applied sciences

MDPI

Article

Multi-Population Differential Evolution Algorithm with Uniform Local Search

Xujie Tan [1], Seong-Yoon Shin [2,*], Kwang-Seong Shin [3,*] and Guangxing Wang [1]

[1] School of Computer and Big Data Science, JiuJiang University, Jiujiang 332005, China
[2] School of Computer Information & Communication Engineering, Kunsan National University, Gunsan 54150, Korea
[3] Department of Digital Contents Engineering, Wonkwang University, Iksan 332005, Korea
* Correspondence: s3397220@kunsan.ac.kr (S.-Y.S.); waver0920@wku.ac.kr (K.-S.S.); Tel.: +82-63-469-4860 (S.-Y.S.); +82-63-850-7270 (K.-S.S.)

Abstract: Differential evolution (DE) is a very effective stochastic optimization algorithm based on population for solving various real-world problems. The quality of solutions to these problems is mainly determined by the combination of mutation strategies and their parameters in DE. However, in the process of solving these problems, the population diversity and local search ability will gradually deteriorate. Therefore, we propose a multi-population differential evolution (MUDE) algorithm with a uniform local search to balance exploitation and exploration. With MUDE, the population is divided into multiple subpopulations with different population sizes, which perform different mutation strategies according to the evolution ratio, i.e., DE/rand/1, DE/current-to-rand/1, and DE/current-to-pbest/1. To improve the diversity of the population, the information is migrated between subpopulations by the soft-island model. Furthermore, the local search ability is improved by way of the uniform local search. As a result, the proposed MUDE maintains exploitation and exploration capabilities throughout the process. MUDE is extensively evaluated on 25 functions of the CEC 2005 benchmark. The comparison results show that the MUDE algorithm is very competitive with other DE variants and optimization algorithms in generating efficient solutions.

Keywords: differential evolution; multiple strategies; multiple population; soft island model; uniform local search

Citation: Tan, X.; Shin, S.-Y.; Shin, K.-S.; Wang, G. Multi-Population Differential Evolution Algorithm with Uniform Local Search. *Appl. Sci.* **2022**, *12*, 8087. https://doi.org/10.3390/app12168087

Academic Editors: Yun Seop Yu, Dongsik Jo, Hee-Cheol Kim, Kwang-Baek Kim and Jeong Wook Seo

Received: 12 July 2022
Accepted: 11 August 2022
Published: 12 August 2022

1. Introduction

Differential evolution (DE), proposed by Price and Storn, is an uncomplicated and powerful heuristics optimization algorithm [1]. Recently, DE has been successfully applied to various areas of optimization problems, such as text classification [2], vehicle path planning [3], energy conservation [4], and linear ordering problems [5].

DE performs three operations of mutation, crossover, and selection to search for the global solution for the population. Furthermore, DE has three parameters: population size *NP*, scaling factor *F*, and crossover control parameter *CR*. It is helpful to choose appropriate operations and parameters to improve the quality of the solution and search efficiency [6]. However, it is time-consuming to select these simply by trial and error [7], so an adaptive method is proposed to improve the performance of DE, which uses a feedback mechanism to adjust the operations and parameters of the algorithm from generation to generation [8]. However, poor performance is also possible when applying the appropriate operations and parameters for one problem to other problems. As a result, researchers have proposed the adaptive method of operations and parameters to improve DE performance [9]. When dealing with large-scale and high-dimensional problems, the improved DE will suffer from dimensional disaster [10]. To solve this problem, many studies have proposed a range of methods to improve DE performance, for example, cooperative coevolution [11], and multi-population strategies [12]. Meanwhile, researchers have revealed that the whole population

is randomly assigned to different domains of space in the process of evolution. According to the domain characteristics, appropriate operations and parameters are allocated to the corresponding domain in order to facilitate search efficiency.

Motivated by these investigations, a novel multi-population DE algorithm, the multi-population differential evolution algorithm with uniform local search (MUDE), was proposed. With MUDE, the whole population is separated into multiple subpopulations with the stochastic method. The best mutation operator is assigned to the subpopulation according to the evolution ratio of each subpopulation. Information migration between subpopulations is performed according to the soft island model to improve population diversity. After the subpopulation has evolved, the best solution for each subpopulation is performed with a uniform local search (UL) to improve solution precision. To analyze the efficiency of MUDE, 25 functions of CEC 2005 with 30 dimensions were extensively conducted. Extensive statistical results revealed that MUDE is an effective and competitive DE revision.

The main objectives of the proposed MUDE include the following:

1. A novel multi-population DE with multi-strategies is proposed to solve multi-modal problems.
2. The soft-island model is applied to exchange information between subpopulations for improving population diversity.
3. The uniform local search is used to improve the local search capability of the population.

The remainder of this paper is organized as follows. Section 2 introduces DE. Section 3 reviews the related work. Our proposed DE algorithm, MUDE, is presented in detail in Section 4. Section 5 gives the experimental results. Section 6 is the conclusions and future work.

2. Differential Evolution

In this chapter, as shown in [13], the contents of the expression of the objective function and the random propagation and the contents of the three stages of promotion of mutation, crossover, and selection are presented. In this paper, there are very important formulas in the contents shown in [13], so it is rewritten to understand the overall contents.

Research manuscripts reporting large datasets that are deposited in a publicly available database should specify where the data have been deposited and provide the relevant accession numbers. If the accession numbers have not yet been obtained at the time of submission, please state that they will be provided during review. They must be provided prior to publication.

Differential evolution performed is to solve the global optimization problem based on real coding. The objective function can be expressed as below:

$$\begin{gathered}\text{Minimize} f(x) st \\ = (x_1,\ x_2,\ \ldots,\ x_D),\ x_i \in [min, max]\end{gathered}$$

D indicates the dimension, and min and max are the range of the solution space.

The population x is randomly propagated as follows

$$x_{i,j} = \min + rnd(max - min) \tag{1}$$

where $rnd \in [0, 1]$ is a random number.

After the population is initialed by Formula (1), the population promotes generations with the three steps below.

2.1. Mutation

At the G generation, the mutant vector v_i can be produced by the following method.

1. DE/rand/1:

$$v_i = x_{r_1} + F_i(x_{r_2} - x_{r_3}) \tag{2}$$

2. DE/best/1:

$$v_i = x_{best} + F_i(x_{r_2} - x_{r_3}) \tag{3}$$

3. DE/current-to-rand/1:

$$v_i = x_i + F_i(x_{r_1} - x_i) + F_i(x_{r_2} - x_{r_3}) \tag{4}$$

4. DE/current-to-best/1

$$v_i = x_i + F_i(x_{best} - x_i) + F_i(x_{r_3} - x_{r_4}) \tag{5}$$

5. DE/current-to-pbest/1

$$v_i = x_i + F_i\left(x_{pbest} - x_i + x_{r_4} - \widetilde{x}_{r_5}\right) \tag{6}$$

where $i = 1, 2, \ldots, NP$, random different integer r_1, r_2, r_3, r_4, and $r_5 \in [1, NP]$. The control parameter $F_i \in [0, 1]$. The x_{best} is the best vector at generation G. x_{pbest} is one of the best top 100 p% of individuals.

2.2. Crossover

After mutation, DE performs crossover on x_i and v_i to generate u_i. The binomial crossover is defined as follows:

$$u_{i,j} = \begin{cases} v_{i,j}, & if(rnd \leq CR_i \ \ or \ \ j = j_{rnd}) \\ x_{i,j}, & otherwise \end{cases} \tag{7}$$

where $j = 1, 2, \cdots, D$, $rnd \in [0, 1]$, $j_{rnd} \in [1, D]$, the condition $j = j_{rnd}$ ensures that the vector u gets at least one variable from v. $CR \in [0, 1]$.

2.3. Selection

Finally, the best vector survives to the next generation by comparing the value of the function. A greedy selection scheme is described as

$$x_i = \begin{cases} u_i, & if(f(u_i) \leq f(x_i) \\ x_i, & otherwise \end{cases} \tag{8}$$

where $f(.)$ is a function.

DE repeatedly generates the three steps above until the termination condition is reached.

3. Previous Work

Evolutionary algorithms (EAs) [14] are meta-heuristic search algorithms inspired by biological evolution mechanisms, i.e., genetic algorithms (GA) [15], particle swarm optimization (PSO) [16], and evolution strategies (ES) [17], and so on. GA is a general iterative algorithm for exploring optimal solutions to problems. These meta-heuristic iterative algorithms, which also include simulated annealing (SA) [18] and tabu search (TS) [19], are very efficient and robust in solving various real-world problems [20].

DE is an efficient population-based optimization algorithm over continuous spaces. DE performance is mainly determined by mutation operators and crossover operators and three parameters [21]. Generally, the appropriate configuration of strategies and parameters may result in fast convergence and computational resources [22]. It is obvious that DE is more effective than the above meta-heuristic iterative algorithms in solving continuous optimization [23]. Therefore, many investigators have put forward various advanced DE variants to improve DE performance recently.

Comprehensive studies on the control parameters have been carried out. To avoid local optimum with maximum probability, scaling factor F should be as large as possible [24]. It is a good choice for values of F to have a value between 0.4 and 0.95 in extensive

experiments [25]. The crossover rate *CR* mainly controls the component number from the mutation individual. A large *CR* value is helpful in improving the convergence speed of the algorithm. It is said that a *CR* between 0.3 and 0.9 is a good choice [26]. This good value is achieved by tuning with trial and error. Therefore, it is very time-consuming to find a suitable parameter. To avoid this problem, various parameter adaptation methods were developed. A self-adaptation method (SDE) [27] is presented, which employs N (0.5, 0.15) to randomly generate *CR* for different individuals, whereas *F* adopts a similar method in [28]. jDE [29] is proposed by Brest et al., as a variant of DE that individually encodes parameters *F* and *CR* for self-adjustment during the DE process. These improvements are mainly tuning parameters to improve DE without considering operations.

Furthermore, numerous studies have engaged in improving mutation operations to improve DE. The standard DE adopts the DE/rand/1 to enhance global search capability. To improve the local search capability, the best individual is taken as the baseline vector in the mutation, such as DE/best/1. However, it is difficult for these mutations to achieve a balance between exploration and exploitation. Therefore, an adaptive DE with an optional external archive (JADE) [30] uses the DE/current-to-pbest/1 strategy to balance exploration and exploitation capabilities.

