS promotes the performance of the majority of test functions. Also, the number of functions with worse results from the four DE schemes with HLS than the DE schemes was 3, 4, 7, and 7, respectively. These functions are mainly unimodal functions. One of the reasons is that the solutions of the unimodal functions are easy to obtain, but HLS has increased the number of evaluations.

Therefore, we concluded that HLS can effectively improve the performance of DE, especially with multimode and shifted/offset functions.

5.4. Effect of the Size of Hadamard Matrix Dimension. In this section, the effect of dimension size of Hadamard matrix was analysed. Since the dimension size of the matrix is a multiple of 2 or 4, to reduce the computing burden, the dimension sizes of Hadamard matrices 4, 8, and 16 are used for experiments (written as HLS-4, HLS-8, and HLS-16). On the other hand, as can be seen from the analysis in Section 4.3, DE1HLS (DE/rand/1 + HLS) is the best one among the four schemes. Thus, DE1HLS is used in the experiment.

Table 3 summarizes the results of the experiment, in which "†" represents the best solution for these three solutions. The statistical results are in the last row of the table.

As can be seen from the results in Table 3, when the Hadamard matrix dimension is 4, the optimal solutions of 16 problems are better than the other two dimension sizes (8 and 16). Table 4 further gives the average ranking of the Hadamard matrices in three different dimensions (based on the Friedman test). The best average ranking is HLS_4. Thus, the following experiment will use HLS_4.

Table 2: Experimental results of four classical operators of DE with HLS for all test functions and comparison without HLS.

Fun	DE/rand/1/bin	DE/rand/1/bin + HLS	DE/best/1/bin	DE/best/1/bin + HLS	DE/rand-to-best/1/bin	DE/rand-to-best/1/bin + HLS	DE/current-to-best/1/bin	DE/current-to-best/1/bin + HLS
	Mean error ± Std error	Mean error ± Std error	Mean error ± Std error	Mean error ± Std error	Mean error ± Std error	Mean err ± Std err	Mean err ± Std err	Mean err ± Std err
f1	$3.47e-18 \pm 7.66e-18-$	$1.12E-112 \pm 6.03E-112$	$1.94e-12 \pm 1.06e-11-$	$2.94E-20 \pm 1.61E-19$	$3.41e-106 \pm 1.85e-105+$	$1.70E-49 \pm 9.31E-49$	$4.90e-108 \pm 1.89e-107+$	$2.90E-17 \pm 1.59E-16$
f2	$6.46e-11 \pm 9.73e-11-$	$2.30E-58 \pm 1.22E-57$	$1.90e-04 \pm 1.04e-03\approx$	$7.53E-04 \pm 4.09E-03$	$1.83e-59 \pm 7.68e-59+$	$1.36E-06 \pm 7.47E-06$	$3.55e-61 \pm 1.27e-60+$	$3.96E-12 \pm 2.17E-11$
f3	$3.46e-02 \pm 3.26e-02+$	$1.65E+00 \pm 2.73E+00$	$1.55e-21 \pm 4.76e-21+$	$2.10E-15 \pm 7.15E-15$	$2.86e-24 \pm 7.23e-24+$	$3.93E-17 \pm 6.83E-17$	$6.53e-25 \pm 1.50e-24+$	$2.39E+00 \pm 1.31E+01$
f4	$1.53e+01 \pm 7.28e+00-$	$1.73E-165 \pm 0.00E+00$	$3.60e+01 \pm 7.31e+00-$	$2.70E-03 \pm 1.45E-02$	$2.23e+01 \pm 5.68e+00-$	$3.72E-175 \pm 0.00E+00$	$2.63e+01 \pm 7.30e+00-$	$5.02E-05 \pm 2.12E-04$
f5	$8.83e+00 \pm 3.21e+00\approx$	$1.19E+01 \pm 1.38E+01$	$9.30e-01 \pm 1.71e+00+$	$2.27E+01 \pm 1.16E+01$	$1.20e+00 \pm 1.86e+00+$	$2.55E+01 \pm 8.66E+00$	$1.46e+00 \pm 1.95e+00+$	$2.27E+01 \pm 1.16E+01$
f6	$2.67e-01 \pm 7.85e-01-$	$0.00E+00 \pm 0.00E+00$	$2.27e+02 \pm 5.80e+02-$	$0.00E+00 \pm 0.00E+00$	$3.03e+01 \pm 7.46e+01-$	$0.00E+00 \pm 0.00E+00$	$1.85e+01 \pm 2.45e+01-$	$0.00E+00 \pm 0.00E+00$
f7	$1.63e-02 \pm 5.98e-03-$	$3.54E-03 \pm 4.00E-03$	$1.11e-02 \pm 7.48e-03-$	$2.30E-03 \pm 4.55E-03$	$8.50e-03 \pm 4.20e-03-$	$1.70E-03 \pm 3.10E-03$	$8.40e-03 \pm 4.20e-03-$	$1.30E-03 \pm 1.36E-03$
f8	$1.32e+04 \pm 2.47e+02-$	$1.26E+04 \pm 0.00E+00$	$1.55e+04 \pm 6.80e+02-$	$1.38E+04 \pm 1.77E+03$	$1.49e+04 \pm 5.66e+02-$	$1.32E+04 \pm 1.29E+03$	$1.54e+04 \pm 6.62e+02-$	$1.30E+04 \pm 1.13E+03$
f9	$2.41e+01 \pm 7.72e+00-$	$0.00E+00 \pm 0.00E+00$	$6.60e+01 \pm 2.03e+01-$	$1.49E+01 \pm 2.40E+01$	$3.66e+01 \pm 8.98e+00-$	$2.78E+00 \pm 8.54E+00$	$4.09e+01 \pm 1.12e+01-$	$6.41E+00 \pm 1.29E+01$
f10	$3.68e-10 \pm 3.02e-10-$	$4.74E-16 \pm 1.23E-15$	$4.78e+00 \pm 2.44e+00-$	$3.46E-01 \pm 1.51E+00$	$2.34e+00 \pm 1.35e+00-$	$3.04E-14 \pm 1.59E-13$	$2.78e+00 \pm 2.04e+00-$	$2.01E-15 \pm 2.02E-15$
f11	$3.37e-03 \pm 8.55e-03-$	$0.00E+00 \pm 0.00E+00$	$5.09e-02 \pm 7.59e-02-$	$1.58E-02 \pm 6.72E-02$	$2.55e-02 \pm 3.08e-02-$	$3.11E-03 \pm 1.08E-02$	$1.55e-02 \pm 1.95e-02-$	$5.75E-04 \pm 2.21E-03$
f12	$4.53e-02 \pm 2.29e-01-$	$1.57E-32 \pm 1.91E-34$	$1.01e+00 \pm 2.15e+00-$	$3.46E-03 \pm 1.89E-02$	$5.00e-01 \pm 8.89e-01-$	$1.73E-02 \pm 7.74E-02$	$7.25e-01 \pm 1.00e+00-$	$1.68E-32 \pm 2.47E-33$
f13	$7.32e-04 \pm 2.79e-03-$	$1.35E-32 \pm 5.57E-48$	$7.44e-01 \pm 1.70e+00-$	$3.66E-04 \pm 2.01E-03$	$5.65e-01 \pm 1.49e+00-$	$4.01E-32 \pm 8.12E-32$	$3.61e-01 \pm 9.67e-01-$	$3.66E-04 \pm 2.01E-03$
f14	$1.65e-18 \pm 4.20e-18-$	$0.00E+00 \pm 0.00E+00$	$8.77e-22 \pm 4.80e-21+$	$1.62E+01 \pm 8.89E+01$	$5.07e-30 \pm 9.73e-30+$	$4.80E-05 \pm 2.63E-04$	$3.57e-30 \pm 2.69e-30+$	$4.16E-17 \pm 2.27E-16$
f15	$3.73e-02 \pm 3.54e-02+$	$1.16E+00 \pm 1.75E+00$	$5.48e-22 \pm 1.45e-21+$	$1.05E-15 \pm 1.88E-15$	$5.30e-24 \pm 2.06e-23+$	$1.12E-06 \pm 6.14E-06$	$8.67e-25 \pm 2.10e-24+$	$7.27E-17 \pm 1.80E-16$
f16	$3.73e-02 \pm 4.47e-02+$	$2.76E-01 \pm 5.10E-01$	$9.76e-03 \pm 4.36e-02-$	$1.42E-05 \pm 4.99E-05$	$3.02e-07 \pm 1.17e-06+$	$3.75E+00 \pm 2.05E+01$	$2.75e-10 \pm 1.34e-09+$	$8.68E-07 \pm 4.75E-06$
f17	$1.81e-03 \pm 4.92e-03-$	$0.00E+00 \pm 0.00E+00$	$4.13e-02 \pm 4.73e-02-$	$7.75E-03 \pm 4.24E-02$	$1.65e-02 \pm 1.65e-02-$	$2.47E-04 \pm 1.35E-03$	$3.13e-02 \pm 3.18e-02-$	$0.00E+00 \pm 0.00E+00$
f18	$3.13e-10 \pm 2.45e-10-$	$3.55E-16 \pm 1.08E-15$	$3.25e+00 \pm 1.95e+00-$	$1.78E-03 \pm 9.72E-03$	$2.31e+00 \pm 1.19e+00-$	$1.18E-15 \pm 1.94E-15$	$2.62e+00 \pm 9.67e-01-$	$1.66E-15 \pm 1.80E-15$
f19	$6.91e-03 \pm 3.79e-02-$	$1.57E-32 \pm 1.77E-34$	$9.70e-01 \pm 1.87e+00-$	$1.71E-32 \pm 3.20E-33$	$4.53e-01 \pm 6.95e-01-$	$1.66E-32 \pm 1.91E-33$	$6.54e-01 \pm 9.23e-01-$	$3.46E-03 \pm 1.89E-02$
f20	$1.19e-17 \pm 4.41e-17-$	$1.35E-32 \pm 5.57E-48$	$1.73e+04 \pm 9.47e+04-$	$1.96E-31 \pm 4.78E-31$	$2.38e-01 \pm 5.94e-01-$	$6.13E-32 \pm 2.62E-31$	$4.67e-01 \pm 1.63e+00-$	$6.79E-32 \pm 2.55E-31$
w/ b/s	16/3/1		15/4/1		13/7/0		13/7/0	

