Memory-Based Differential Evolution Algorithms with Self-Adaptive Parameters for Optimization Problems

Shang-Kuan Chen , Gen-Han Wu * , Yu-Hsuan Wu

Posted Date: 7 April 2025

doi: 10.20944/preprints202504.0563.v1

Keywords: differential evolution; particle swarm optimization; self-adaptive parameters; optimization

 doi:10.20944/preprints202504.0563.v1

Article

Memory-Based Differential Evolution Algorithms with Self-Adaptive Parameters for Optimization Problems

Shang-Kuan Chen [1], Gen-Han Wu [2,*] and Yu-Hsuan Wu [2]

[1] Department of Computer Science & Engineering, Yuan Ze University, Taoyuan 32003, Taiwan

[2] Department of Industrial Engineering & Engineering, Yuan Ze University, Taoyuan 32003, Taiwan

* Correspondence: genhanwu@saturn.yzu.edu.tw; Tel.: +886-3-4638800#2511

Abstract: In this study, twelve modified differential evolution algorithms with memory properties and adaptive parameters were proposed to solve the optimization problem. In the experimental process, these modified differential evolution algorithms were applied to 23 continuous test functions. Experiments show that MBDE2 and IHDE-BPSO3 are superior to the original differential evolution algorithm and its extended variants, and the best solutions can be found in most of the problems. It is inducted that the proposed improved differential evolution algorithm can adapt to most problems and obtain better results, and adding the concept of memory property is a great improvement to the capability of the proposed improved differential evolution algorithm.

Keywords: differential evolution; particle swarm optimization; self-adaptive parameters; optimization

1. Introduction

Differential evolution algorithm [1] is a fairly mainstream heuristic algorithm, which has been studied and improved by most studies [2–11]. Because the process is fast and simple, it is very suitable for solving optimization problems with less time consuming. However, the differential evolution algorithm does not guarantee that the global optimal solution can be found, so under this problem, the differential evolution algorithm has developed a variety of mutation algorithms. The mutation algorithm refers to the part that generates a random solution, such as DE/rand/1 or DE/rand/2, or changes the formula of the algorithm to form a new algorithm. To compare novel algorithms, the differential evolution algorithm and its deformation still maintain a good position in most problems, indicating that the diversity of variation in the selection of mutation algorithms is quite important. The weight-changing differential evolution algorithm used in this study has maintained good results in previous studies. The parameter settings are based on the user's past experience, and the concept of memory is rarely used in the original differential evolution algorithm. Although the original differential evolution algorithm can have good results in the average performance, it cannot get good results with the same parameter settings for various problems. Therefore, in this study, a different differential evolution algorithm is proposed to improve the concepts of adaptive and memory, and it is named the Improved Hybrid Differential Evolution Algorithm (IHDE). In this paper, the concept of particle swarm optimization algorithm is tried to influence the structure of the solution generated by the formula, so that it can achieve good results on most problems.

2. Related Works

Storn and Price [1] published a differential evolution algorithm in 1997, which is classified as an intelligent optimization algorithm. At the beginning, the differential evolution algorithm will first set the basic parameters, including group size, scaling factor, and crossover probability and so on, and

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 April 2025 doi:10.20944/preprints202504.0563.v1

generate an initial set of solutions x based on the upper and lower bounds of the given solution and according to rand (0, 1). Then, the initial solution x is mutated to generate a mutation solution, and then the structure of the solution is changed by means of crossover probability to produce a crossover solution x'. In the selected section, if a cross solution x' is better than the current solution x, the original initial solution is replaced by a cross solution, otherwise, nothing replaced. The algorithm repeats the mutation, crossover, and selection until the end of the iteration. Figure 1 shows the conceptual diagram of the differential evolution algorithm. This example takes dimension size 2 as an example, showing the outline of the algorithm and the process of generating the mutation solution, which is the result of the weighted difference of the initial value of two arbitrary selections and the addition of the third value to obtain the mutation solution. In the figure, V is the mutation solution; X is the initial solution; F is the scaling factor; G is the iteration; and each of i, r_1, r_2, r_3 is a random group with no repetitions.

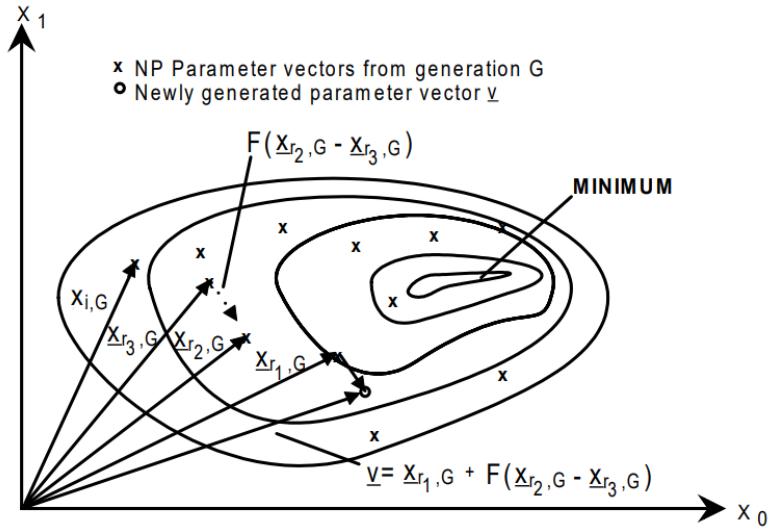

Figure 1. Conceptual diagram of differential evolution algorithm (Storn and Price [1]).

The differential evolution algorithm is also a stochastic model that simulates biological evolution, through multiple iterations and preserves those individuals who are better adapted to the environment. The advantage of differential evolution algorithms is with shorter steps compared to other algorithms. There are only four steps: initialization, mutation, crossover, and selection. Therefore, the convergence time of the algorithm is relatively less than other algorithm, but it still retains strong global convergence ability and robustness.