Currently, an ensemble of strategies and parameters is being explored in attempt to reduce trial and error. A variety of strategies and parameters are combined into a pool, and a suitable scheme is selected for next-generation evolution according to prior experience. SaDE [31] has been proposed, in which the strategies and the control parameters are adaptively adjusted based on historical experience in generating valuable solutions. Composite differential evolution (CoDE) [32] has been shown to improve DE performance by combining several effective trial vector generation strategies with some suitable control parameter settings. With cultivated differential evolution (CUDE) [33], an optimal strategy is chosen as the current population to optimize from the pool, with various mutation strategies and parameters determined according to commensal learning.

In the above mentioned improved algorithm, the diversity of the population becomes worse in the process of evolution. Therefore, the island model was introduced into the evolutionary algorithm to improve its performance. MPCCA [34] is a cooperative algorithm with many populations, in which the population is split into many subpopulations according to individuals with different directions. MPEDE [35] is proposed by Wu et al. to generate a collection of multiple strategies, which concurrently consists of three differential strategies: "current-to-pbest/1", "current-to-rand/1", and "rand/1". Tong et al. [36] proposed an improved multi-population ensemble differential evolution (IMPEDE). With IMPEDE, a novel efficient mutation strategy "DE/pbad-to-pbest" is proposed, reserving the best individuals and the worst to balance the exploitation and the exploration. Moreover, IMPEDE utilizes the weighted Lehmer mean method to promote parameter adaptation. Li et al. [37] proposed a differential evolution algorithm with multi-population cooperation and multi-strategy integration (MPMSDE), which uses strategy rankings to allocate computational capacities to different strategies. The new strategy "DE/pbad-to-pbest-to-gbest" can not only accelerate the convergence of DE but also balance the exploration and exploitation. As a result, the use of multiple populations and strategies contributes to an improved DE performance. However, the migration mechanism and local search ability of the population need to be further improved.

4. The Proposed Algorithm

In this section, a new multiple population DE algorithm (namely MUDE) is proposed, which employs appropriate strategies for each subpopulation according to evolutionary ratios and adopts SIM to transfer individuals between subpopulations. After several generations of evolution, a uniform local search is performed among subpopulations to enhance exploitation.

4.1. Soft-Island Model

The soft-island model (SIM) is primarily used to improve population diversity [38]. The core concept of SIM is the probabilistic migration between different islands.

The initial population is almost divided into various differential islands. When an individual is migrated between islands, they are selected from the current island or other islands by probabilistic P. A SIM flow chart is shown in Figure 1. The population is divided into n islands using a random method. When migrating with individual information of the kth island, the individual information is obtained from other islands according to the probability P.

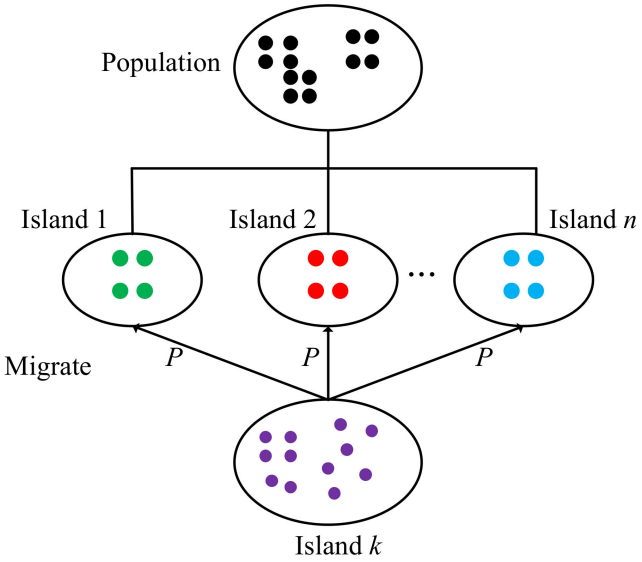

Figure 1. The soft-island model.

4.2. Uniform Local Search

Uniform design is a scientific experimental design tool [39]. The number of uniform design experiments is much less than another design tool, i.e., orthogonal design. If there are k factors including q levels ($q > 1$) in an experiment, the number of orthogonal designs in the experiment is q^2, which is too large when q is comparatively large. But the number of uniform design experiments is q under the same set of factors and levels and it provides the same functionality as orthogonal design. More details about the uniform design can be found in Ref. [40].

In the uniform design, a uniform design table is used to build an experimental plan. A uniform design table is often denoted by $U_n(n^s)$, in which the uniform design table includes n rows and s columns, and the value of each cell is obtained by the formula (n*s mod n). Table 1 presents $U_7(7^6)$ as one example.

Table 1. Uniform design table $U_7(7^6)$.

No	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	4	6	1	3	5
3	3	6	2	5	1	4
4	4	1	5	2	6	3
5	5	3	1	6	4	2
6	6	5	4	3	2	1
7	7	7	7	7	7	7

Uniform local search (ULS) uses the uniform design table to achieve the local optimization of the population. An example of ULS is shown in Figure 2.

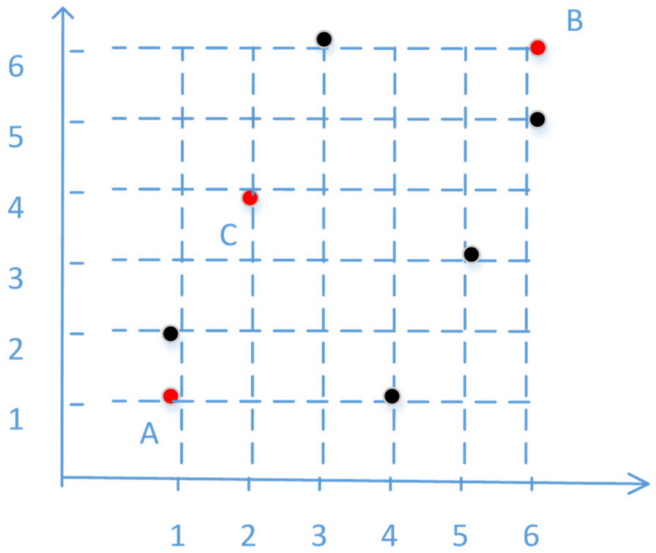

Figure 2. Example of ULS.

In this example, the uniform design table is $U_6(6^6)$. Points A(1, 1) and B(6, 6) were two randomly selected individuals from the population. The space with A and B is divided into six parts along each dimension, i.e., [1 2 3 4 5 6; 1 2 3 4 5 6].

The six points are combined by the first column and second column from the uniform design table, i.e., (1 2), (2 4), (3 6), (4 1), (5 3), and (6 5). Finally, the best point is selected by the fitness value, i.e., C.

4.3. MUDE (Multi-Population Differential Evolution)

In this subsection, we put forward a novel multi-island based DE algorithm, MUDE, which implements a differential mutation strategy to the island by evolutionary ratio. The information between islands is exchanged according to SIM, and the local exploration is carried out by ULS. The MUDE procedure is shown in Figure 3.

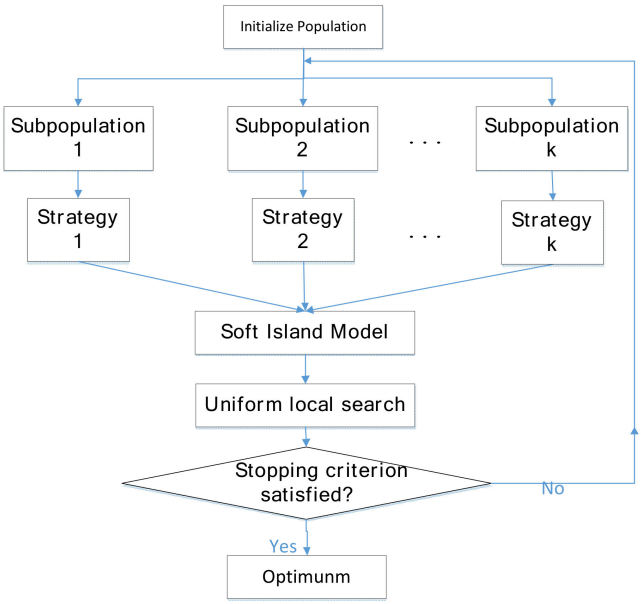

Figure 3. MUDE procedure.

In Figure 3, the soft island model is used to migrate individuals between subpopulations, and the optimal individual between subpopulations is to perform a local search by uniform local search. Meanwhile, the MUDE pseudocode is provided in Algorithm 1.

There are four input parameters in MUDE, i.e., population size (NP), individual dimension (D), migration probability (P), and benchmark function (f). The global optimal is the output value of the algorithm. On lines 1 and 2, the population p are randomly initialized and fed into the benchmark function to obtain fitness values. On line 3, we choose three strategies to build a strategy pool, Equations (2), (4), and (5).

On lines 4 and 5, the stopping condition of MUDE is controlled by the function evaluation times FES, and the MUDE will stop evolving when FES is larger than MaxFes. The population is randomly split into k subpopulations on line 6. On line 7, parameters F_i and CR_i are calculated from the Cauchy distribution and normal distribution respectively, i.e., $N(uF_i, 0.1)$ and randn$(uCR_i, 0.1)$. uFi and uCR_i are updated as follows.

$$uF_i = (1 - c) \cdot uF_i + c \cdot mean_L(S_{F,i}) \tag{9}$$

$$uCR_i = (1 - c) \cdot uCR_i + c \cdot mean_A(S_{CR,i}) \tag{10}$$

where, $mean_L$ and $mean_A$ are the Lehmer mean.

On lines 8 and 9, first, the fitness value of each island between generations is computed. Then the subpopulation chooses the appropriate strategy according to the ratio. On line 10, the individual of the subpopulation is migrated according to SIM. Finally, the global best fit_{best} is returned.

In MUDE, the multi-strategy mechanism is beneficial to enhance the exploitation ability of the population, the SIM improves population diversity, and ULS improves the exploration of the population.

Algorithm 1: MUDE algorithm

Input: NP, D, P, f
Output: The population's best solution: fit_{best}
1. Generate the population p by the Equation (1);
2. Calculate the individual function values fit;
3. Strategy pool $Sp = \{Sp_1, Sp_2, \ldots, Sp_n\}$, set $uCR_i = 0.5$, $uF_i = 0.5$ $f_i = 0$, $Fes_i = 0$ for each strategy;
4. $FES = NP$;
5. while $FES \leq MaxFes$
6. The population p is the randomly divided k subpopulation, $p = \{p_1, p_2, \ldots, p_k\}$;
7. Calculate F_i and Cr_i for each subpopulation;
8. Pick a Sp_i for each subpopulation according to evolutionary ratio;
9. Perform strategy Sp_i for p_i;
10. Migrate individual between subpopulation by probability P of SIM;
11. Perform the local exploration from each p by ULS;
12. end while
13. Return fit_{best}.