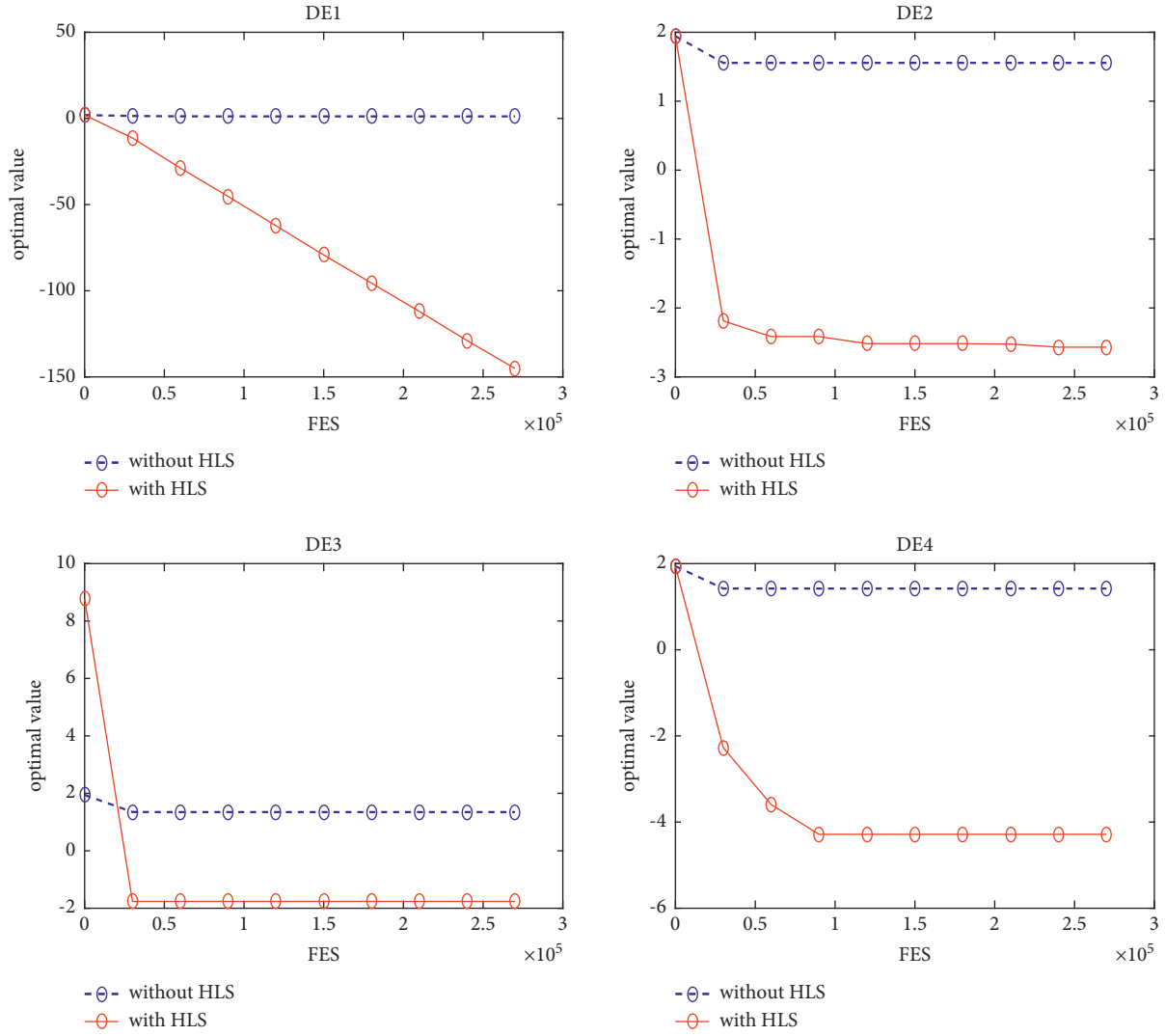

Figure 3: f4 convergence curves.

5.5. Implementation in jDE. In this section, the proposed HLS is implemented in jDE [14], which is a very powerful state-of-the-art DE variant. jDEHLS is compared with four state-of-the-art DE variants (jDE, SaDE, ODE, and OXDE). The experimental results and Wilcoxon's rank sum test are summarized in Table 5.

Compared with jDE, jDEHLS is superior to jDE on 10 functions and similar to jDE on 10 functions. Compared with SaDE, jDEHLS is superior to SaDE on 10 functions, but inferior to SaDE on 3 functions and similar to SaDE on 7 functions. Compared with ODE, jDEHLS is superior to ODE on 13 functions, but inferior to ODE on 1 function and similar to ODE on 6 functions. Compared with OXDE, jDEHLS is superior to OXDE on 17 functions, but inferior to OXDE on 1 function and similar to OXDE on 2 functions.

In short, HLS can improve the performance of jDE. On 20 test functions, the performance of jDEHLS is the best one among the five methods.

To judge whether the results of the five methods differ in a statistically significant way, a nonparametric statistical test called Friedman test is conducted. The test results are presented in Table 6. As shown in Table 6, the average ranking values for these five algorithms can be sorted in the following order: jDEHLS, jDE, ODE, SaDE, and OXDE.

In addition, the multiproblem Wilcoxon's test was conducted to check the behaviour of the six algorithms. The results are summarized in Table 6. We can find that $R+$ values are higher than R- in all cases, This suggests that jDEHLS is markedly superior to OXDE, SaDE, ODE, and jDE.