Particle swarm optimization (PSO) algorithm concept was proposed by two scholars, Kennedy and Eberhart [12]. At the beginning, set parameters including population size, inertia weight, acceleration constant, maximum velocity, and etc. A random swarm of particles is then generated, and their position is adjusted based on the individual experience of each particle's flight and the best experience in the past. The particles produced at the beginning can be treated as a set of solutions x, and their velocity equation v, the individual optimal position (p_{best}), and the global optimal position (g_{best}) affect the individual position of the next iteration. The algorithm then iteratively finds the best position (i.e., the best solution), and the algorithm continues until the end of the iteration or reaches the given *max_steps*. Figure 2 is a conceptual diagram of the particle swarm optimization algorithm, where k is the iteration, p_{best} is the individual optimal position, g_{best} is the global optimal position, x is the initial solution (or contemporary solution), v is the velocity, and new x and v will be generated according to the following formula.

 doi:10.20944/preprints202504.0563.v1

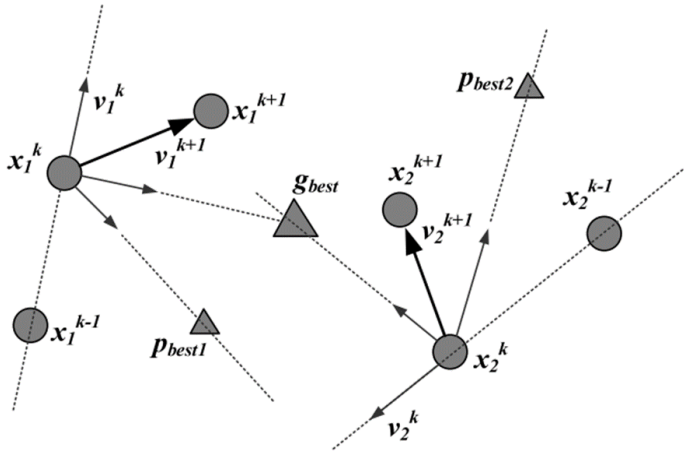

Figure 2. Conceptual diagram of the particle swarm optimization algorithm (Hizarci *et al*. [13]).

The main framework of the particle swarm optimization algorithm is to generate the initial solution and initial velocity, and then update the velocity and position according to the velocity and position formula until the end of the iteration. The formula is as follows:

Velocity update formula:

$$v_i^{t+1} = \omega \times v_i^t + c_1 \times r_1 \times (p_{i,best}^t - X_i^t) + c_2 \times r_2 \times (g_{best} - X_i^t) \quad (1)$$

where v is the velocity, t is the iteration, X is the contemporary solution, p_{best} is the individual optimal position, g_{best} is the global optimal position, and c_1 and c_2 are the acceleration constants, ω is the inertia weight, r_1, and r_2 are random values between 0 and 1.

Position update formula:

$$X_i^{t+1} = X_i^t + v_i^{t+1} \quad (2)$$

In the position update formula, a new particle position can be obtained, which is the contemporary solution for the next iteration. The speed update formula and the position update formula are repeated until the end of the iteration or until the user-set stop condition is reached.

There are many variations of differential evolution algorithms, among which Qin and Suganthan [14] proposed a differential evolution algorithm for two mutation algorithms at the same time (Self-adaptive Differential Evolution, SaDE); Huang *et al.*[15] proposed a differential evolution algorithm of the cooperative evolution (Co-evolutionary Differential Evolution, CDE). The whole part is divided into two groups with interaction, and the algorithm also affects each other when it is carried out.

Parouha and Das [16] proposed a modified differential evolution algorithm called MBDE (Memory based Hybrid Differential Evolution). In this paper, the formula of the differential evolution algorithm is added to the memory method, and the individual optimal position and the global optimal position are introduced to change the concept that the past solution of the differential evolution algorithm does not affect the future generation solution. And then, the mutation and crossover steps are changed to Swarm Mutation and Swarm Crossover, and finally the greedy selection method is improved to Elitism, which greatly improves the overall algorithm performance. The improved formula is shown as follows:

Improved Swarm Mutation:

$$V_i^{t+1} = X_i^t + \left|\frac{f(p_{i,best}^t)}{f(X_{i,worst}^t)}\right| (p_{i,best}^t - X_i^t) + \left|\frac{f(g_{best})}{f(X_{i,worst}^t)}\right| (g_{best} - X_i^t) \quad (3)$$

 doi:10.20944/preprints202504.0563.v1

In this part, the original mutation algorithm DE/rand/2 takes into account the concept of memory, where V is the mutation solution, t is the iteration number, X is the contemporary solution, p_{best} is the individual best position, g_{best} is the global best position, x_{worst} is the contemporary worst solution, and $f(\bullet)$ is the objective function.

Improved Swarm Crossover:

$$U_{i,j}^{t+1} = \begin{cases} V_{i,j}^{t+1} + \boldsymbol{rand}(0,1)\left(g_{best} - p_{i,best}^{t}\right) & if \quad rand_{ij} \le p_{cr} \\ X_{ij}^{t} + \boldsymbol{rand}(0,1)\left(g_{best} - p_{i,best}^{t}\right) & otherwise \end{cases} \tag{4}$$

The crossover algorithm also takes into account the concept of memory, where V is the mutation solution, t is the iteration number, X is the contemporary solution, p_{best} is the individual best position, g_{best} is the global best position, rand(0, 1) and $rand_{ij}$. are random real numbers between 0 and 1, and p_{cr}. is the user-defined crossover probability.

Elitism:

This part is to change the original greedy choice method to the elite choice method. This method is proposed to discard the overall algorithm diversity, because the diversity has been processed in the previous mutation step and the cross step, so there is no need to deal with it in this part. Whereas, elitism obeys the following three rules:

(1) In the final elite selection section, the initial population (or contemporary target vector) and the resulting species from the final crossover are combined.

(2) The merged populations will be sorted in ascending order according to the value of the target function.

(3) The better NP (population number) individuals are retained to move on to the next iteration, and the remaining individuals are allowed to be removed.