5. Experimental Results

5.1. Benchmark Functions and Experimental Setting

To evaluate MUDE more efficiently, we selected a series of 25 testing functions from the CEC 2005 benchmark to implement the experiment. Among the 25 functions, F_1~F_5 are unimodal functions, and F_6~F_{25} are multimodal functions. In multimodal functions, F_6~F_{12} are basic functions, F_{13}~F_{14} are expanded functions, and F_{15}~F_{25} are hybrid composition functions. More descriptions of F_1~F_{25} can be found in [41].

In all experiments, the MUDE parameters are allocated as follows:

1. Dimension: $D = 30$;
2. Population size: $NP = 100$;
3. Scaling factor: $F_i = N(uF_i, 0.1)$;

4. Crossover: $CR_i = N(uCR_i, 0.1)$;
5. Three different mutation strategies: Equations (1), (5), and (6);
6. Termination criterion of function evaluations

(MaxFES): MaxFES = D $\times$ 10 $\times$ 10^{+4}.

Furthermore, each function is run independently 25 times in each algorithm. All the experiments were executed on a computer with a 3.2 GHz quad-core processor and Windows 10 with 8 GB RAM. For the results, the optimal values and the evolutionary process of the function at each stage were recorded separately. Moreover, Wilcoxon's statistical tests and Friedman's statistical tests were conducted to further verify the results [42].

5.2. Experimental Results and Comparisons with Other DE Variants

In this subsection, the advantage of MUDE is obtained by extensively comparing MUDE with four other up-to-date DE versions, CoDE, jDE, JADE, and SaDE, whose control parameters F and CR are consistent with the original literature. The maximum number of fitness evaluations (MaxFES) is fixed at 3 $\times$ 10^5 in all algorithms. Table 1 shows the experimental comparison results of 25 runs on 25 functions. The Wilcoxon test results are shown at the bottom of Table 1, where the symbol "$-/+/\approx$" represents whether the average value of the algorithm is worse, better, or similar to that of MUDE.

Table 2 shows the mean and standard deviation of the optimal fitness for 25 independent runs, and the best mean values are shown in bold. It is seen that MUDE is better than CoDE, JADE, jDE, and SaDE on 12, 11, 14, and 14 functions, respectively, but worse than these challengers on 8, 6, 3, and 5 functions, respectively. For the unimodal functions F_1~F_5, the MUDE performance is superior to the other four algorithms for this type of benchmark function. As for the multimodal function F_6~F_{25}, MUDE outperforms CoDE, JADE, jDE, and SaDE on 8, 9, 10, and 10 functions, respectively. CoDE beats MUDE on F_6, F_8, F_{11}, F_{14}, and F_{22}. JADE exceeds MUDE on F_5, F_{13}, and F_{14}. jDE and SaDE excel MUDE in only one function. Accordingly, MUDE's performance is the best one in 20 multimodal functions. It is clear that information migration of MUDE can improve population diversity.

Along with the prior analysis, Wilcoxon's test and Friedman's signed-rank test were performed on all function experimental results. The average ranking of five algorithms is provided in Table 3. Friedman's statistic result shows that the smaller the value, the better the performance. Therefore, MUDE is the most competitive one in the five DE variants algorithms.

The statistical results indicate that MUDE is the most challenging algorithm for 25 benchmark functions. There are two aspects reasons why MUDE can perform well. First, the ULS of MUDE leads to efficient exploitation and high convergence precision under such circumstances. Second, the information migrating mechanism among different islands improves population diversity.

To illustrate the convergence of algorithm, fitness variation graphs of the 25 functions were made for the convergence course, which includes the mean fitness values over 25 runs with dimension D = 30. Figure 4 shows the convergence graphs of the 25 functions. In Figure 4, MUDE can continuously converge to better solutions for the unimodal function, but for the multimodal function, the algorithm may get stuck in a local solution. For function F_{18}–F_{24}, convergence curves are almost the same. In general, MUDE improves the convergence of the DE algorithm.

Table 2. Comparison with other DE algorithms for dimension = 30.

	CoDE		JADE		jDE		SaDE		MUDE	
F	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
F_1	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$
F_2	6.76×10^{-16}	1.40×10^{-15}	$\mathbf{8.20 \times 10^{-29}}$	5.60×10^{-29}	1.07×10^{-6}	2.09×10^{-6}	2.00×10^{-5}	3.31×10^{-5}	2.44×10^{-26}	6.20×10^{-26}
F_3	$1.08 \times 10^{+5}$	$5.93 \times 10^{+4}$	$7.47 \times 10^{+3}$	$6.40 \times 10^{+3}$	$1.78 \times 10^{+5}$	$1.00 \times 10^{+5}$	$4.37 \times 10^{+5}$	$2.66 \times 10^{+5}$	$\mathbf{2.98 \times 10^{+2}}$	$1.19 \times 10^{+3}$
F_4	6.56×10^{-3}	1.17×10^{-2}	1.23×10^{-13}	5.69×10^{-13}	3.52×10^{-2}	9.40×10^{-2}	$4.63 \times 10^{+0}$	$5.65 \times 10^{+0}$	$\mathbf{5.93 \times 10^{-17}}$	2.56×10^{-16}
F_5	$4.00 \times 10^{+2}$	$3.13 \times 10^{+2}$	$\mathbf{2.60 \times 10^{-7}}$	1.16×10^{-6}	$3.99 \times 10^{+2}$	$2.96 \times 10^{+2}$	$2.24 \times 10^{+3}$	$6.13 \times 10^{+2}$	1.67×10^{-4}	6.30×10^{-4}
F_6	$\mathbf{8.05 \times 10^{-9}}$	3.98×10^{-8}	$1.09 \times 10^{+1}$	$2.76 \times 10^{+1}$	$2.68 \times 10^{+1}$	$2.74 \times 10^{+1}$	$5.37 \times 10^{+1}$	$2.85 \times 10^{+1}$	$3.70 \times 10^{+0}$	$1.46 \times 10^{+1}$
F_7	$4.70 \times 10^{+3}$	2.83×10^{-12}	$4.70 \times 10^{+3}$	2.73×10^{-12}	$4.70 \times 10^{+3}$	2.95×10^{-12}	$4.70 \times 10^{+3}$	9.47×10^{-13}	$\mathbf{5.91 \times 10^{-3}}$	5.55×10^{-3}
F_8	$2.02 \times 10^{+1}$	1.18×10^{-1}	$2.09 \times 10^{+1}$	2.02×10^{-1}	$2.09 \times 10^{+1}$	5.03×10^{-2}	$2.09 \times 10^{+1}$	5.09×10^{-2}	$\mathbf{2.09 \times 10^{+1}}$	4.56×10^{-2}
F_9	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	3.55×10^{-14}	6.55×10^{-14}
F_{10}	$4.14 \times 10^{+1}$	$1.23 \times 10^{+1}$	$2.47 \times 10^{+1}$	$5.17 \times 10^{+0}$	$5.57 \times 10^{+1}$	$8.76 \times 10^{+0}$	$3.94 \times 10^{+1}$	$9.75 \times 10^{+0}$	$\mathbf{2.40 \times 10^{+1}}$	$1.03 \times 10^{+1}$
F_{11}	$\mathbf{1.35 \times 10^{+1}}$	$3.62 \times 10^{+0}$	$2.58 \times 10^{+1}$	$1.66 \times 10^{+0}$	$2.77 \times 10^{+1}$	$1.66 \times 10^{+0}$	$2.18 \times 10^{+1}$	$8.20 \times 10^{+0}$	$1.74 \times 10^{+1}$	$7.14 \times 10^{+0}$
F_{12}	$2.82 \times 10^{+3}$	$2.67 \times 10^{+3}$	$6.65 \times 10^{+3}$	$4.79 \times 10^{+3}$	$9.14 \times 10^{+3}$	$8.39 \times 10^{+3}$	$3.78 \times 10^{+3}$	$3.16 \times 10^{+3}$	$1.61 \times 10^{+3}$	$1.97 \times 10^{+3}$
F_{13}	$1.51 \times 10^{+0}$	2.52×10^{-1}	$\mathbf{1.44 \times 10^{+0}}$	1.23×10^{-1}	$1.67 \times 10^{+0}$	1.35×10^{-1}	$4.49 \times 10^{+0}$	3.48×10^{-1}	$\mathbf{2.04 \times 10^{+0}}$	2.38×10^{-1}
F_{14}	$1.22 \times 10^{+1}$	4.50×10^{-1}	$\mathbf{1.22 \times 10^{+1}}$	3.08×10^{-1}	$1.30 \times 10^{+1}$	1.64×10^{-1}	$1.29 \times 10^{+1}$	1.67×10^{-1}	$1.23 \times 10^{+1}$	2.85×10^{-1}
F_{15}	$4.24 \times 10^{+2}$	$6.63 \times 10^{+1}$	$3.84 \times 10^{+2}$	$9.43 \times 10^{+1}$	$\mathbf{3.68 \times 10^{+2}}$	$7.48 \times 10^{+1}$	$3.92 \times 10^{+2}$	$5.72 \times 10^{+1}$	$4.00 \times 10^{+2}$	$6.45 \times 10^{+1}$
F_{16}	$8.83 \times 10^{+1}$	$7.10 \times 10^{+1}$	$1.06 \times 10^{+2}$	$1.32 \times 10^{+2}$	$7.99 \times 10^{+1}$	$2.01 \times 10^{+1}$	$6.53 \times 10^{+1}$	$2.46 \times 10^{+1}$	$\mathbf{6.37 \times 10^{+1}}$	$3.66 \times 10^{+1}$
F_{17}	$8.28 \times 10^{+1}$	$7.18 \times 10^{+1}$	$1.43 \times 10^{+2}$	$1.42 \times 10^{+2}$	$1.30 \times 10^{+2}$	$1.91 \times 10^{+1}$	$\mathbf{6.22 \times 10^{+1}}$	$3.52 \times 10^{+1}$	$6.22 \times 10^{+1}$	$7.99 \times 10^{+1}$
F_{18}	$9.00 \times 10^{+2}$	$2.09 \times 10^{+1}$	$9.04 \times 10^{+2}$	9.97×10^{-1}	$9.04 \times 10^{+2}$	7.96×10^{-1}	$\mathbf{8.46 \times 10^{+2}}$	$5.72 \times 10^{+1}$	$9.04 \times 10^{+2}$	2.34×10^{-1}
F_{19}	$9.04 \times 10^{+2}$	8.48×10^{-1}	$9.04 \times 10^{+2}$	$1.13 \times 10^{+0}$	$9.04 \times 10^{+2}$	8.23×10^{-1}	$\mathbf{8.51 \times 10^{+2}}$	$5.91 \times 10^{+1}$	$9.04 \times 10^{+2}$	2.55×10^{-1}
F_{20}	$9.05 \times 10^{+2}$	$1.18 \times 10^{+0}$	$9.04 \times 10^{+2}$	9.83×10^{-1}	$9.04 \times 10^{+2}$	9.64×10^{-1}	$\mathbf{8.51 \times 10^{+2}}$	$5.82 \times 10^{+1}$	$9.04 \times 10^{+2}$	2.22×10^{-1}
F_{21}	$5.00 \times 10^{+2}$	9.91×10^{-14}	$5.00 \times 10^{+2}$	6.46×10^{-14}	$5.00 \times 10^{+2}$	8.61×10^{-14}	$\mathbf{5.00 \times 10^{+2}}$	5.80×10^{-14}	$5.00 \times 10^{+2}$	6.14×10^{-14}
F_{22}	$\mathbf{8.58 \times 10^{+2}}$	$2.68 \times 10^{+1}$	$8.69 \times 10^{+2}$	$2.08 \times 10^{+1}$	$8.74 \times 10^{+2}$	$1.83 \times 10^{+1}$	$9.21 \times 10^{+2}$	$1.51 \times 10^{+1}$	$8.64 \times 10^{+2}$	$1.77 \times 10^{+1}$
F_{23}	$5.34 \times 10^{+2}$	4.00×10^{-4}	$\mathbf{5.34 \times 10^{+2}}$	2.48×10^{-13}	$5.34 \times 10^{+2}$	1.39×10^{-4}	$5.34 \times 10^{+2}$	2.92×10^{-4}	$5.34 \times 10^{+2}$	2.58×10^{-13}
F_{24}	$2.00 \times 10^{+2}$	2.90×10^{-14}	$2.00 \times 10^{+2}$	2.90×10^{-14}	$2.00 \times 10^{+2}$	2.90×10^{-14}	$2.00 \times 10^{+2}$	2.90×10^{-14}	$2.00 \times 10^{+2}$	2.90×10^{-14}
F_{25}	$1.64 \times 10^{+3}$	$4.95 \times 10^{+0}$	$1.63 \times 10^{+3}$	$3.52 \times 10^{+0}$	$1.63 \times 10^{+3}$	$4.41 \times 10^{+0}$	$1.65 \times 10^{+3}$	$3.42 \times 10^{+0}$	$\mathbf{2.09 \times 10^{+2}}$	5.27×10^{-1}
$-/+/\approx$	8/12/5		6/11/8		3/14/8		5/14/6			