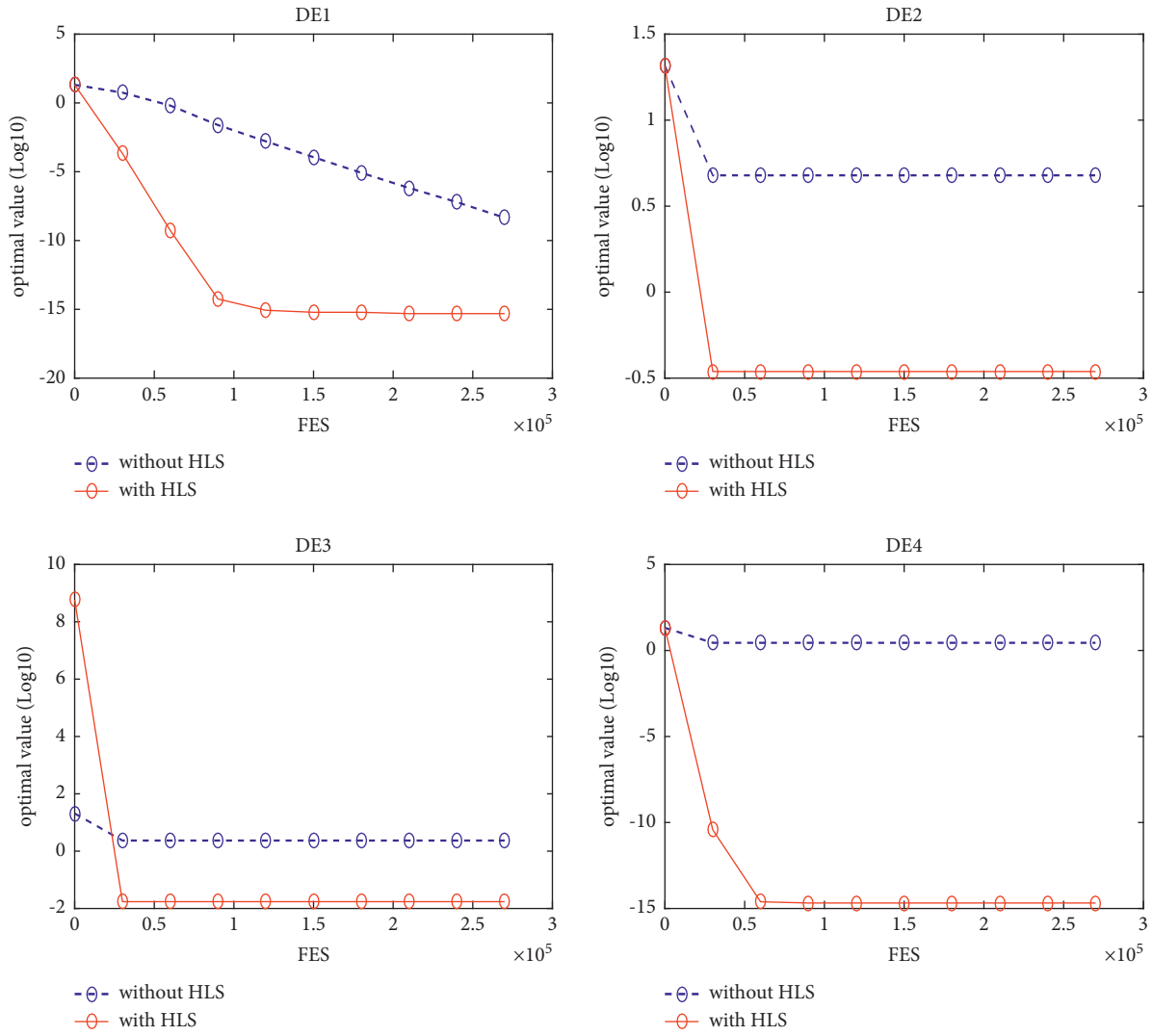

Figure 4: f10 convergence curves.

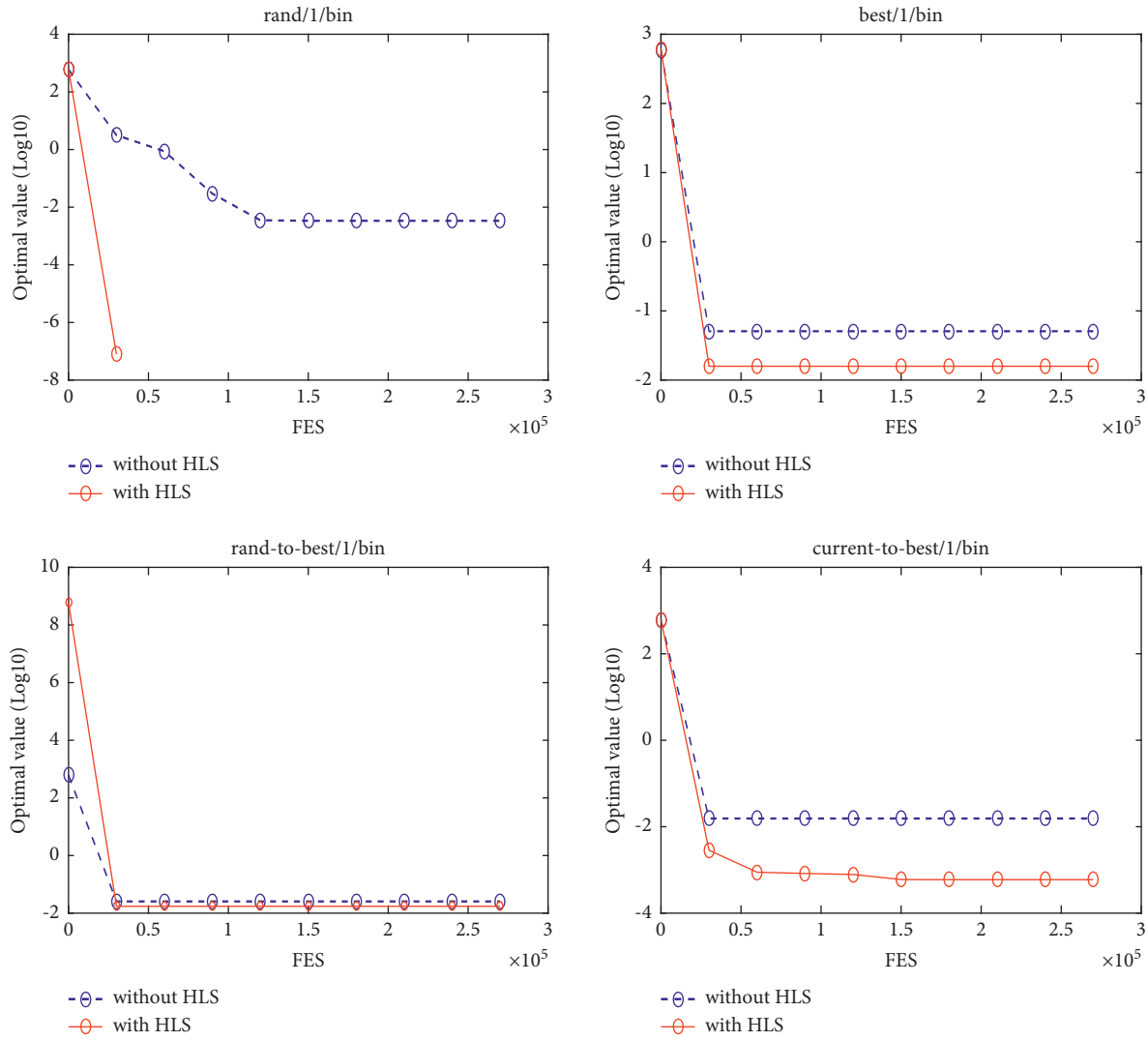

Figure 5: f11 convergence curves.

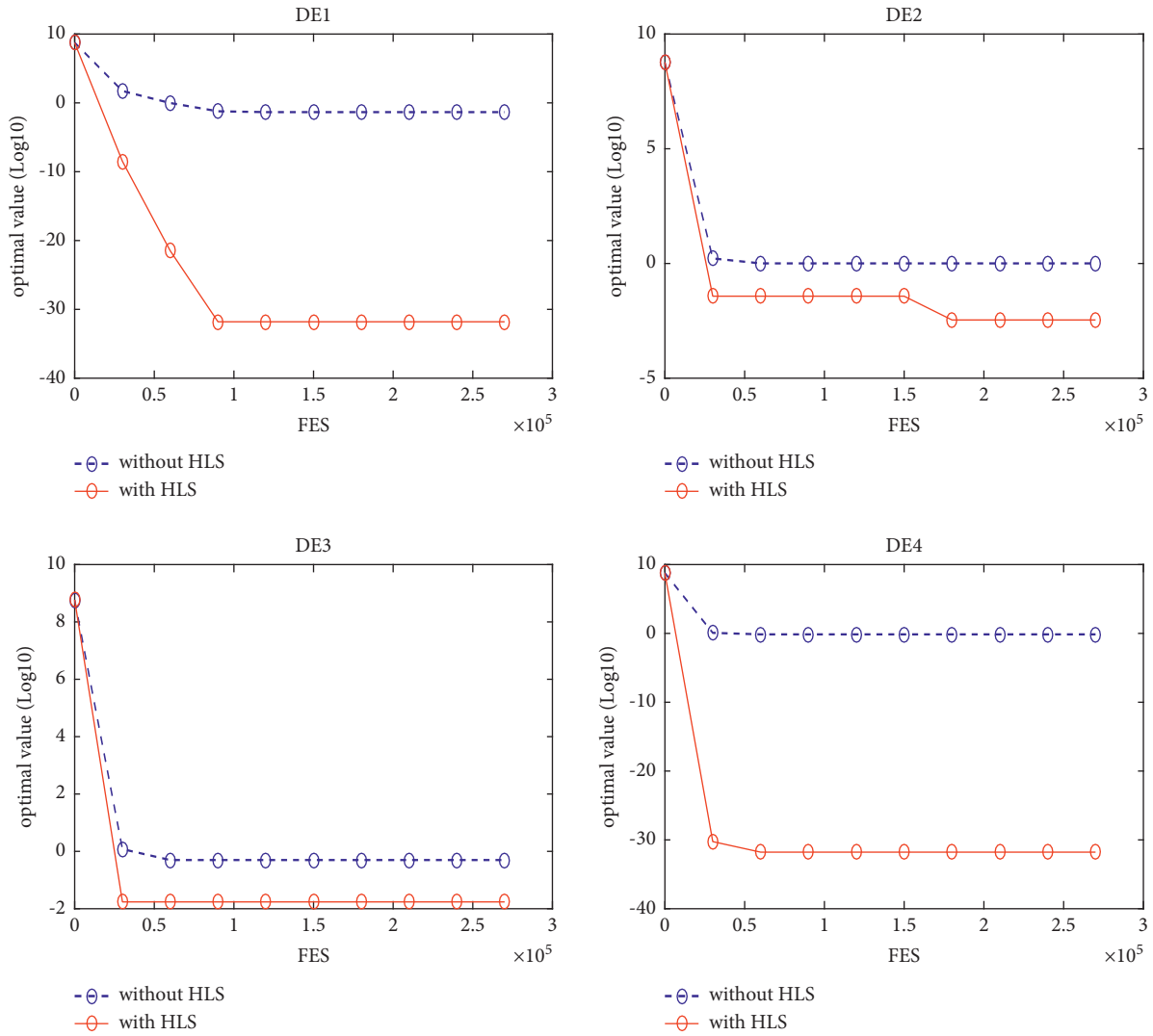

Figure 6: f12 convergence curves.