Chen *et al*.[17] proposed HPSO (Hybrid Particle Swarm Optimizer), in which the parameters required for the original PSO formula were brought into the concept of adaptation, and the formula was improved for generating target value and velocity of next step. The acceleration constant algorithm is based on the TVAC(Time Varying Acceleration Coefficient) commonly used in the past literature and the inertia weight is modified. The modified formula is as follows:

Velocity update formula:

$$\begin{aligned} & v_i^{t+1} = \chi(v_i^t + c_1 \times r_1 \times (p_{i,best}^t - X_i^t) + c_2 \times r_2 \times (g_{best} - X_i^t)) \\ & \chi = \frac{2}{\left|2 - \phi - \sqrt{|\phi^2 - 4 \times \phi|}\right|}, \quad \phi = c_1 + c_2 \\ & c_1 = (c_{1f} - c_{1i}) \times \frac{M_j}{M_{max}} + c_{1i}, \quad c_2 = (c_{2f} - c_{2i}) \times \frac{M_j}{M_{max}} + c_{2i} \end{aligned} \tag{5}$$

where $c_{1f} = 0.5, c_{1i} = 2.5, c_{2f} = 2.5, c_{2i} = 0.5$, χ replaces the inertia weight and further affects the individual optimal position and the proportion of the global optimal position on the velocity, v is the velocity, t is the iteration number, X is the contemporary solution, p_{best} is the individual optimal position, and g_{best} is the global optimal position.

Position update formula:

$$\begin{aligned} & X_i^{t+1} = X_i^t \times w_{ij} + v_i^{t+1} \times w'_{ij} + \rho \times \\ & g_{best} \times \psi \\ & w_{ij} = \psi = \frac{exp(f(j)/u)}{1 + exp(-f(j)/u)^{iter}}, \quad w'_{ij} = \\ & 1 - w_{ij} \end{aligned} \tag{6}$$

where u is the average value of the initial population substituting the objective function, and $f(\bullet)$ is the objective function. This improvement greatly reduces the problem of user parameter setting. For parameter adaptation, the parameter values can be changed according to iterations, so that the algorithm can produce different results in different iteration periods. The proportion of the

 doi:10.20944/preprints202504.0563.v1

individual optimal position and the global optimal position can also be changed according to the needs of the algorithm and the progress of iteration. The acceleration constant is set by the adaptive algorithm commonly used in previous literature, and the overall velocity update formula is affected by another inertia weight.

Hizarci *et al*. [13] proposed Binary Particle Swarm Optimization (BPSO), which is different from the previous method in that it sets an adaptive algorithm with a speedup constant. The acceleration constant algorithm commonly used in the past is abandoned, and the inertia weight is adaptive, so that it can reduce the impact of the speed of the previous iteration according to the iteration. For the other formulas, it is a formula that uses the original particle swarm optimization algorithm entirely. The acceleration constant adaptation and inertia weight adaptation formulas are as follows:

Acceleration constant formula:

$$c_1 = c_{1i} + 2 \times e^{-(2.2\times iter/max_{iter})^2}$$
$$c_2 = c_{1f} - 2 \times e^{-(2.2\times iter/max_{iter})^2} \quad (7)$$

where c_{1i} = 0.5, c_{1f}= 2.5, $iter$ is the iteration at that time, max_{iter} is the maximum number of iterations.

Inertia weight formula:

$$\omega(\text{iter}) = \omega_{max} - \frac{\omega_{max} - \omega_{min}}{max_{iter}} \times iter \quad (8)$$

where ω_{max} = 0.9 , and ω_{min} = 0.4.

Based on the review of the above papers, it can be found that the differential evolution algorithm with memory properties has good performance and can effectively deal with the problem of group diversity. The acceleration constant, which has a lot of influence on the velocity formula of the particle swarm optimization algorithm, is nothing more than a factor that has a great influence. In this study, we will consider applying the formula of particle swarm optimization to the differential evolution algorithm. and continue the other steps of the differential evolution algorithm to intersect to disrupt the structure of the solution. The target algorithm can have the ability to adapt to most problems and have good results.

3. The Proposed Method

In this paper, twelve modified differential evolution algorithms were proposed. The proposed algorithm can be divided into four parts, namely:

(1) Simply increase the selection group.
(2) Added additional cross-solutions based on contemporary solutions.
(3) The crossover step uses the mutated solution as the base to generate a new crossover solution.
(4) Differential evolution algorithms are improved based on the improved particle swarm algorithm.

The basic concepts and corresponding algorithms of these four classifications are described in Table 1.

Table 1. Classification of target algorithms.

	Algorithm parts	Conception
a.	Simply increase the selection group	The mutant individuals generated by the selection step and the mutation step are ranked and selected.
b.	Added additional cross-solutions based on contemporary solutions. (IHDE, IHDE_2, IHDE_2cross)	For the crossover formula, the part is changed slightly, and the new crossover solution is more inclined to be searched around the contemporary solution. In this part of the algorithm, we will select the contemporary solution, the mutation solution, the crossover solution, and the new crossover solution in the selection step.
c.	The crossover step uses the mutated solution as the base to generate a new crossover solution. (IHDE_mgi, IHDE_mgm)	The solution of the original crossover step is changed so that it tends to search near the mutated solution to generate a new crossover solution.

 doi:10.20944/preprints202504.0563.v1

d.	Differential evolution algorithms are improved based on the improved particle swarm algorithm. (IHDE_BPSO3, IHDE_BPSO4, IHDE_BPSO5, IHDE_HPSO3, IHDE_HPSO4, IHDE_HPSO5)	The velocity and position update formulas of different particle swarm algorithms are used to generate new mutation solutions and new crossover solutions corresponding to the differential evolution algorithm.

Simply increase the selecting individuals (MBDE_2): The concept is that the original MBDE algorithm has a good effect, but it is hoped that the algorithm can find more and better solutions quickly in the early stage by taking advantage of the large search range of mutation solutions. Therefore, the overall structure and formula of the original MBDE are retained. Changed the selection section so that it also takes mutants into account when selecting them.

MBDE2：

Mutation：Make use of formula (3).

Crossover：

$$\boldsymbol{U}_{i,j}^{t+1} = \begin{cases} \boldsymbol{V}_{i,j}^{t+1} + \boldsymbol{rand}(\mathbf{0},\mathbf{1})\left(\boldsymbol{g}_{best} - \boldsymbol{p}_{i,best}^{t}\right) & \boldsymbol{if} \quad \boldsymbol{rand}_{ij} \le \boldsymbol{p}_{cr} \\ \boldsymbol{X}_{ij}^{t} + \boldsymbol{rand}(\mathbf{0},\mathbf{1})\left(\boldsymbol{g}_{best} - \boldsymbol{p}_{i,best}^{t}\right) \boldsymbol{otherwise} \end{cases} \tag{9}$$

Selection：Contemporary solutions, $\boldsymbol{X}_i^t$ variant solutions $\boldsymbol{V}_i^{t+1}$ and cross-solution $\boldsymbol{U}_i^{t+1}$, to retain good NP (population) individuals to the next iteration.