Table 3. Average ranking based on Friedman's test.

Algorithm	CoDE	JADE	jDE	SaDE	MUDE
Ranking	2.84	2.84	3.54	3.36	2.24

Furthermore, the total running time of the 25 functions is compared in five different DE algorithms, as shown in Figure 5.

In Figure 5, the abscissa and ordinate represent the running time (h) and different DE algorithms, respectively. The results show that JADE has the shortest running time and jDE has the longest running time. The proposed MUDE leads to a longer running time because of population migration and local search.

5.3. Result with Different Probabilistics among Islands

In MUDE, an individual of one island is migrated to another island by random probabilistic P. To verify the effect of P on DE performance, we select five different values of 0, 0.2, 0.4, 0.6, 0.8, 0.9, and 1 on $D = 30$ for experiments of 25 independent runs. Table 4 shows the average ranking of different probabilistic based on Friedman's test. In Table 4, it is obvious that MUDE produces better performance when $P = 0.8$ and 0.9. The individual comes from another island in the process of migration when $P = 0$. In addition, migration never occurs when $P = 1$. Therefore, MUDE shows poor effectiveness in those two cases, but consequently, MUDE shows good diversity through migration of the soft-island model.

Table 4. Average ranking different of probabilistic P based on Friedman's test.

P	0	0.2	0.4	0.6	0.8	0.9	1.0
Ranking	4.22	3.98	4.28	3.86	3.72	3.82	4.12

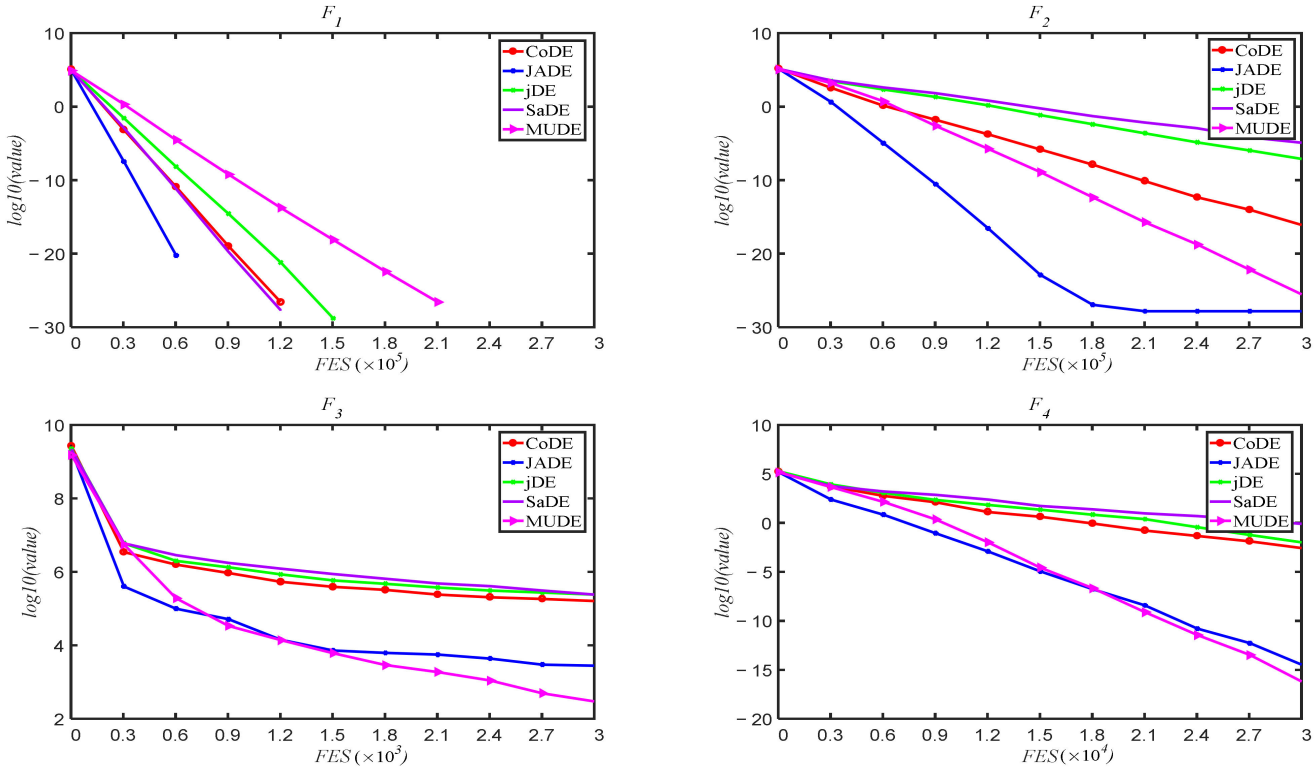

Figure 4. *Cont.*

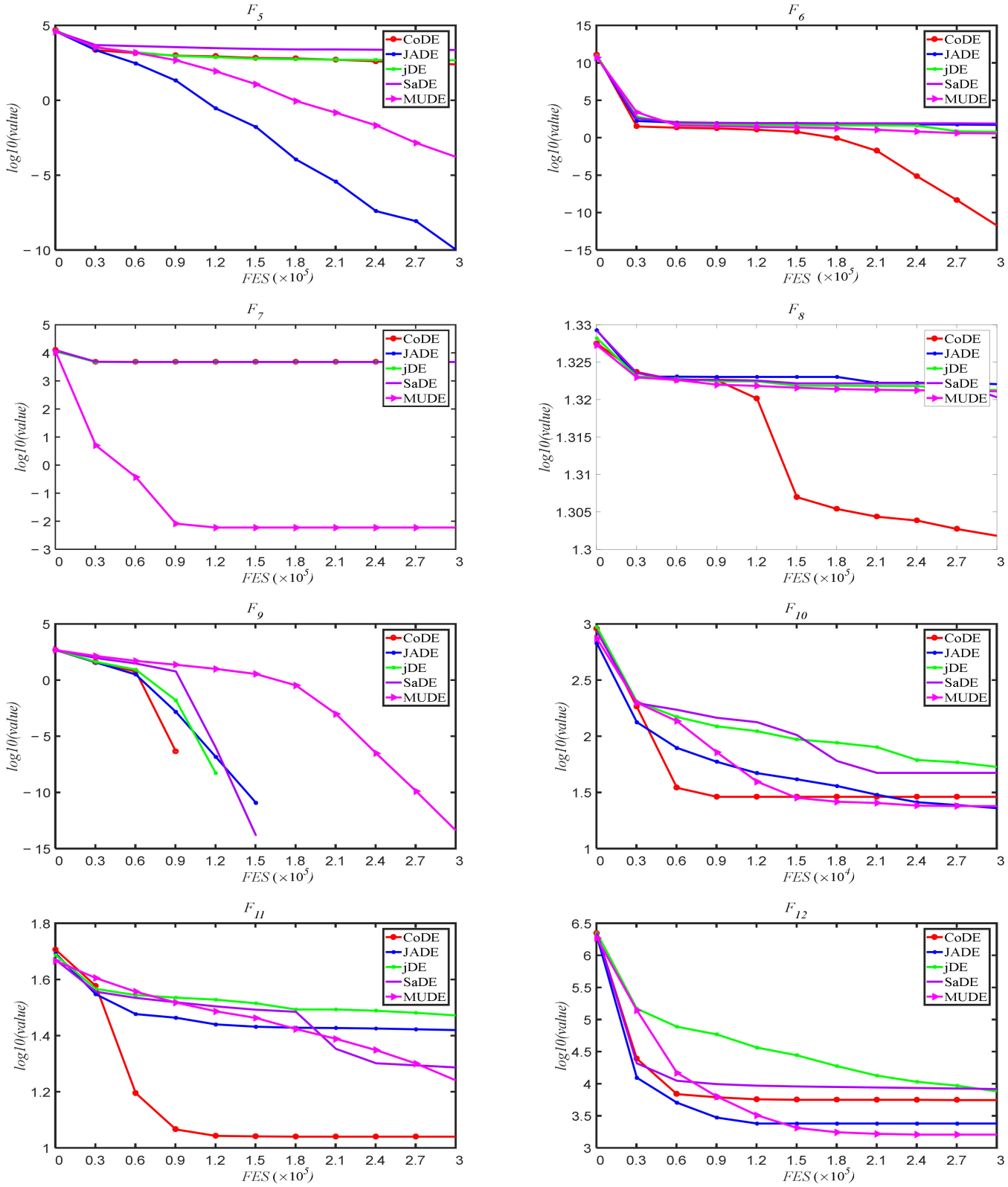

Figure 4. *Cont.*

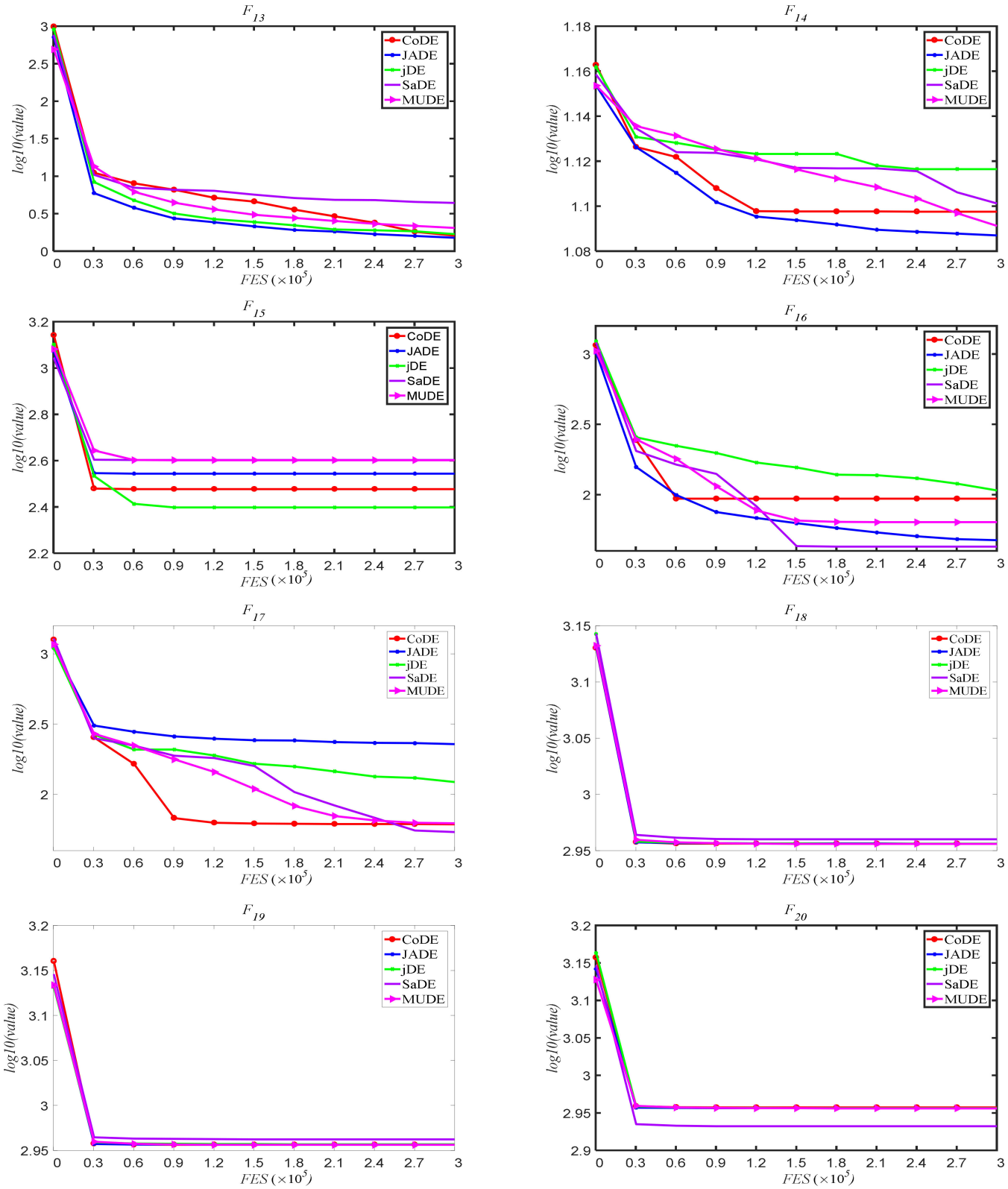

Figure 4. *Cont.*

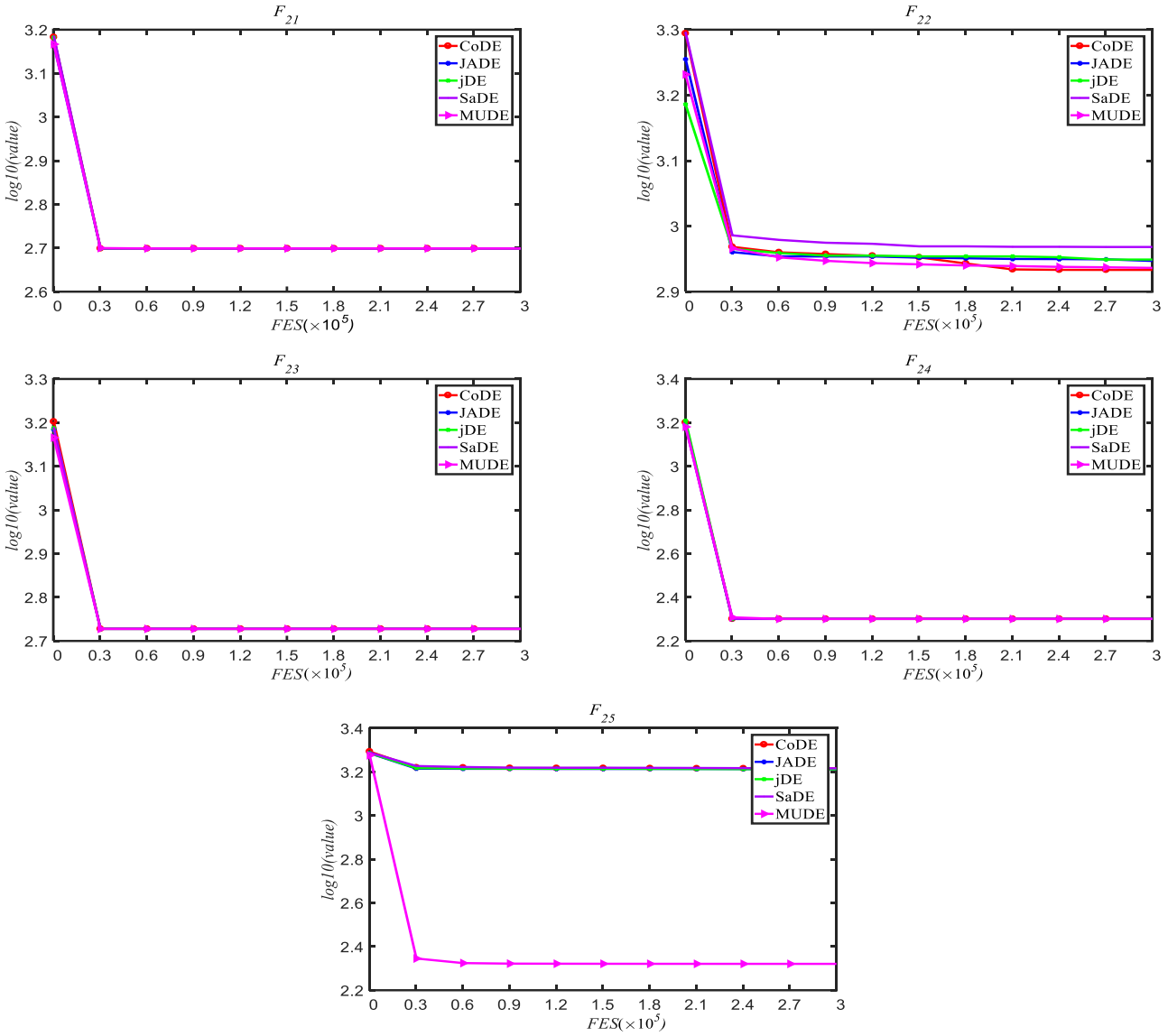

Figure 4. The variation process of the average optimum solutions for F_1~F_{25} with dimension D = 30 over 25 runs.

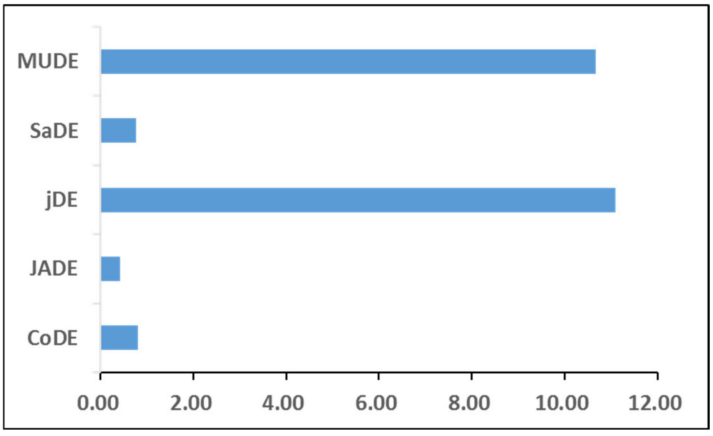

Figure 5. The total running time of 25 functions in different DE algorithms.

To further indicate the choice of P, Figure 6 shows the average value of 25 functions for different P values. In Figure 6, we see that MUDE performance gradually improves as P goes from 0 to 0.6. MUDE performs best when $P = 0.9$. Therefore, in this paper, we choose $P = 0.9$ for experiments.

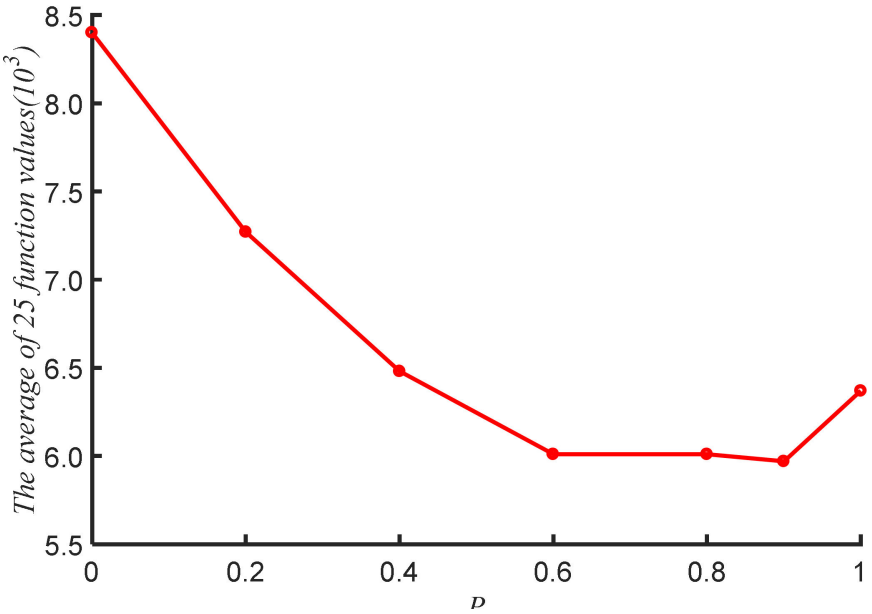

Figure 6. The average values of 25 function in different P.