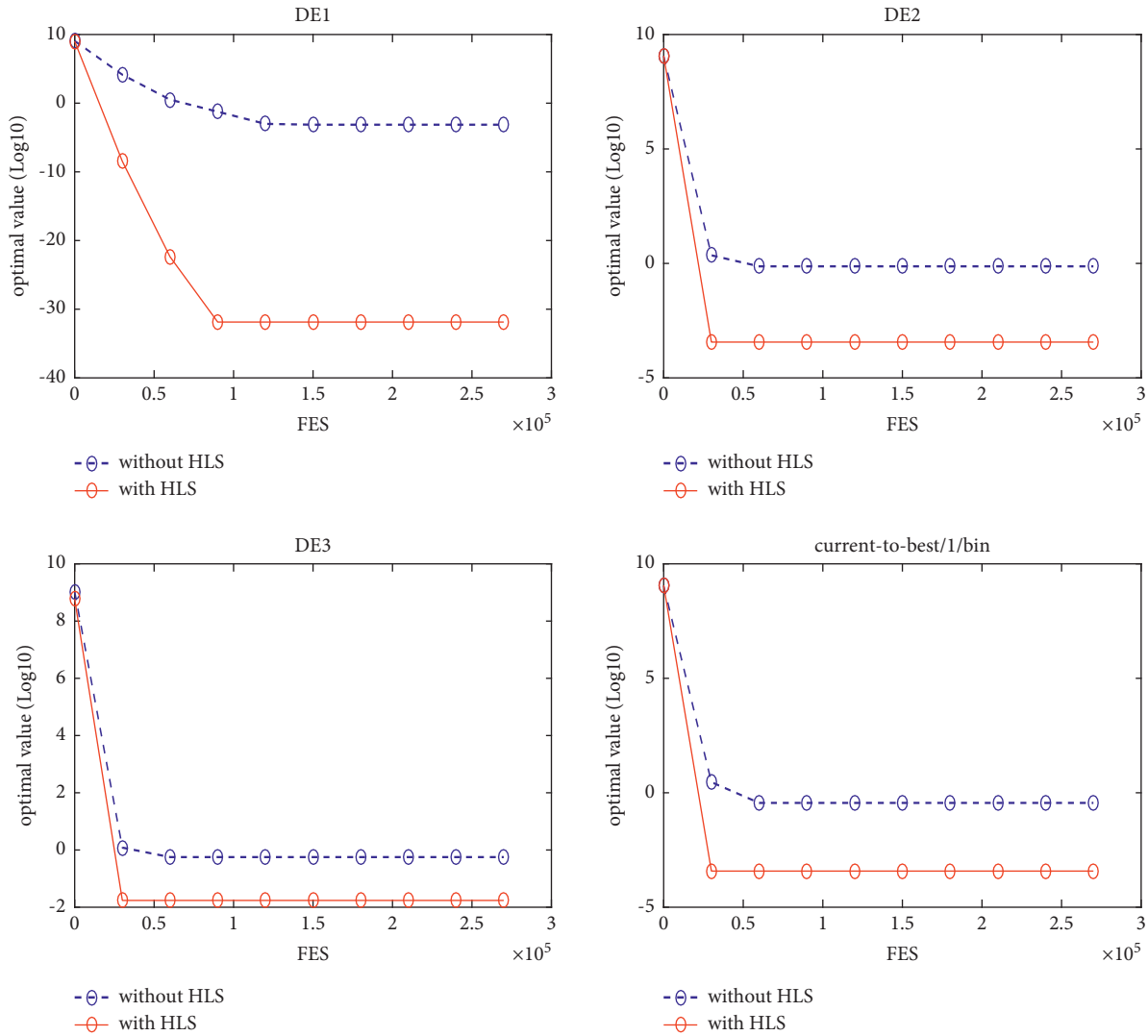

Figure 7: f13 convergence curves.

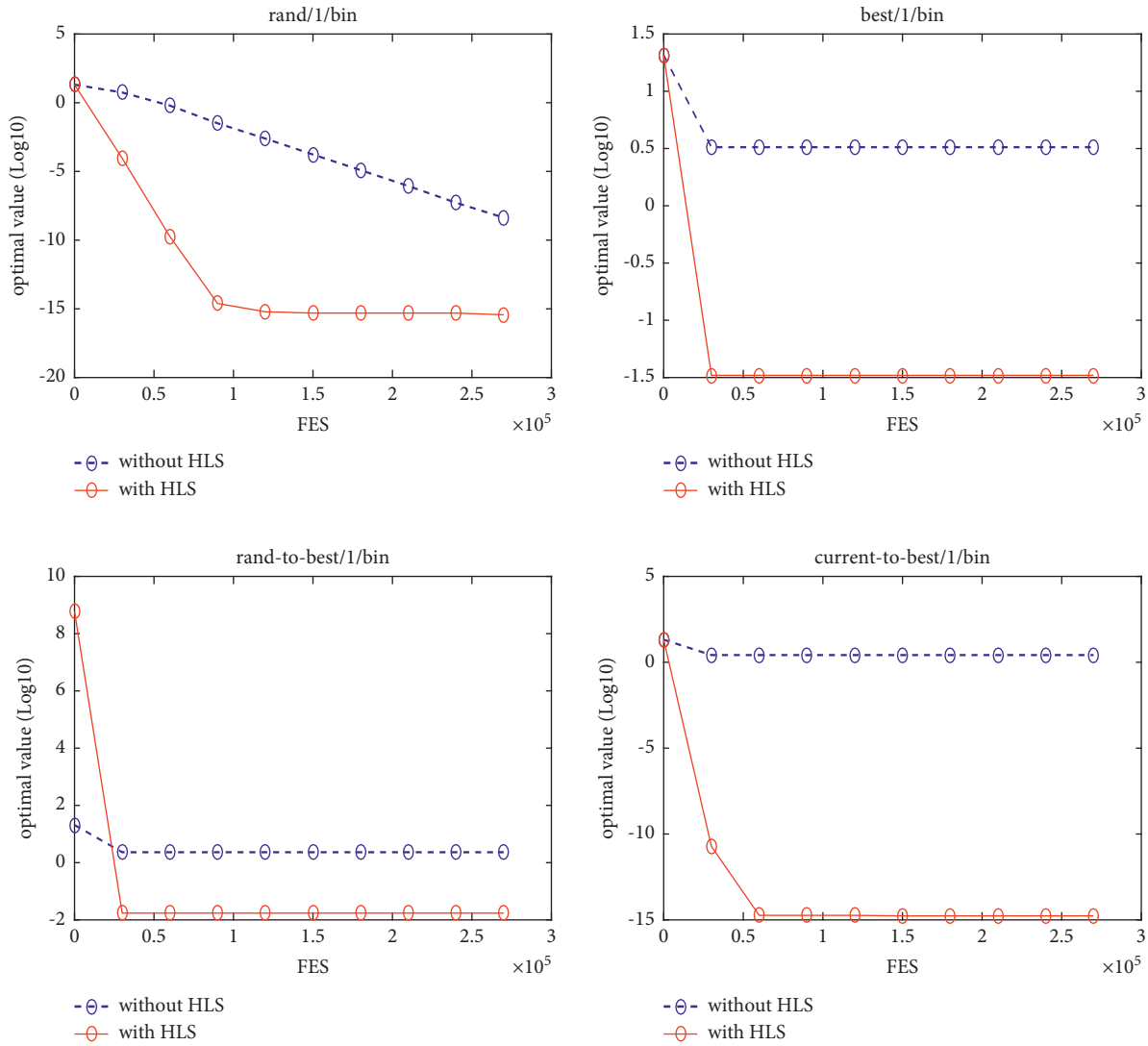

Figure 8: f18 convergence curves.