The concept of IHDE, IHDE_2 and IHDE_2cross based on contemporary solutions to generate additional crossover solutions is mainly to propose a new crossover formula. This new crossover formula will be related on contemporary solution-based vectors. In this study, the mutation solution can already be used as a group of independent individuals to compete with other individuals. So the new cross-solution part doesn't have any effect on the mutated solution. Finally, the three algorithms consider different steps to choose separately.

IHDE：

Mutation：Make use of formula (3).

Crossover：Make use of formula (10) to obtain crossover solution $\boldsymbol{U2}_i^{t+1}$.

Selection：Select the mutation solution $\boldsymbol{V}_i^{t+1}$ and crossover solution $\boldsymbol{U2}_i^{t+1}$, and retain the better NP (population number) individuals to the next iteration.

IHDE - 2：

Mutation：Make use of formula (3).

Crossover：

$$\boldsymbol{U2}_{i,j}^{t+1} = \begin{cases} \boldsymbol{X}_{ij}^{t} + \boldsymbol{rand}(\mathbf{0},\mathbf{1})\left(\boldsymbol{g}_{best} - \boldsymbol{p}_{i,best}^{t}\right) & \boldsymbol{if} \quad \boldsymbol{rand}_{ij} \le \boldsymbol{p}_{cr} \\ \boldsymbol{X}_{ij}^{t} + \boldsymbol{rand}(\mathbf{0},\mathbf{1})\left(\boldsymbol{g}_{best} - \boldsymbol{X}_{i}^{t}\right) & \boldsymbol{otherwise} \end{cases} \tag{10}$$

Selection：Select the contemporary solution $\boldsymbol{X}_i^t$, mutation solution $\boldsymbol{V}_i^{t+1}$ and crossover solution $\boldsymbol{U2}_i^{t+1}$, and retain the better NP (population number) individuals to the next iteration.

IHDE - 2cross：

Mutation：Make use of formula (3).

Crossover 1：Make use of formula (9) to obtain crossover solution $\boldsymbol{U}_i^{t+1}$.

Crossover 2：Make use of formula (10) to obtain crossover solution $\boldsymbol{U2}_i^{t+1}$.

Selection：Select the contemporary solution $\boldsymbol{X}_i^t$, mutation solution $\boldsymbol{V}_i^{t+1}$ and crossover solution $\boldsymbol{U}_i^{t+1}$ and $\boldsymbol{U2}_i^{t+1}$, and retain the better NP (population number) individuals to the next iteration.

The IHDE_mgi and IHDE_mgm are based on the crossover step that is to generate in a new crossover solution based on the mutated solution. The main concept of these two algorithms is that although the mutated solution is sufficient to be used as a solution to compete with other individuals to become the next generation, its wide search range can be better if it is used to influence again with past memories at the crossover step. Therefore, these two algorithms are proposed, and these two algorithms are slightly different in the intersection part.

 doi:10.20944/preprints202504.0563.v1

IHDE - mgi：

Mutation：Make use of formula (3).

Crossover：

$$\boldsymbol{U_{i,j}^{t+1}} = \begin{cases} \boldsymbol{V_{ij}^{t+1}} + \boldsymbol{rand(0,1)}\left(\boldsymbol{g_{best}} - \boldsymbol{p_{i,best}^{t}}\right) & \boldsymbol{if} \quad \boldsymbol{rand_{ij}} \leq \boldsymbol{p_{cr}} \\ \boldsymbol{V_{ij}^{t+1}} + \boldsymbol{rand(0,1)}(\boldsymbol{g_{best}} - \boldsymbol{X_{i}^{t}}) & \boldsymbol{otherwise} \end{cases} \tag{11}$$

Selection： Select the contemporary solution $\boldsymbol{X_i^t}$, mutation solution $\boldsymbol{V_i^{t+1}}$ and crossover solution $\boldsymbol{U_i^{t+1}}$, and retain the better NP (population number) individuals to the next iteration.

IHDE - mgm：

Mutation：Make use of formula (3).

Crossover：

$$\boldsymbol{U_{i,j}^{t+1}} = \begin{cases} \boldsymbol{V_{ij}^{t+1}} + \boldsymbol{rand(0,1)}\left(\boldsymbol{g_{best}} - \boldsymbol{p_{i,best}^{t}}\right) & \boldsymbol{if} \quad \boldsymbol{rand_{ij}} \leq \boldsymbol{p_{cr}} \\ \boldsymbol{V_{ij}^{t+1}} + \boldsymbol{rand(0,1)}\left(\boldsymbol{g_{best}} - \boldsymbol{V_{i}^{t+1}}\right) & \boldsymbol{otherwise} \end{cases} \tag{12}$$

Selection： Select the contemporary solution $\boldsymbol{X_i^t}$, mutation solution $\boldsymbol{V_i^{t+1}}$ and crossover solution $\boldsymbol{U_i^{t+1}}$, and retain the better NP (population number) individuals to the next iteration.

Improved IHDE_BPSO3, IHDE_BPSO4 and IHDE_BPSO5 based on different algorithms of Binary Particle Swarm Optimization (BPSO). BPSO is used because it proposes a new adaptive acceleration constant algorithm. In the study of Hizarci *et al*. [13], this adaptive algorithm has obtained a good ranking compared with the adaptive algorithm commonly used in the previous literature of particle swarm optimization algorithms, and has shown potential effects. Therefore, the speed and position of BPSO can be used to update the formula and its adaptive algorithm to generate different mutation solutions and crossover solutions. And in two of these algorithms, new cross-solutions are generated, which disrupt the structure of the solution in a probabilistic way.

IHDE - BPSO3：

Mutation：Make use of formula (1) and (7). The parameters of the mutation part are set also using equation (7).