5.4. Experimental on D = 10, 30, 50

To analyze the extensibility of MUDE, populations with three dimensions are selected for experiments. During the experiment, the number of function evaluations remains consistent. Table 5 shows the mean, standard deviation, and run time (second) of the optimal fitness on three dimensions for 25 independent runs. In Table 5, the best mean values are shown in bold. It is clearly seen that the unimodal function of $D = 10$ is superior to the other dimensions. In multimodal functions, except for F_{16}, F_{17}, F_{23}, and F_{25}, which have the best performers on $D = 30$, the other functions perform best on $D = 10$. As a result, in MUDE, the optimal value of a function deteriorates with increasing dimensions, while the running time increases exponentially. In addition, the results show that MUDE can converge to the optimal solution with sufficient iterations.

5.5. Comparison with other EAs

To further validate MUDE, other optimization algorithms are used to compare it, such as PSO, Tabu, GA, and so on. To evaluate each algorithm fairly, the termination condition of each algorithm is that the number of fitness evaluations is set to 3×10^5. The comparison results of $D = 30$ with other algorithms are shown in Table 6. In Table 6, the best mean values are shown in bold. The results show that the performance of PSO is relatively weak except for the 15th function, and the Tabu and GA algorithms are relatively weak. However, during the whole algorithm running process, the total running time of MUDE is relatively long. In general, MUDE is a challenging and efficient optimization algorithm for solving real-world problems.

Table 5. Comparison with the reference algorithm for dimensions D = 10, 30, 50.

	D = 10			D = 30			D = 50		
F	**Mean**	**Std**	**Time**	**Mean**	**Std**	**Time**	**Mean**	**Std**	**Time**
F_1	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	$3.54 \times 10^{+1}$	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	$4.84 \times 10^{+1}$	$0.00 \times 10^{+0}$	$0.00 \times 10^{+0}$	$5.18 \times 10^{+1}$
F_2	$\mathbf{0.00 \times 10^{+0}}$	$0.00 \times 10^{+0}$	$3.66 \times 10^{+1}$	2.44×10^{-26}	6.20×10^{-26}	$5.93 \times 10^{+1}$	3.43×10^{-6}	1.17×10^{-5}	$6.91 \times 10^{+1}$
F_3	$\mathbf{5.82 \times 10^{-26}}$	1.68×10^{-26}	$5.10 \times 10^{+1}$	$2.98 \times 10^{+2}$	$1.19 \times 10^{+3}$	$5.88 \times 10^{+1}$	$1.88 \times 10^{+5}$	$8.69 \times 10^{+4}$	$6.17 \times 10^{+1}$
F_4	$\mathbf{0.00 \times 10^{+0}}$	$0.00 \times 10^{+0}$	$4.43 \times 10^{+1}$	5.93×10^{-17}	2.56×10^{-16}	$6.15 \times 10^{+1}$	$4.79 \times 10^{+0}$	$4.08 \times 10^{+0}$	$8.45 \times 10^{+1}$
F_5	$\mathbf{0.00 \times 10^{+0}}$	$0.00 \times 10^{+0}$	$6.57 \times 10^{+1}$	1.67×10^{-4}	6.30×10^{-4}	$6.31 \times 10^{+1}$	$1.61 \times 10^{+3}$	$5.74 \times 10^{+2}$	$7.15 \times 10^{+1}$
F_6	$\mathbf{6.98 \times 10^{-21}}$	2.34×10^{-20}	$3.77 \times 10^{+1}$	$3.70 \times 10^{+0}$	$1.46 \times 10^{+1}$	$4.90 \times 10^{+1}$	$3.11 \times 10^{+1}$	$1.93 \times 10^{+1}$	$5.22 \times 10^{+1}$
F_7	$\mathbf{6.91 \times 10^{-4}}$	2.41×10^{-3}	$3.75 \times 10^{+1}$	5.91×10^{-3}	5.55×10^{-3}	$5.97 \times 10^{+1}$	4.04×10^{-3}	8.45×10^{-3}	$9.15 \times 10^{+1}$
F_8	$\mathbf{2.03 \times 10^{+1}}$	5.80×10^{-2}	$4.21 \times 10^{+1}$	$2.09 \times 10^{+1}$	4.56×10^{-2}	$7.04 \times 10^{+1}$	$2.12 \times 10^{+1}$	3.09×10^{-2}	$8.92 \times 10^{+1}$
F_9	$\mathbf{0.00 \times 10^{+0}}$	$0.00 \times 10^{+0}$	$3.88 \times 10^{+1}$	3.55×10^{-14}	6.55×10^{-14}	$5.35 \times 10^{+1}$	1.17×10^{-2}	1.94×10^{-2}	$5.71 \times 10^{+1}$
F_{10}	$\mathbf{2.01 \times 10^{+0}}$	$1.00 \times 10^{+0}$	$3.97 \times 10^{+1}$	$2.40 \times 10^{+1}$	$1.03 \times 10^{+1}$	$6.53 \times 10^{+1}$	$4.90 \times 10^{+1}$	$1.44 \times 10^{+1}$	$6.73 \times 10^{+1}$
F_{11}	$\mathbf{1.65 \times 10^{+0}}$	$1.16 \times 10^{+0}$	$2.43 \times 10^{+2}$	$1.74 \times 10^{+1}$	$7.14 \times 10^{+0}$	$5.00 \times 10^{+2}$	$3.97 \times 10^{+1}$	$1.13 \times 10^{+1}$	$9.88 \times 10^{+2}$
F_{12}	$\mathbf{1.55 \times 10^{+0}}$	$4.54 \times 10^{+0}$	$9.85 \times 10^{+1}$	$1.61 \times 10^{+3}$	$1.97 \times 10^{+3}$	$1.65 \times 10^{+2}$	$1.17 \times 10^{+4}$	$8.93 \times 10^{+3}$	$3.52 \times 10^{+2}$
F_{13}	$\mathbf{3.09 \times 10^{-1}}$	5.65×10^{-2}	$4.46 \times 10^{+1}$	$2.04 \times 10^{+0}$	2.38×10^{-1}	$6.26 \times 10^{+1}$	$4.60 \times 10^{+0}$	5.31×10^{-1}	$7.16 \times 10^{+1}$
F_{14}	$\mathbf{2.10 \times 10^{+0}}$	3.01×10^{-1}	$4.83 \times 10^{+1}$	$1.23 \times 10^{+1}$	2.85×10^{-1}	$8.73 \times 10^{+1}$	$2.21 \times 10^{+1}$	4.02×10^{-1}	$1.00 \times 10^{+2}$
F_{15}	$\mathbf{9.40 \times 10^{+0}}$	$1.93 \times 10^{+1}$	$8.87 \times 10^{+2}$	$4.00 \times 10^{+2}$	$6.45 \times 10^{+1}$	$1.34 \times 10^{+3}$	$3.48 \times 10^{+2}$	$9.63 \times 10^{+1}$	$2.42 \times 10^{+3}$
F_{16}	$8.78 \times 10^{+1}$	$7.76 \times 10^{+0}$	$8.11 \times 10^{+2}$	$\mathbf{6.37 \times 10^{+1}}$	$3.66 \times 10^{+1}$	$1.31 \times 10^{+3}$	$5.50 \times 10^{+1}$	$3.01 \times 10^{+1}$	$2.39 \times 10^{+3}$
F_{17}	$9.06 \times 10^{+1}$	$1.98 \times 10^{+1}$	$8.17 \times 10^{+2}$	$\mathbf{6.22 \times 10^{+1}}$	$7.99 \times 10^{+1}$	$1.32 \times 10^{+3}$	$6.09 \times 10^{+1}$	$7.65 \times 10^{+1}$	$2.39 \times 10^{+3}$
F_{18}	$\mathbf{6.00 \times 10^{+2}}$	$2.50 \times 10^{+2}$	$8.20 \times 10^{+2}$	$9.04 \times 10^{+2}$	2.34×10^{-1}	$1.41 \times 10^{+3}$	$9.18 \times 10^{+2}$	$3.78 \times 10^{+0}$	$2.66 \times 10^{+3}$
F_{19}	$\mathbf{6.40 \times 10^{+2}}$	$2.38 \times 10^{+2}$	$8.22 \times 10^{+2}$	$9.04 \times 10^{+2}$	2.55×10^{-1}	$1.41 \times 10^{+3}$	$9.17 \times 10^{+2}$	$6.32 \times 10^{+0}$	$2.53 \times 10^{+3}$
F_{20}	$\mathbf{6.00 \times 10^{+2}}$	$2.50 \times 10^{+2}$	$8.26 \times 10^{+2}$	$9.04 \times 10^{+2}$	2.22×10^{-1}	$1.42 \times 10^{+3}$	$9.16 \times 10^{+2}$	$8.88 \times 10^{+0}$	$2.52 \times 10^{+3}$

Table 5. *Cont.*

	$D = 10$			$D = 30$			$D = 50$		
F	**Mean**	**Std**	**Time**	**Mean**	**Std**	**Time**	**Mean**	**Std**	**Time**
F_{21}	$\mathbf{4.68 \times 10^{+2}}$	$7.48 \times 10^{+1}$	$8.26 \times 10^{+2}$	$5.00 \times 10^{+2}$	6.14×10^{-14}	$1.36 \times 10^{+3}$	$5.93 \times 10^{+2}$	$1.93 \times 10^{+2}$	$2.53 \times 10^{+3}$
F_{22}	$\mathbf{7.44 \times 10^{+2}}$	$5.13 \times 10^{+0}$	$9.61 \times 10^{+2}$	$8.64 \times 10^{+2}$	$1.77 \times 10^{+1}$	$1.81 \times 10^{+3}$	$9.12 \times 10^{+2}$	$1.86 \times 10^{+1}$	$3.26 \times 10^{+3}$
F_{23}	$6.15 \times 10^{+2}$	$1.02 \times 10^{+2}$	$8.25 \times 10^{+2}$	$\mathbf{5.34 \times 10^{+2}}$	2.58×10^{-13}	$1.41 \times 10^{+3}$	$7.29 \times 10^{+2}$	$2.37 \times 10^{+2}$	$2.62 \times 10^{+3}$
F_{24}	$2.00 \times 10^{+2}$	$0.00 \times 10^{+0}$	$6.63 \times 10^{+2}$	$2.00 \times 10^{+2}$	2.90×10^{-14}	$1.10 \times 10^{+3}$	$2.64 \times 10^{+2}$	$2.23 \times 10^{+2}$	$1.84 \times 10^{+3}$
F_{25}	$3.69 \times 10^{+2}$	$2.07 \times 10^{+0}$	$6.89 \times 10^{+2}$	$\mathbf{2.09 \times 10^{+2}}$	5.27×10^{-1}	$1.19 \times 10^{+3}$	$2.15 \times 10^{+2}$	$1.05 \times 10^{+0}$	$1.98 \times 10^{+3}$

Table 6. Comparison with other algorithms for dimensions $D = 30$.