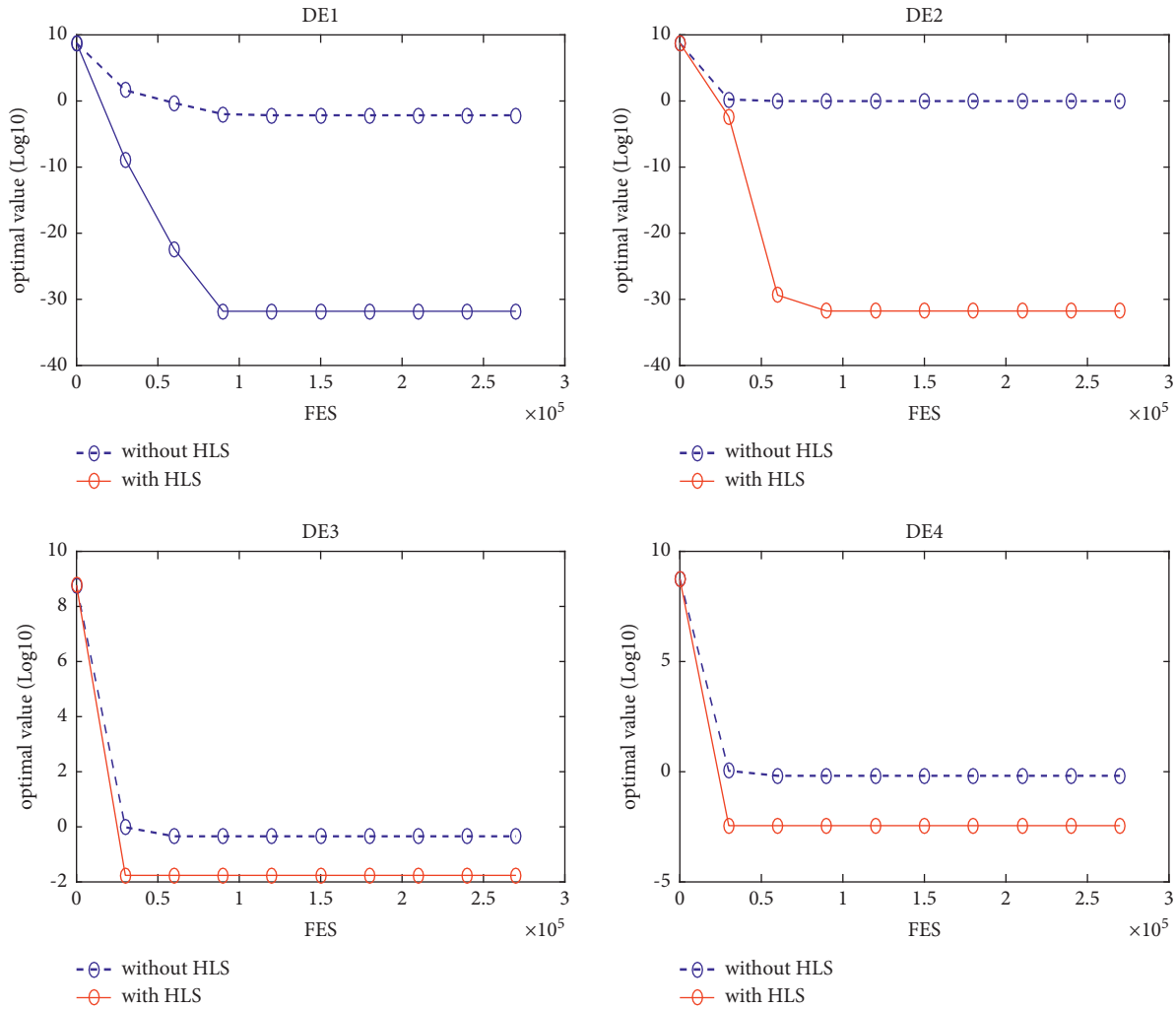

Figure 9: f19 convergence curves.

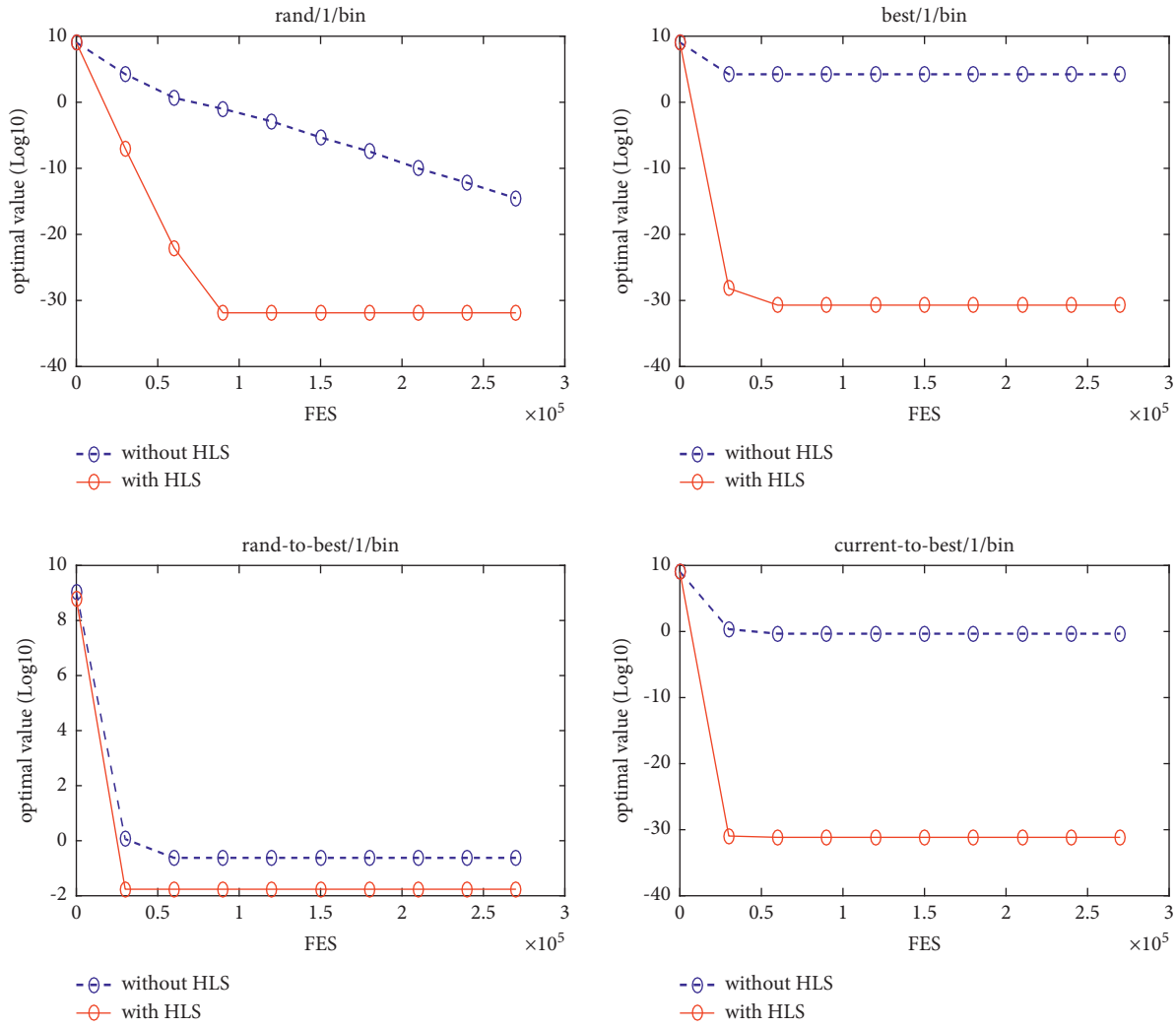

Figure 10: f20 convergence curves.

Table 3: Experimental results of HLS with three sizes of Hadamard matrix.