Crossover：Make use of formula (9).

Selection： Select the contemporary solution $\boldsymbol{X_i^t}$, mutation solution $\boldsymbol{V_i^{t+1}}$ and crossover solution $\boldsymbol{U_i^{t+1}}$, and retain the better NP (population number) individuals to the next iteration.

IHDE - BPSO4：

Mutation 1：Make use of formula (1) to obtain mutation solution $\boldsymbol{V_i^{t+1}}$.

Mutation 2：Make use of formula (2) to obtain mutation solution $\boldsymbol{V2_i^{t+1}}$. The parameters of the mutation part are set also using equations (7) and (8).

Crossover 1：Make use of formula (9) to obtain crossover solution $\boldsymbol{U_i^{t+1}}$.

Crossover 2：

$$\boldsymbol{U2_{i,j}^{t+1}} = \begin{cases} \boldsymbol{V2_i^{t+1}} & , \ \boldsymbol{rand_{ij}} \leq \boldsymbol{p_{cr1}} \\ \boldsymbol{V_{i,j}^{t+1}} + \boldsymbol{rand(0,1)}\left(\boldsymbol{g_{best}} - \boldsymbol{p_{i,best}^{t}}\right), & \boldsymbol{p_{cr1}} < \boldsymbol{rand_{ij}} \leq \boldsymbol{p_{cr2}} \\ \boldsymbol{X_{i,j}^{t}} + \boldsymbol{rand(0,1)}\left(\boldsymbol{g_{best}} - \boldsymbol{p_{i,best}^{t}}\right), & \boldsymbol{otherwise} \end{cases} \tag{13}$$

$\boldsymbol{p_{cr1}}$ is set to 0.5, because the variant solution 2 generated based on BPSO will be better, so it is hoped that the structure of the solution will be affected with a greater probability. However, p_{cr2} is set to 0.75, so that the mutated solution 1 and the contemporary solution can affect the structure of the solution with a small probability.

Selection： Select the contemporary solution $\boldsymbol{X_i^t}$, mutation solution $\boldsymbol{V2_i^{t+1}}$ and crossover solution $\boldsymbol{U_i^{t+1}}$ **and** $\boldsymbol{U2_i^{t+1}}$, and retain the better NP (population number) individuals to the next iteration.

IHDE - BPSO5：

 doi:10.20944/preprints202504.0563.v1

Mutation 1：Make use of formula (1) to obtain mutation solution $\boldsymbol{V_i^{t+1}}$.

Mutation 2：Make use of formula (2) to obtain mutation solution $\boldsymbol{V2_i^{t+1}}$. The parameters of the mutation part are set also using equations (7) and (8).

Crossover 1：Make use of formula (9) to obtain crossover solution $\boldsymbol{U_i^{t+1}}$.

Crossover 2：Make use of formula (13) to obtain crossover solution $\boldsymbol{U2_i^{t+1}}$.

Selection： Select the contemporary solution $\boldsymbol{X_i^t}$, mutation solution $\boldsymbol{V_i^{t+1}}$ and $\boldsymbol{V2_i^{t+1}}$ and crossover solution $\boldsymbol{U_i^{t+1}}$ **and** $\boldsymbol{U2_i^{t+1}}$, and retain the better NP (population number) individuals to the next iteration.

Improved IHDE_HPSO3, IHDE_HPSO4 and IHDE_HPSO5 based on different algorithms of improved particle swarm optimization. The idea of using HPSO is due to the addition of novel adaptive methods to the velocity update formula and position update formula, as well as the change from weights that only affect the previous generation of generation velocity solutions to affect the entire velocity formula. Then, there are many improvements to the adaptive parameters, so the proposed method integrates its concepts into the algorithm architecture to generate different solutions.

IHDE - HPSO3：

Mutation：Make use of formula (5).

Crossover：Make use of formula (9).

Selection： Select the contemporary solution $\boldsymbol{X_i^t}$, mutation solution $\boldsymbol{V_i^{t+1}}$ and crossover solution $\boldsymbol{U_i^{t+1}}$, and retain the better NP (population number) individuals to the next iteration.

IHDE - HPSO4：

Mutation 1：Make use of formula (5) to obtain mutation solution $\boldsymbol{V_i^{t+1}}$.

Mutation 2：Make use of formula (6) to obtain mutation solution $\boldsymbol{V2_i^{t+1}}$.

Crossover 1：Make use of formula (9) to obtain crossover solution $\boldsymbol{U_i^{t+1}}$.

Crossover 2：Make use of formula (13) to obtain crossover solution $\boldsymbol{U2_i^{t+1}}$.

Selection： Select the contemporary solution $\boldsymbol{X_i^t}$, mutation solution $\boldsymbol{V2_i^{t+1}}$ and crossover solution $\boldsymbol{U_i^{t+1}}$ **and** $\boldsymbol{U2_i^{t+1}}$, and retain the better NP (population number) individuals to the next iteration.

IHDE - HPSO5：

Mutation 1：Make use of formula (5) to obtain mutation solution $\boldsymbol{V_i^{t+1}}$.

Mutation 2：Make use of formula (6) to obtain mutation solution $\boldsymbol{V2_i^{t+1}}$.

Crossover 1：Make use of formula (9) to obtain crossover solution $\boldsymbol{U_i^{t+1}}$.

Crossover 2：Make use of formula (13) to obtain crossover solution $\boldsymbol{U2_i^{t+1}}$.

Selection： Select the contemporary solution $\boldsymbol{X_i^t}$, mutation solution $\boldsymbol{V_i^{t+1}}$ and $\boldsymbol{V2_i^{t+1}}$ and crossover solution $\boldsymbol{U_i^{t+1}}$ **and** $\boldsymbol{U2_i^{t+1}}$, and retain the better NP (population number) individuals to the next iteration.

The above 12 improvements will be compared with the DE/rand/1 proposed by Storn and Price[1] and the MBDE proposed by Parouha and Das[16], and the results will be presented in the next section.