	PSO		**Tabu**		**GA**		**MUDE**	
F	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
F_1	1.98×10^{-25}	2.80×10^{-25}	$3.61 \times 10^{+5}$	$8.26 \times 10^{+3}$	$2.35 \times 10^{+3}$	$1.51 \times 10^{+3}$	$\mathbf{0.00 \times 10^{+0}}$	$0.00 \times 10^{+0}$
F_2	$3.20 \times 10^{+2}$	$3.68 \times 10^{+2}$	$1.01 \times 10^{+8}$	$2.14 \times 10^{+7}$	$3.31 \times 10^{+4}$	$1.02 \times 10^{+4}$	$\mathbf{2.44 \times 10^{-26}}$	6.20×10^{-26}
F_3	$1.03 \times 10^{+7}$	$1.03 \times 10^{+7}$	$2.51 \times 10^{+1}0$	$1.84 \times 10^{+9}$	$1.21 \times 10^{+8}$	$7.10 \times 10^{+7}$	$\mathbf{2.98 \times 10^{+2}}$	$1.19 \times 10^{+3}$
F_4	$1.26 \times 10^{+3}$	$1.42 \times 10^{+3}$	$1.16 \times 10^{+8}$	$1.75 \times 10^{+7}$	$5.89 \times 10^{+4}$	$1.61 \times 10^{+4}$	$\mathbf{5.93 \times 10^{-17}}$	2.56×10^{-16}
F_5	$1.03 \times 10^{+4}$	$2.63 \times 10^{+3}$	$7.90 \times 10^{+4}$	$2.88 \times 10^{+3}$	$1.69 \times 10^{+4}$	$3.93 \times 10^{+3}$	$\mathbf{1.67 \times 10^{-4}}$	6.30×10^{-4}
F_6	$5.14 \times 10^{+1}$	$3.58 \times 10^{+1}$	$7.21 \times 10^{+1}1$	$1.38 \times 10^{+1}1$	$7.49 \times 10^{+7}$	$1.01 \times 10^{+8}$	$\mathbf{3.70 \times 10^{+0}}$	$1.46 \times 10^{+1}$
F_7	$6.67 \times 10^{+3}$	$1.48 \times 10^{+2}$	$1.76 \times 10^{+4}$	$4.73 \times 10^{+2}$	$5.43 \times 10^{+3}$	$2.43 \times 10^{+2}$	$\mathbf{5.91 \times 10^{-3}}$	5.55×10^{-3}
F_8	$2.10 \times 10^{+1}$	3.47×10^{-2}	$2.19 \times 10^{+1}$	8.56×10^{-2}	$2.09 \times 10^{+1}$	1.04×10^{-1}	$\mathbf{2.09 \times 10^{+1}}$	4.56×10^{-2}
F_9	$5.87 \times 10^{+1}$	$3.80 \times 10^{+1}$	$1.13 \times 10^{+3}$	$2.08 \times 10^{+1}$	$1.18 \times 10^{+2}$	$2.89 \times 10^{+1}$	$\mathbf{3.55 \times 10^{-14}}$	6.55×10^{-14}
F_{10}	$1.08 \times 10^{+2}$	$4.43 \times 10^{+1}$	$2.22 \times 10^{+3}$	$3.25 \times 10^{+0}$	$3.64 \times 10^{+2}$	$4.28 \times 10^{+1}$	$\mathbf{2.40 \times 10^{+1}}$	$1.03 \times 10^{+1}$

Table 6. *Cont.*

	PSO		Tabu		GA		MUDE	
F	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
F_{11}	$2.10 \times 10^{+1}$	$4.62 \times 10^{+0}$	$6.41 \times 10^{+1}$	$4.24 \times 10^{+0}$	$3.53 \times 10^{+1}$	$2.20 \times 10^{+0}$	$\mathbf{1.74 \times 10^{+1}}$	$7.14 \times 10^{+0}$
F_{12}	$1.90 \times 10^{+4}$	$1.28 \times 10^{+4}$	$4.78 \times 10^{+6}$	$3.86 \times 10^{+5}$	$2.24 \times 10^{+5}$	$8.42 \times 10^{+4}$	$\mathbf{1.61 \times 10^{+3}}$	$1.97 \times 10^{+3}$
F_{13}	$2.36 \times 10^{+0}$	8.95×10^{-3}	$1.06 \times 10^{+5}$	$1.20 \times 10^{+5}$	$1.86 \times 10^{+1}$	$5.20 \times 10^{+0}$	$\mathbf{2.04 \times 10^{+0}}$	2.38×10^{-1}
F_{14}	$1.26 \times 10^{+1}$	1.50×10^{-1}	$1.51 \times 10^{+1}$	1.62×10^{-2}	$1.33 \times 10^{+1}$	2.68×10^{-1}	$\mathbf{1.23 \times 10^{+1}}$	2.85×10^{-1}
F_{15}	$\mathbf{3.88 \times 10^{+2}}$	$1.58 \times 10^{+2}$	$2.22 \times 10^{+3}$	$0.00 \times 10^{+0}$	$5.41 \times 10^{+2}$	$9.78 \times 10^{+1}$	$4.00 \times 10^{+2}$	$6.45 \times 10^{+1}$
F_{16}	$2.15 \times 10^{+2}$	$1.12 \times 10^{+2}$	$2.09 \times 10^{+3}$	$2.95 \times 10^{+1}$	$4.40 \times 10^{+2}$	$9.27 \times 10^{+1}$	$\mathbf{6.37 \times 10^{+1}}$	$3.66 \times 10^{+1}$
F_{17}	$2.65 \times 10^{+2}$	$1.21 \times 10^{+2}$	$1.91 \times 10^{+3}$	$3.54 \times 10^{+1}$	$5.13 \times 10^{+2}$	$8.02 \times 10^{+1}$	$\mathbf{6.22 \times 10^{+1}}$	$7.99 \times 10^{+1}$
F_{18}	$9.78 \times 10^{+2}$	$7.22 \times 10^{+1}$	$2.39 \times 10^{+3}$	$0.00 \times 10^{+0}$	$1.01 \times 10^{+3}$	$4.49 \times 10^{+1}$	$\mathbf{9.04 \times 10^{+2}}$	2.34×10^{-1}
F_{19}	$9.81 \times 10^{+2}$	$5.71 \times 10^{+1}$	$2.72 \times 10^{+3}$	$4.66 \times 10^{+2}$	$1.00 \times 10^{+3}$	$3.87 \times 10^{+1}$	$\mathbf{9.04 \times 10^{+2}}$	2.55×10^{-1}
F_{20}	$9.71 \times 10^{+2}$	$6.81 \times 10^{+1}$	$3.06 \times 10^{+3}$	$5.40 \times 10^{+2}$	$1.01 \times 10^{+3}$	$4.11 \times 10^{+1}$	$\mathbf{9.04 \times 10^{+2}}$	2.22×10^{-1}
F_{21}	$8.11 \times 10^{+2}$	$3.38 \times 10^{+2}$	$3.72 \times 10^{+3}$	$1.56 \times 10^{+2}$	$1.06 \times 10^{+3}$	$1.59 \times 10^{+2}$	$\mathbf{5.00 \times 10^{+2}}$	6.14×10^{-14}
F_{22}	$1.04 \times 10^{+3}$	$3.76 \times 10^{+1}$	$3.82 \times 10^{+3}$	$5.44 \times 10^{+0}$	$1.20 \times 10^{+3}$	$6.72 \times 10^{+1}$	$\mathbf{8.64 \times 10^{+2}}$	$1.77 \times 10^{+1}$
F_{23}	$7.27 \times 10^{+2}$	$2.82 \times 10^{+2}$	$3.53 \times 10^{+3}$	$4.19 \times 10^{+2}$	$1.08 \times 10^{+3}$	$1.77 \times 10^{+2}$	$\mathbf{5.34 \times 10^{+2}}$	2.58×10^{-13}
F_{24}	$2.83 \times 10^{+2}$	$2.87 \times 10^{+2}$	$2.81 \times 10^{+3}$	$3.06 \times 10^{+0}$	$1.22 \times 10^{+3}$	$1.91 \times 10^{+2}$	$\mathbf{2.00 \times 10^{+2}}$	2.90×10^{-14}
F_{25}	$1.74 \times 10^{+3}$	$1.65 \times 10^{+1}$	$2.15 \times 10^{+3}$	$1.35 \times 10^{+2}$	$1.76 \times 10^{+3}$	$3.33 \times 10^{+1}$	$\mathbf{2.09 \times 10^{+2}}$	5.27×10^{-1}
$-/+/\approx$	24/1/0		25/0/0		25/0/0			

6. Conclusions

In this paper, we propose a novel multiple island differential evolution algorithm, a multi-population differential evolution algorithm with uniform local search (MUDE), which improves population diversity through migration with the island. In the course of evolution, the whole population is randomly split into many sub-islands, and each sub-island carries out different strategies according to the evolution ratio. To advance DE diversity, individuals of the island are migrated through the soft-island model. Uniform local search is used to improve population exploitation. The experimental results show that MUDE is effective and efficient by comparing it with four DE variants on a set of 25 functions of CEC 2005 with three dimensions D = 10, 30, and 50. In addition, other optimization algorithms are also used to compare with the MUDE algorithm. Statistical results show that MUDE completely outperforms other algorithms, i.e., PSO, Tabu, and GA. However, MUDE has high time complexity due to population migration and local search. Overall, MUDE is a challenging algorithm because of improving population diversity and local search capabilities.