Fun	DE/rand/1/bin + HLS_4	DE/rand/1/bin + HLS_8	DE/rand/1/bin + HLS_16
f1	$1.12E-112 \pm 6.03E-112$†	$7.60E-84 \pm 4.16E-83$	$3.12E-59 \pm 1.63E-58$
f2	$2.30E-58 \pm 1.22E-57$†	$2.81E-45 \pm 1.07E-44$	$1.15E-30 \pm 6.27E-30$
f3	$1.65E+00 \pm 2.73E+00$†	$1.61E+01 \pm 1.58E+01$	$1.95E+02 \pm 2.89E+02$
f4	$1.73E-165 \pm 0.00E+00$†	$6.10E-129 \pm 3.31E-128$	$1.58E-88 \pm 7.30E-88$
f5	$1.19E+01 \pm 1.38E+01$	$1.11E+01 \pm 1.39E+01$†	$1.29E+01 \pm 1.41E+01$
f6	$0.00E+00 \pm 0.00E+00$†	$0.00E+00 \pm 0.00E+00$†	$0.00E+00 \pm 0.00E+00$†
f7	$3.54E-03 \pm 4.00E-03$	$3.50E-03 \pm 3.33E-03$†	$4.63E-03 \pm 4.28E-03$
f8	$1.26E+04 \pm 0.00E+00$+	$1.26E+04 \pm 0.00E+00$†	$1.26E+04 \pm 0.00E+00$†
f9	$0.00E+00 \pm 0.00E+00$†	$0.00E+00 \pm 0.00E+00$†	$0.00E+00 \pm 0.00E+00$†
f10	$4.74E-16 \pm 1.23E-15$†	$4.74E-16 \pm 1.23E-15$†	$1.07E-15 \pm 1.66E-15$
f11	$0.00E+00 \pm 0.00E+00$†	$0.00E+00 \pm 0.00E+00$†	$0.00E+00 \pm 0.00E+00$†
f12	$1.57E-32 \pm 1.91E-34$	$1.57E-32 \pm 5.57E-48$†	$1.57E-32 \pm 5.57E-48$†
f13	$1.35E-32 \pm 5.57E-48$	$1.35E-32 \pm 5.57E-48$†	$1.35E-32 \pm 5.57E-48$†
f14	$0.00E+00 \pm 0.00E+00$†	$0.00E+00 \pm 0.00E+00$†	$0.00E+00 \pm 0.00E+00$†
f15	$1.16E+00 \pm 1.75E+00$†	$1.13E+01 \pm 1.05E+01$	$1.61E+02 \pm 1.52E+02$
f16	$2.76E-01 \pm 5.10E-01$†	$2.88E+00 \pm 3.635E+00$	$4.42E+01 \pm 4.02E+01$
f17	$0.00E+00 \pm 0.00E+00$†	$0.00E+00 \pm 0.00E+00$†	$0.00E+00 \pm 0.00E+00$†
f18	$3.55E-16 \pm 1.08E-15$†	$3.55E-16 \pm 1.08E-15$†	$9.47E-16 \pm 1.60E-15$
f19	$1.57E-32 \pm 1.77E-34$†	$1.58E-32 \pm 7.07E-34$†	$1.57E-32 \pm 5.57E-48$†
f20	$1.35E-32 \pm 5.57E-48$†	$1.35E-32 \pm 5.57E-48$†	$1.35E-32 \pm 5.57E-48$†
Best Num	16	14	10

Table 4: Friedman test results.

Algorithm	Friedman value
DE/rand/1/bin + HLS_4	1.85
DE/rand/1/bin + HLS_8	1.95
DE/rand/1/bin + HLS_16	2.22

Table 5: Experimental results of jDE, SaDE, ODE, OXDE, and jDEHLS for all functions.

	jDE	SaDE	ODE	OXDE	jDEHLS
f1	$1.31e-61 \pm 2.02e-61$-	$4.10e-131 \pm 1.54e-130$+	$1.39e-75 \pm 6.59e-75$-	$3.67e-59 \pm 7.32e-59$-	$2.83E-118 \pm 1.05E-117$
f2	$1.91e-36 \pm 1.75e-36$-	$2.80e-79 \pm 5.88e-79$+	$1.21e-23 \pm 1.29e-23$-	$4.09e-33 \pm 3.66e-33$-	$3.92E-62 \pm 1.13E-61$
f3	$2.67e-07 \pm 4.76e-07$-	$5.66e-07 \pm 1.25e-06$-	$4.98e-08 \pm 6.62e-08$-	$2.91e-05 \pm 3.39e-05$-	$6.29E-09 \pm 1.80E-08$
f4	$8.05e-01 \pm 1.72e+00$-	$2.11e-07 \pm 1.15e-06$-	$2.89e-10 \pm 1.58e-09$-	$7.08e+00 \pm 4.64e+00$-	$2.15E-94 \pm 1.18E-93$
f5	$9.58e+00 \pm 9.25e-01$-	$3.14e+00 \pm 2.61e+00$+	$2.53e+01 \pm 8.26e-01$-	$1.20e+00 \pm 1.86e+00$+	$4.68E+00 \pm 1.06E+01$
f6	$0.00e+00 \pm 0.00e+00$≈	$0.00e+00 \pm 0.00e+00$≈	$0.00e+00 \pm 0.00e+00$≈	$0.00e+00 \pm 0.00e+00$≈	$0.00E+00 \pm 0.00E+00$
f7	$3.38e-03 \pm 8.43e-04$-	$2.60e-03 \pm 1.10e-03$-	$7.87e-04 \pm 2.23e-04$-	$4.08e-03 \pm 1.59e-03$-	$4.68E-04 \pm 3.12E-04$
f8	$3.82e-04 \pm 0.00e+00$≈	$3.82e-04 \pm 0.00e+00$≈	$6.57e+03 \pm 5.69e+02$-	$3.94e+00 \pm 2.16e+01$-	$3.82E-04 \pm 0.00$+00
f9	$0.00e+00 \pm 0.00e+00$≈	$0.00e+00 \pm 0.00e+00$≈	$2.91e+01 \pm 1.78e+01$-	$1.01e+01 \pm 2.79e+00$-	$0.00E+00 \pm 0.00E+00$
f10	$3.91e-15 \pm 1.08e-15$-	$1.24e-01 \pm 3.22e-01$-	$3.55e-15 \pm 0.00e+00$-	$3.10e-02 \pm 1.70e-01$-	$0.00E+00 \pm 0.00E+00$
f11	$0.00e+00 \pm 0.00e+00$≈	$2.22e-03 \pm 4.90e-03$-	$1.15e-03 \pm 3.02e-03$-	$1.15e-03 \pm 3.09e-03$-	$0.00E+00 \pm 0.00E+00$
f12	$1.57e-32 \pm 5.57e-48$≈	$3.46e-03 \pm 1.89e-02$≈	$1.58e-32 \pm 2.63e-34$-	$1.04e-02 \pm 3.16e-02$-	$1.57E-32 \pm 5.57E-48$
f13	$1.35e-32 \pm 5.57e-48$≈	$1.10e-03 \pm 3.3e-03$≈	$1.37e-32 \pm 9.00e-34$≈	$1.83e-03 \pm 8.20e-03$≈	$1.35E-32 \pm 5.57E-48$
f14	$0.00e+00 \pm 0.00e+00$≈	$1.64e-33 \pm 9.00e-33$≈	$6.57e-33 \pm 3.60e-32$≈	$1.31e-31 \pm 5.89e-31$	$0.00E+00 \pm 0.00E+00$
f15	$2.07e-07 \pm 2.95e-07$-	$3.28e-07 \pm 5.69e-07$-	$1.03e-07 \pm 2.14e-07$-	$1.45e-05 \pm 1.59e-05$-	$5.6E-08 \pm 2.93E-07$
f16	$3.87e-07 \pm 4.23e-07$-	$1.66e-01 \pm 5.26e-01$-	$1.21e-07 \pm 1.63e-07$+	$1.15e-04 \pm 1.95e-04$-	$1.99E-07 \pm 1.08E-06$
f17	$0.00e+00 \pm 0.00e+00$≈	$2.22e-03 \pm 5.35e-03$-	$3.29e-04 \pm 1.80e-03$≈	$2.55e-03 \pm 4.51e-03$-	$0.00E+00 \pm 0.00E+00$
f18	$3.67e-15 \pm 6.49e-16$-	$3.10e-02 \pm 1.70e-01$-	$3.55e-15 \pm 0.00e+00$-	$3.55e-15 \pm 0.00e+00$-	$0.00E+00 \pm 0.00E+00$
f19	$1.57e-32 \pm 5.57e-48$≈	$1.18e-02 \pm 3.21e-02$-	$1.57e-32 \pm 5.57e-48$≈	$1.58e-32 \pm 3.29e-34$≈	$1.57E-32 \pm 5.57E-48$
f20	$1.35e-32 \pm 5.57e-48$≈	$3.66e-04 \pm 2.01e-03$≈	$1.61e-32 \pm 1.44e-32$≈	$1.10e-03 \pm 3.35e-03$-	$1.35E-32 \pm 5.57E-48$
−/+/≈	10/0/0	10/3/7	13/1/6	16/1/3	

Table 6: Statistical results of jDE, SaDE, ODE, OXDE, and jDE based on the multiproblem Wilcoxon's test.