4. Experimental Results

In subsection 4.1, it will introduce the optimization problems to be tested in this study, and run in Python 3.7, with an Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz, an Intel UHD Graphics 630 GPU, and 16.0 GB of RAM. Then, in subsections 4.2 and 4.3, the results of the differential evolution algorithm are compared with most well-known algorithms. From the results, it can be seen that the original differential evolution algorithm has good performance among many algorithms. Finally, in subsection 4.4, the comparison between the improved differential evolution algorithm and the original differential evolution algorithm and its variants is presented.

4.1. Benchmark Functions

In order to test the performance of the proposed improved differential evolution algorithm, 23 test functions used in the study of Abualigah *et al.*[18] were used to test the performance of the

 doi:10.20944/preprints202504.0563.v1

original differential evolution algorithm to be improved, as shown in Table 2~4. These test functions contain unimodal test functions, multimodal test functions, and fixed-dimension multimodal test functions.

Table 2. Unimodal test functions.

Function	Description	Dimensions	Range	f_{min}
F1	$f(x)=\sum_{i=1}^{n} x_i^2$	30, 100, 500, 1000	[−100,100]	0
F2	$f(x)=\sum_{i=0}^{n} \lvert x_i\rvert+\prod_{i=0}^{n} \lvert x_i\rvert$	30, 100, 500, 1000	[−10,10]	0
F3	$f(x)=\sum_{i=1}^{d}(\sum_{j=1}^{i} x_j)^2$	30, 100, 500, 1000	[−100,100]	0
F4	$f(x)=max_i\{\lvert x_i\rvert, 1\le i\le n\}$	30, 100, 500, 1000	[−100,100]	0
F5	$f(x)=\sum_{i=1}^{n-1}[100(x_i^2-x_{i+1})^2+(1-x_i)^2]$	30, 100, 500, 1000	[−30,30]	0
F6	$f(x)=\sum_{i=1}^{n}([x_i+0.5])^2$	30, 100, 500, 1000	[−100,100]	0
F7	$f(x)=\sum_{i=0}^{n} ix_i^4+\text{random}[0,1)$	30, 100, 500, 1000	[−128,128]	0

The unimodal test functions F1~F7 will be tested at dimension 30 and set 30 with the initial group number, set the number of iterations to 500, and test the above test settings with 30 average results, and the range of each function is shown in Table 2.

Table 3. Multimodal test functions.

Function	Description	Dimensions	Range	f_{min}
F8	$f(x)=\sum_{i=1}^{n}(-x_i sin(\sqrt{\lvert x_i\rvert}))$	30, 100, 500, 1000	[−500,500]	−418.9829 ×n
F9	$f(x)=\sum_{i=1}^{n}[x_i^2-10cos(2\pi x_i)+10]$	30, 100, 500, 1000	[−5.12,5.12]	0
F10	$f(x)=-20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-exp(\frac{1}{n}\sum_{i=1}^{n}cos(2\pi x_i))+20+e$	30, 100, 500, 1000	[−32,32]	0
F11	$f(x)=1+\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}cos(\frac{x_i}{\sqrt{i}})$	30, 100, 500, 1000	[−600,600]	0
F12	$f(x)=\frac{\pi}{n}\{10sin(\pi y_1)\}+\sum_{i=1}^{n-1}(y_i-1)^2[1+10sin^2(\pi y_{i+1})+\sum_{i=1}^{n}u(x_i,10,100,4)]$, where $y_i=1+\frac{x_i+1}{4}$, $u(x_i,a,k,m)\begin{cases}K(x_i-a)^m & \text{if } x_i>a\\ 0 & -a\le x_i\ge a\\ K(-x_i-a)^m & -a\le x_i\end{cases}$	30, 100, 500, 1000	[−50,50]	0
F13	$f(x)=0.1(sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+sin^2(3\pi x_i+1)]+(x_n-1)^2 1+sin^2(2\pi x_n))+\sum_{i=1}^{n}u(x_i,5,100,4)$	30, 100, 500, 1000	[−50,50]	0

The multimodal test functions F8 to F13 will be tested in 30 dimensions, the initial population number of 30, the number of iterations 500, and the average of 30 tests will be used as the basic settings, and the range are shown in Table 3.

Table 4. Fixed-dimension multimodal test functions.

Function	Description	Dimensions	Range	f_{min}
F14	$f(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})}\right)^{-1}$	2	[−65,65]	1
F15	$f(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right]^2$	4	[−5,5]	0.00030
F16	$f(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	[−5,5]	−1.0316
F17	$f(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10(1-\frac{1}{8\pi})cos\ x_1+10$	2	[−5,5]	0.398
F18	$f(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2)^2\times(18-32x_i+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	[−2,2]	3
F19	$f(x)=-\sum_{i=1}^{4}c_i exp\left(-\sum_{i=1}^{3}a_{ij}(x_j-p_{ij})^2\right)$	3	[−1,2]	−3.86
F20	$f(x)=-\sum_{i=1}^{4}c_i exp\left(-\sum_{i=1}^{6}a_{ij}(x_j-p_{ij})^2\right)$	6	[0,1]	−.32
F21	$f(x)=-\sum_{i=1}^{5}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	[0,1]	−10.1532
F22	$f(x)=-\sum_{i=1}^{7}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	[0,1]	−10.4028
F23	$f(x)=-\sum_{i=1}^{10}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	[0,1]	−10.5363

The fixed-dimension multimodal functions from F14 to F23 are set to the initial population of 30, the number of iterations is 500, and the average of 30 tests is set as the basic settings, and its dimensions and ranges are shown in Table 4.

 doi:10.20944/preprints202504.0563.v1

All kinds of test functions in unimodal, multimodal and fixed-dimension multimodal test functions can detect the convergence speed of the algorithm, the ability of the algorithm to jump out of the part and the convergence ability of the algorithm as a whole, and most of these test functions belong to the test functions with high search difficulty, so the prototype of most well-known algorithms may not be able to find the best value, even the emerging variants of the algorithm are the same. The algorithm in this study will test the formulas of the improved differential evolution algorithm and its adaptive parameter settings, and the parameters will no longer be set in a random or fixed way, and can obtain good results in these test functions, so that the performance of the algorithm can be effectively improved.