In the future, we will focus on extensive practical optimization problems, and reduce time complexity in other ways. Furthermore, another tendency is to combine DE with machine learning to improve its performance.

Author Contributions: Conceptualization, X.T. and S.-Y.S.; methodology, K.-S.S. and G.W.; writing—original draft preparation X.T. and S.-Y.S.; supervision, X.T., K.-S.S. and G.W. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was supported by Wonkwang University in 2021.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Storn, R.; Price, K. Differential Evolution–A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
2. Diab, D.M.; El Hindi, K.M. Using Differential Evolution for Fine Tuning Naïve Bayesian Classifiers and Its Application for Text Classification. *Appl. Soft Comput.* **2017**, *54*, 183–199. [CrossRef]
3. Pan, J.S.; Liu, N.; Chu, S.C. A Hybrid Differential Evolution Algorithm and Its Application in Unmanned Combat Aerial Vehicle Path Planning. *IEEE Access* **2020**, *8*, 17691–17712. [CrossRef]
4. Al-Sakkaf, A.; Mohammed Abdelkader, E.; Mahmoud, S.; Bagchi, A. Studying Energy Performance and Thermal Comfort Conditions in Heritage Buildings: A Case Study of Murabba Palace. *Sustainability* **2021**, *13*, 12250. [CrossRef]
5. Baioletti, M.; Milani, A.; Santucci, V. Variable Neighborhood Algebraic Differential Evolution: An Application to The Linear Ordering Problem with Cumulative Costs. *Inf. Sci.* **2020**, *507*, 37–52. [CrossRef]
6. Deng, W.; Xu, J.; Song, Y.; Xhao, H. Differential Evolution Algorithm with Wavelet Basis Function and Optimal Mutation Strategy for Complex Optimization Problem. *Appl. Soft Comput.* **2021**, *100*, 106724. [CrossRef]
7. Al-Dabbagh, R.D.; Neri, F.; Idris, N.; Baba, M.S. Algorithmic Design Issues in Adaptive Differential Evolution Schemes: Review and Taxonomy. *Swarm Evol. Comput.* **2018**, *43*, 284–311. [CrossRef]
8. Mohamed, A.W.; Suganthan, P.N. Real-parameter Unconstrained Optimization Based on Enhanced Fitness-adaptive Differential Evolution Algorithm with Novel Mutation. *Soft Comput.* **2018**, *22*, 3215–3235. [CrossRef]
9. Sun, G.; Li, C.; Deng, L. An adaptive regeneration framework based on search space adjustment for differential evolution. *Neural Comput. Appl.* **2021**, *33*, 9503–9519. [CrossRef]
10. Deng, W.; Shang, S.; Cai, X.; Zhao, H.; Song, Y.; Xu, J. An improved differential evolution algorithm and applications to optimization problems. *Soft Comput.* **2021**, *25*, 5277–5298. [CrossRef]
11. Deng, W.; Shang, S.; Cai, X.; Zhao, H.; Zhou, Y.; Chen, H.; Deng, W. Quantum differential evolution with cooperative coevolution framework and hybrid mutation strategy for large scale optimization. *Knowl. Based Syst.* **2021**, *224*, 107080. [CrossRef]
12. Vafashoar, R.; Meybodi, M.R. A multi-population differential evolution algorithm based on cellular learning automata and evolutionary context information for optimization in dynamic environments. *Appl. Soft Comput.* **2020**, *88*, 106009. [CrossRef]
13. Tan, X.; Lee, H.A.; Shin, S.Y. Cooperative Coevolution Differential Evolution Based on Spark for Large-Scale Optimization Problems. *J. Inf. Commun. Converg. Eng.* **2021**, *19*, 155–160. [CrossRef]

14. Alsalibi, B.; Mirjalili, S.; Abualigah, L.; Ismael yahya, R.; Gandomi, A.H. A comprehensive survey on the recent variants and applications of membrane-inspired evolutionary algorithms. *Arch. Comput. Methods Eng.* **2022**, *29*, 3041–3057. [CrossRef]
15. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [CrossRef]
16. Song, B.; Wang, Z.; Zou, L. An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve. *Appl. Soft Comput.* **2021**, *100*, 106960. [CrossRef]
17. Li, Z.; Lin, X.; Zhang, Q.; Liu, H. Evolution strategies for continuous optimization: A survey of the state-of-the-art. *Swarm Evol. Comput.* **2020**, *56*, 100694. [CrossRef]
18. Delahaye, D.; Chaimatanan, S.; Mongeau, M. Simulated Annealing: From Basics to Applications. In *Handbook of Metaheuristics*; Springer: Cham, Switzerland, 2019; pp. 1–35. [CrossRef]
19. Laguna, M. Tabu search. In *Handbook of Heuristics*; Springer: Cham, Switzerland, 2018; pp. 741–758. [CrossRef]
20. Youssef, H.; Sait, S.M.; Adiche, H. Evolutionary algorithms, simulated annealing and tabu search: A comparative study. *Eng. Appl. Artif. Intell.* **2001**, *14*, 167–181. [CrossRef]
21. Meng, Z.; Pan, J.S.; Tseng, K.K. PaDE: An Enhanced Differential Evolution Algorithm with Novel Control Parameter Adaptation Schemes for Numerical Optimization. *Knowl. Based Syst.* **2019**, *168*, 80–99. [CrossRef]
22. Pant, M.; Zaheer, H.; Garcia-Hernandez, L.; Abraham, A. Differential Evolution: A review of more than two decades of research. *Eng. Appl. Artif. Intell.* **2020**, *90*, 103479. [CrossRef]
23. Kachitvichyanukul, V. Comparison of three evolutionary algorithms: GA, PSO, and DE. *Ind. Eng. Manag. Syst.* **2012**, *11*, 215–223. [CrossRef]
24. Meng, Z.; Pan, J.S. HARD-DE: Hierarchical ARchive Based Mutation Strategy with Depth Information of Evolution for The Enhancement of Differential Evolution on Numerical Optimization. *IEEE Access* **2019**, *7*, 12832–12854. [CrossRef]
25. Ronkkonen, J.; Kukkonen, S.; Price, K.V. Real-Parameter Optimization with Differential Evolution. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2–5 September 2005; pp. 192–199. [CrossRef]
26. Gämperle, R.; Müller, S.D.; Koumoutsakos, P. A Parameter Study for Differential Evolution. Advances in Intelligent Systems, Fuzzy Systems. *Evol. Comput.* **2002**, *10*, 293–298. Available online: https://www.cse-lab.ethz.ch/wp-content/uploads/2008/04/gmk_wseas_2002.pdf (accessed on 10 October 2002).
27. Omran, M.G.H.; Salman, A.; Engelbrecht, A.P. Self-adaptive Differential Evolution. In *Lecture Notes in Computer Science, Proceedings of the International Conference on Computational and Information Science (CIS 2005), Xi'an, China, 15–19 December 2005*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 192–199. [CrossRef]
28. Abbass, H.A. The Self-adaptive Pareto Differential Evolution Algorithm. In Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02), Honolulu, HI, USA, 12–17 May 2002; pp. 831–836. [CrossRef]
29. Brest, J.; Greiner, S.; Boskovic, B.; Mernik, M.; Zumer, V. Self-adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Trans. Evol. Comput.* **2006**, *10*, 646–657. [CrossRef]
30. Zhang, J.; Sanderson, A.C. JADE: Adaptive Differential Evolution with Optional External Archive. *IEEE Trans. Evol. Comput.* **2009**, *13*, 945–958. [CrossRef]
31. Qin, A.K.; Huang, V.L.; Suganthan, P.N. Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization. *IEEE Trans. Evol. Comput.* **2008**, *13*, 398–417. [CrossRef]
32. Wang, Y.; Cai, Z.; Zhang, Q. Differential Evolution with Composite Trial Vector Generation Strategies and Control Parameters. *IEEE Trans. Evol. Comput.* **2011**, *15*, 55–66. [CrossRef]
33. Peng, H.; Wu, Z.; Deng, C. Enhancing Differential Evolution with Commensal Learning and Uniform Local Search. *Chin. J. Electron.* **2017**, *26*, 725–733. [CrossRef]
34. Shang, R.; Wang, Y.; Wang, J.; Jiao, L.; Wang, S.; Qi, L. A Multi-Population Cooperative Coevolutionary Algorithm for Multi-objective Capacitated Arc Routing Problem. *Inf. Sci.* **2014**, *277*, 609–642. [CrossRef]
35. Wu, G.; Mallipeddi, R.; Suganthan, P.N.; Wang, R.; Chen, H. Differential Evolution with Multi-population Based Ensemble of Mutation Strategies. *Inf. Sci.* **2016**, *329*, 329–345. [CrossRef]
36. Tong, L.; Dong, M.; Jing, C. An improved multi-population ensemble differential evolution. *Neurocomputing* **2018**, *290*, 130–147. [CrossRef]
37. Li, X.; Wang, L.; Jiang, Q.; Li, N. Differential evolution algorithm with multi-population cooperation and multi-strategy integration. *Neurocomputing* **2021**, *421*, 285–302. [CrossRef]
38. Akhmedova, S.; Stanovov, V.; Semenkin, E. Soft Island model for population-based optimization algorithms. In *Lecture Notes in Computer Science, Proceedings of the International Conference on Swarm Intelligence, Shanghai, China, 17–22 June 2018*; Springer: Cham, Switzerland, 2018; pp. 68–77. [CrossRef]
39. Peng, H.; Tan, X.; Deng, C.; Peng, S. SparkCUDE: A Spark-based Differential Evolution for Large-scale Global Optimisation. *Int. J. High Perform. Syst. Archit.* **2017**, *7*, 211–222. [CrossRef]
40. Fang, K.T.; Lin, D.K.J.; Winker, P.; Zhang, Y. Uniform Design: Theory and Application. *Technometrics* **2000**, *42*, 237–248. [CrossRef]

41. Suganthan, P.N.; Hansen, N.; Liang, J.J.; Deb, K.; Chen, Y.P.; Auger, A.; Tiwari, S. *Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization*; Technical Report; Nanyang Technological University: Singapore, 2005. Available online: https://scholar.google.com/scholar?as_q=Problem+Definitions+and+Evaluation+Criteria+for+the+CEC+2005+Special+Session+on+Real-parameter+Optimization&as_occt=title&hl=en&as_sdt=0%2C31 (accessed on 30 May 2005).
42. Eftimov, T.; Korošec, P. Introduction to Statistical Analysis. In *Deep Statistical Comparison for Meta-Heuristic Stochastic Optimization Algorithms*; Springer: Cham, Switzerland, 2022; pp. 23–31. [CrossRef]