jDEHLS vs	*R*+	R-	*P* value	A = 0.05
OXDE	182.5	27.5	0.003592	4.20
SaDE	162.5	27.5	0.005954	3.45
ODE	194.5	15.5	0.000736	3.05
jDE	174.5	16.0	0.001378	2.65

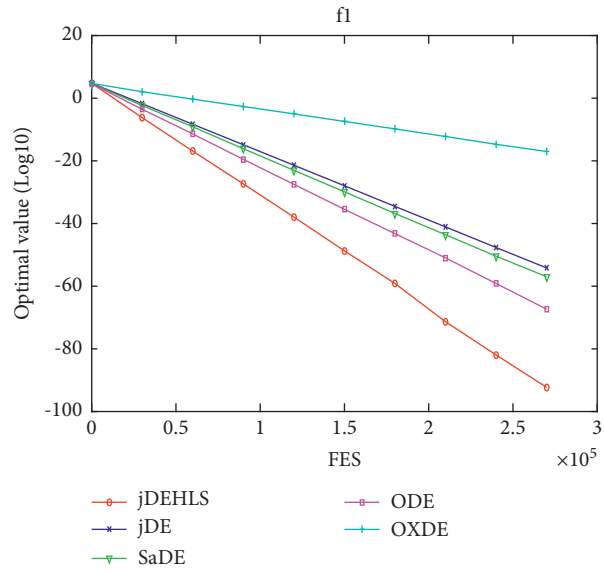

Figure 11: f1 convergence curves.

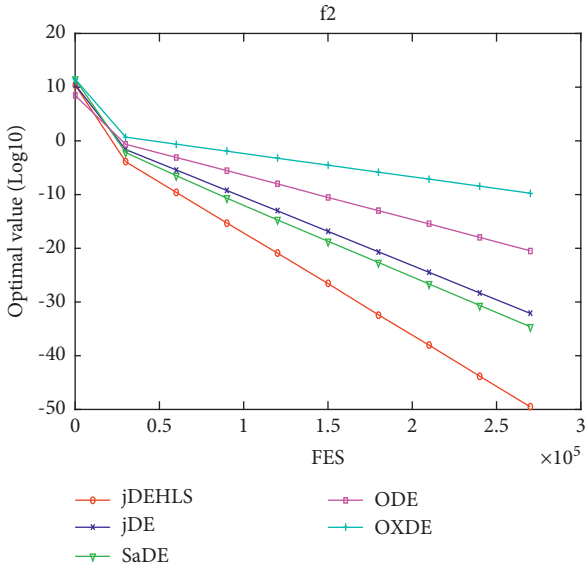

Figure 12: f2 convergence curves.

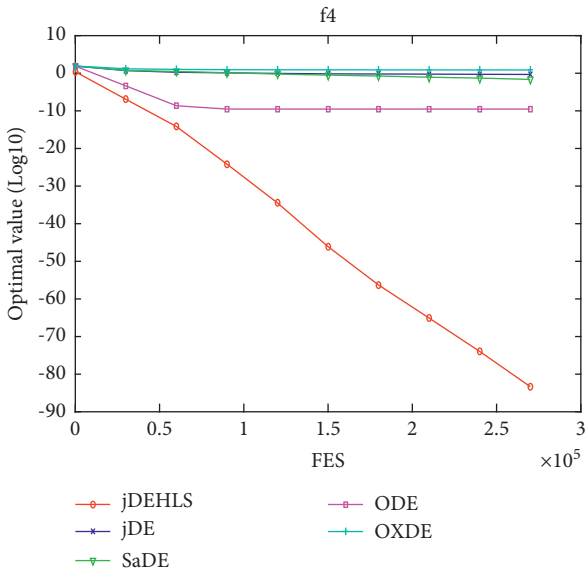

Figure 13: f4 convergence curves.

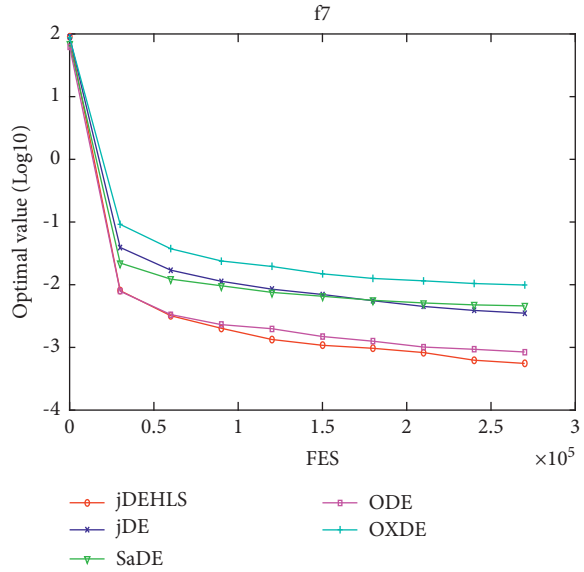

Figure 14: f7 convergence curves.

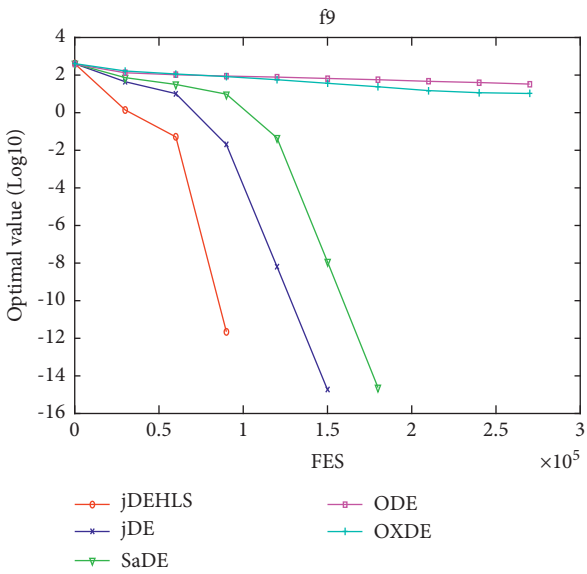

Figure 15: f9 convergence curves.

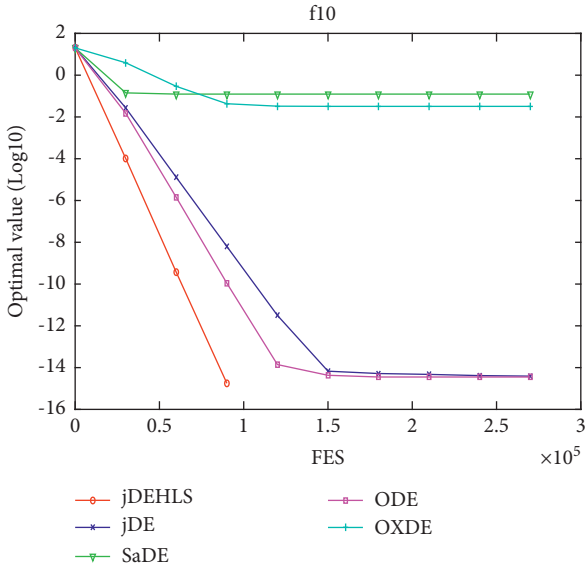

Figure 16: f10 convergence curves.

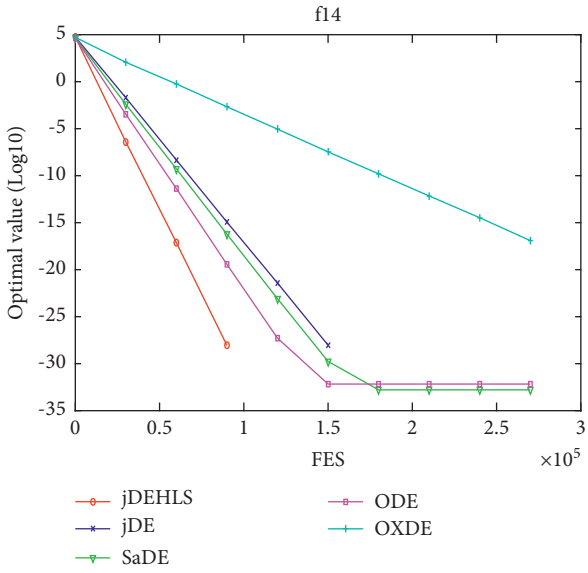

Figure 17: f14 convergence curves.

Figures 11–18 show the convergence process of the five algorithms on eight representative test functions. As illustrated in this eight graphs, jDEHLS converges faster than other algorithms.

Figure 19 shows the ranking of the numbers of optimal solutions achieved by each algorithm. Obviously, jDEHLS is also the first ranking.