In the unimodal, multimodal and fixed-dimension multimodal test functions, the convergence speed of the algorithm, the ability of the algorithm to jump out of the locality, and the convergence ability of the algorithm as a whole can be detected. Most of these test functions are difficult to search, so prototypes of most well-known algorithms do not necessarily find optimal values, even for emerging variants of algorithms. In recent years, some scholars have been proposing new algorithms, various algorithm variants, or changing their parameter settings to cope with these complex test functions. The proposed algorithm will test the formulas of the improved differential evolution algorithm and its adaptive parameter settings. The experimental results show that good results can be obtained in these test functions, and the performance of the algorithm can be effectively improved.

4.2. Algorithm and Parameter Settings Compared with the Original Differential Evolution Algorithm

This subsection introduces the basic parameter settings of the original differential evolution algorithm and 11 well-known algorithms to be improved in this study. With the exception of the original differential evolution algorithm, all parameter settings were set using the ones used in the literature presented by Abualigah *et al*. [18]. These algorithms will be used in the 23 test functions described in subsection 4.1, which are as follows:

- Particle Swarm Optimization, PSO
- Cuckoo Search Algorithm, CS
- Biogeography-based Optimization, BBO
- Differential Evolution, DE
- Gravitational Search Algorithm, GSA
- Firefly Algorithm, FA
- Genetic Algorithm, GA
- Moth-Flame Optimization, MFO
- Grey Wolf Optimizer, GWO
- Bat Algorithm, BAT
- Flower Pollination Algorithm, FPA
- Arithmetic Optimization Algorithm, AOA

The improved differential evolution algorithm changes the crossover probability to 0.1 and compares it with the differential evolution algorithm with a crossover probability of 0.5 used by Abualigah *et al*.[18]. The reason for choosing 0.1 is that 0.1 is often compared in the literature on related differential evolution algorithms. This experiment will be set to an average of 30 runs, 500 iterations, and 30 in the F1~F13 dimension. The algorithm is set to both differential evolution algorithms with the scaling factor F being 0.5. The results of the comparison are presented in Table 5.

Table 5. Comparison of 30 times of differential evolution algorithm with crossover probability of 0.1 and 0.5.

F	DE (P_{cr} = 0.1)	The DE used by Abualigah *et al.*[18] (P_{cr} = 0.5)
F1	1.38E-03	1.33E-03
F2	1.88E-04	6.83E-03

 doi:10.20944/preprints202504.0563.v1

F3	1.59E-01	3.97E+04
F4	4.19E+01	1.15E+01
F5	1.03E-01E	1.06E+02
F6	1.08E-04	1.44E-03
F7	1.35E+01	5.24E-02
F8	-1.26E+04	-6.82E+03
F9	3.34E+01	1.58E+02
F10	1.04E-02	1.21E-02
F11	1.00E+00	3.52E-02
F12	1.04E-06	2.25E-05
F13	1.09E-13	9.12E-03
F14	9.98E-01	1.23E+00
F15	1.55E-03	5.63E-04
F16	-1.03E+00	-1.03E+00
F17	3.98E-01	3.98E-01
F18	3.00E+00	3.00E+00
F19	-2.01E+00	-3.86E+00
F20	-3.27E+00	-3.27E+00
F21	-5.05E+00	-8.65E+00
F22	-5.08E+00	-9.75E+00
F23	-5.12E+00	-1.05E+00
No1 Rank	15	12

Note: No.1 Rank is the number of times of getting the first place.

Table 5 shows when the crossover probability P_{cr} = 0.1 is used in this study, the effect is better. The parameter settings for the above 11 algorithms are shown in Table 6.

Table 6. Parameter settings of the comparison algorithm（Abualigah *et al.* [18]）.

Algorithm	Parameter	Value
PSO	Topology	Fully connected
	Cognitive and social constant	(C1, C2) 2, 2
	Inertia weight	Linear reduction from 0.9 to 0.1
	Velocity limit	10% of dimension range
CS	p_a	0.25
	Probability of modifying a habitat	1
BBO	Probability limits of immigrations	[0, 1]
	Size of each step	1
	I and E	1
	Probability of mutation	0.005
DE	Scaling factor	0.5
	Crossover probability	0.1
GSA	Alpha, G0, Rnorm, Rpower	20, 100, 2, 1
FA	α	0.5
	β	0.2
	γ	1
GA	Type	Real coded
	Selection	Roulette wheel (Proportionate)
	Crossover	Whole arithmetic (Probability = 0.8, α = [−0.5, 1.5])
	Mutation	Gaussian (Probability = 0.05)
MFO	a	[−2 −1]
	b	1
GWO	Convergence parameter (a)	Linear reduction from 2 to 0
BAT	Q_{min}	0
	Q_{max}	2
	A	0.5
	r	0.5
FPA	Probability switch p	0.8
AOA	α	5
	μ	0.5

 doi:10.20944/preprints202504.0563.v1

The above 11 algorithms will be tested and compared with the original differential evolution algorithm, where the mutation algorithm of the differential evolution algorithm is DE/rand/1, and the basic parameters for the test problem are set as follows:

$x_{min,j}$ and $x_{max,j}$: Set to be the upper and lower limits of each test function.

Population NP : Set to be 30.

The above are the parameter settings of the original differential evolution algorithm. These settings will be applied to each test function 30 times and averaged, the iteration is set to 500 times, and the dimension settings are limited according to each function (30 for F1~F13). The results of the application of the original differential evolution algorithm with other comparative algorithms are described in Section 4.3.

4.3. The Original Differential Evolution Algorithm Was Applied to the Results Comparison of Test Functions

This section will apply the original differential evolution algorithm to the three types of test functions described in Section 4.1 and compare it with the algorithm described in Section 4.2. The results obtained by applying the original differential evolution algorithm used in this study to the problem will be compared with the results of other algorithms in Abualigah *et al.*[18]. The table shows how each algorithm performs in each type of test function. The Gravitational Search Algorithm (GSA) [19] cannot be used in the fixed-dimensional multimodal test function (F14~F23), so the results are displayed with "-" and the ranking is set to the last place. The comparison results are shown in Table 7~9, and the gray bottom part is marked to mean the best value.

Table 7 shows the results of the differential evolution algorithm compared to 11 other different algorithms. It is the best of the 12 methods in F5 and F6, F1 is 4th and F2 and F3 are 3rd, and F4 and F7 are 7th and 11th respectively, and the results are not good.