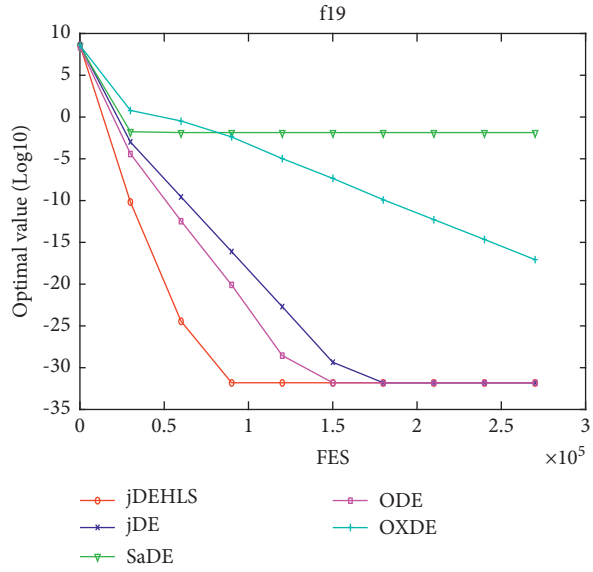

Figure 18: f19 convergence curves.

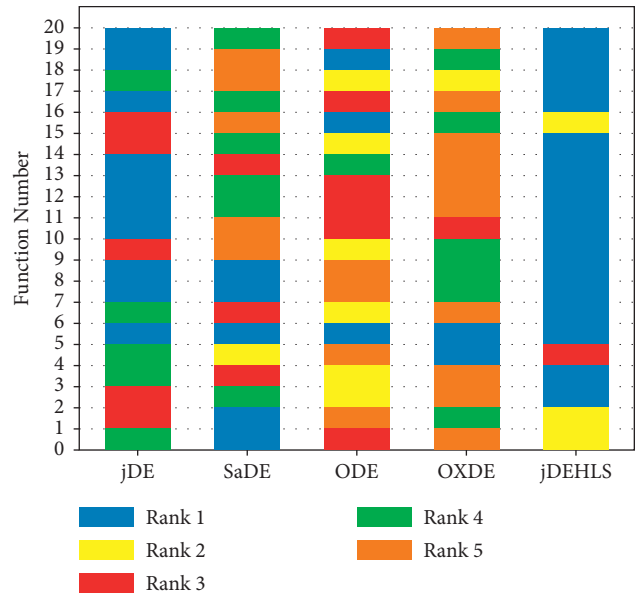

Figure 19: Ranking of five algorithms for 20 test functions.

6. Conclusion

This paper presents a new local search operator based on Hadamard matrix, which is called HLS. HLS searches the subspace defined by two randomly selected individuals. HLS can improve the probability of finding a better solution in a specified space by constructing multiple offspring. This is very beneficial to promote the balance between exploration

and exploitation. Implementation of four classical DE algorithms and one DE variant, jDE, demonstrates the effectiveness of HLS. In future work, a parameter adaption mechanism for HLS is expected to be developed. In addition, using HLS for large-scale optimization problems will be considered. Finally, the proposed HLS operator may be used to tackle some complex real-world optimization problems.

The source code of DEHLS is available at https://github.com/gitdxg110/DEHLS_v1.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (nos. 61763019, 61364025, and 62041603), the Natural Science Foundation of Jiangxi Province (nos. 20202BABL202036 and 20202BABL202019), the Science and Technology Foundation of Jiangxi Province, China (nos. GJJ180891, GJJ170953, and GJJ201808), and the "Thirteenth Five-Year Plan" of Education Science in Jiangxi Province for 2017 (no. 17YB211).

References

[1] R. Storn and K. Price, "Differential evolution—–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[2] L. Lin and M. Zhu, "Efficient tracking of moving target based on an improved fast differential evolution algorithm," *IEEE Access*, vol. 6, pp. 6820–6828, 2018.

[3] D. Zou, S. Li, X. Kong, H. Ouyang, and Z. Li, "Solving the dynamic economic dispatch by a memory-based global differential evolution and a repair technique of constraint handling," *Energy*, vol. 147, pp. 59–80, 2018.

[4] W. Yi, Y. Zhou, L. Gao, X. Li, and C. Zhang, "Engineering design optimization using an improved local search based epsilon differential evolution algorithm," *Journal of Intelligent Manufacturing*, vol. 29, no. 7, pp. 1559–1580, 2018.

[5] R. Yan, W. Li, P. Jiang, and Y. Zhou, "A modified differential evolution algorithm for resource constrained multi-project scheduling problem," *Journal of Computers*, vol. 9, no. 8, pp. 1922–1927, 2014.

[6] C. K. Desai and A. A. Shaikh, "Drill wear monitoring using artificial neural network with differential evolution learning," in *Proceedings of the IEEE International Conference on Industrial Technology*, pp. 2019–2022, Mumbai, India, December 2006.

[7] M. Versaci and A. Palumbo, "Magnetorheological fluids: qualitative comparison between a mixture model in the extended irreversible thermodynamics framework and an Herschel–Bulkley experimental elastoviscoplastic model," *International Journal of Non-linear Mechanics*, vol. 118, no. 103288, pp. 1–13, 2020.

[8] M. Versaci, A. Cutrupi, and A. Palumbo, "A magneto-thermo-static study of a magneto-rheological fluid damper: a finite element analysis," *IEEE Transactions on Magnetics*, vol. 57, no. 1, pp. 1–10, 2020.

[9] K. R. Opara and J. Arabas, "Differential Evolution: a survey of theoretical analyses," *Swarm and evolutionary computation*, vol. 44, pp. 546–558, 2019.

[10] M. Ramadas, A. Abraham, and S. Kumar, "ReDE-A revised mutation strategy for differential evolution algorithm," *International Journal of Intelligent Engineering and Systems*, vol. 9, pp. 51–58, 2016.

[11] A. W. Mohamed and A. S. Almazyad, "Differential evolution with novel mutation and adaptive crossover strategies for solving large scale global optimization problems," *Applied Computational Intelligence and Soft Computing*, vol. 2017, Article ID 7974218, 18 pages, 2017.

[12] W. Gong and Z. Cai, "Differential evolution with ranking-based mutation operators," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 2066–2081, 2013.

[13] H. Peng, Z. Guo, C. Deng, and Z. Wu, "Enhancing differential evolution with random neighbors based strategy," *Journal of Computational Science*, vol. 26, pp. 501–511, 2018.

[14] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[15] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[16] W. Zhu, Y. Tang, J.-A. Fang, and W. Zhang, "Adaptive population tuning scheme for differential evolution," *Information Sciences*, vol. 223, no. 2, pp. 164–191, 2013.

[17] Y. Wang, Z. Cai, and Q. Zhang, "Enhancing the search ability of differential evolution through orthogonal crossover," *Information Sciences*, vol. 185, no. 1, pp. 153–177, 2012.

[18] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2007.

[19] J. Sun, Q. Zhang, and E. Tsang, "A new evolutionary algorithm for global optimization," *Information Sciences*, vol. 169, no. 3, pp. 249–262, 2005.

[20] H. Peng, Z. Wu, and C. Deng, "Enhancing differential evolution with commensal learning and uniform local search," *Chinese Journal of Electronics*, vol. 26, no. 4, pp. 725–733, 2017.

[21] N. Noman and H. Iba, "Enhancing differential evolution performance with local search for high dimensional function optimization," in *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, pp. 967–974, New York, NY, USA, June 2005.

[22] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 107–125, 2008.

[23] M. Ali, M. Pant, and A. Nagar, "Two local search strategies for Differential Evolution," in *Proceedings of the IEEE 5th International Conference on Bio-Inspired Computing: Theories and Applications IEEE*, pp. 1429–1435, Changsha, China, May 2010.

[24] A. W. Mohamed, "RDEL: Restart differential evolution algorithm with local search mutation for global numerical

optimization," *Egyptian Informatics Journal*, vol. 15, no. 3, pp. 175–188, 2014.

[25] N. Noman, D. Bollegala, and H. Iba, "Differential evolution with self-adaptive local search," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, pp. 1099–1106, Dublin, Ireland, July 2011.

[26] M. L. Ortiz and N. Xiong, "Using random local search helps in avoiding local optimum in differential evolution," in *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications*, pp. 816–021, Innsbruck, Austria, 2014.

[27] H. Peng and Z. Wu, "Heterozygous differential evolution with Taguchi local search," *Soft Computing*, vol. 19, no. 1, pp. 3273–3291, 2015.

[28] A. T. Butson, "Generakized hadamard matrices," *Proceedings of the American Mathematical Society*, vol. 13, no. 6, pp. 894–898, 1960.

[29] X. Xin Yao, Y. Yong Liu, and G. Guangming Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[30] H. Wang, Z. Wu, and Y. Liu, "Space transformation search: a new evolutionary technique," in *Proceedings of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pp. 537–544, Shanghai, China, May 2009.