Table 7. Twelve improved differential evolution algorithms are applied to the unimodal functions .

F	M	DE	GA	PSO	BBO	FPA	GWO	BAT	FA	CS	MFO	GSA	AOA
F1	AVE	1.38E-03	1.03E+03	1.83E+04	7.59E+01	2.01E+13	**1.18E-27**	6.59E+04	7.11E-03	9.06E-04	1.01E+03	6.08E+02	6.67E-07
	Rank	4	9	10	6	12	**1**	11	5	3	8	7	2
F2	AVE	1.88E-04	2.47E+01	3.58E+02	1.36E-03	3.22E+01	9.71E-17	2.71E+08	4.34E-01	1.49E-01	3.19E+01	2.27E+01	**0.00E+00**
	Rank	3	8	11	4	10	2	12	6	5	9	7	**1**
F3	AVE	1.59E-01	2.65E+04	4.05E+04	1.21E+04	1.41E+03	5.12E-05	1.38E+05	1.66E+03	2.10E-01	2.34E+04	1.35E+05	**6.87E-06**
	Rank	3	9	10	7	5	2	12	6	4	8	11	**1**
F4	AVE	4.19E+01	5.17E+01	4.39E+01	3.02E+01	2.38E+01	**1.24E-06**	8.51E+01	1.11E-01	9.65E-02	7.00E+01	7.87E+01	1.40E-03
	Rank	7	9	8	6	5	**1**	12	4	3	10	11	2
F5	AVE	**1.03E-01**	1.95E+04	1.96E+07	1.82E+03	3.17E+05	2.70E+01	2.10E+08	7.97E+01	2.76E+01	7.35E+03	7.41E+02	2.49E+01
	Rank	**1**	9	11	7	10	3	12	5	4	8	6	2
F6	AVE	**1.08E-04**	9.01E+02	1.87E+04	6.71E+01	1.70E+03	8.44E-01	6.69E+04	6.94E-03	3.13E-03	2.68E+03	3.08E+03	3.47E-04
	Rank	**1**	7	11	6	8	5	12	4	3	9	10	2
F7	AVE	1.35E+01	1.91E-01	1.07E+01	2.91E-03	3.44E-01	1.70E-03	4.57E+01	6.62E-02	7.29E-02	4.50E+00	1.12E-01	**3.92E-06**
	Rank	11	7	10	3	8	2	12	4	5	9	6	**1**
N01 Rank		2	0	0	0	0	2	0	0	0	0	0	**3**
Average Rank		4.2857	8.2857	10.1429	5.5714	8.2857	2.2857	11.8571	4.8571	3.8571	8.7143	8.2857	**1.5714**

Note: No.1 Rank is the number of times of getting the first place; Average Rank is the average of the rankings.

From the results of Table 8, it can be seen that the performance of the original differential evolution algorithm in the multimodal test function, F8, F12 and F13 are all first, while F10 is the third, F9 and F11 are poor, ranking 7th and 6th respectively, and the average performance in F8~F13 is good.

Table 8. Twelve improved differential evolution algorithms are applied to the multimodal test function.

 doi:10.20944/preprints202504.0563.v1

F	M	DE	GA	PSO	BBO	FPA	GWO	BAT	FA	CS	MFO	GSA	AOA
F8	AVE	**-1.26E+04**	**-1.26E+04**	-3.86E+03	-1.24E+04	-6.45E+03	-5.91E+03	-2.33E+03	-5.85E+03	-5.19E+01	-8.48E+03	-2.35E+03	-1.22E+04
	Rank	**1**	**1**	9	3	6	7	11	8	12	5	10	4
F9	AVE	3.34E+01	9.04E+00	2.87E+02	**0.00E+00**	1.82E+02	2.19E+00	1.92E+02	3.82E+01	1.51E+01	1.59E+02	3.10E+01	3.42E-07
	Rank	7	4	12	**1**	10	3	11	8	5	9	6	2
F10	AVE	1.04E-02	1.36E+01	1.75E+01	2.13E+00	7.14E+00	1.03E-03	1.92E+01	4.58E-02	3.29E-02	1.74E+01	3.74E+00	**8.88E-16**
	Rank	3	9	11	6	8	2	12	5	4	10	7	**1**
F11	AVE	1.00E+00	1.01E+01	1.70E+02	1.46E+00	1.73E+01	4.76E-03	6.01E+02	4.23E-03	4.29E-05	3.10E+01	4.86E-01	**0.00E+00**
	Rank	6	8	11	7	9	4	12	3	2	10	5	**1**
F12	AVE	**1.04E-06**	4.77E+00	1.51E+07	6.68E-01	3.05E+02	4.83E-02	4.71E+08	3.13E-04	5.57E-05	2.46E+02	4.63E-01	4.28E-06
	Rank	**1**	8	11	7	10	5	12	4	3	9	6	2
F13	AVE	**1.09E-13**	1.52E+01	5.73E+07	1.82E+00	9.59E+04	5.96E-01	9.40E+08	2.08E-03	8.19E-03	2.73E+07	7.61E+00	3.10E-01
	Rank	**1**	8	11	6	9	5	12	2	3	10	7	4
No1 Rank		3	1	0	1	0	0	0	0	0	0	0	2
Average Rank		3.1667	6.3333	10.8333	5.0000	8.6667	4.3333	11.6667	5.0000	4.8333	8.8333	6.8333	2.3333

Note: No.1 Rank is the number of times of getting the first place; Average Rank is the average of the rankings.

Table 9 shows the results of the differential evolution algorithm applied to the fixed-dimension multimodal test function. The results show that the differential evolution algorithm can achieve the first place in F14, F16, F17 and F18, while F15 and F20 are both fifth, and F19, F21, F22 and F23 are 11th, 9th, 11th and 8th, respectively. It can be seen that there is still a lot of room for improvement of differential evolution algorithm in the fixed-dimensional multimodal test problem.

Table 9. Twelve improved differential evolution algorithms are applied to fixed-dimensional multimodal test functions.

F	M	DE	GA	PSO	BBO	FPA	GWO	BAT	FA	CS	MFO	GSA	AOA
F14	AVE	**9.98E-01**	**9.98E-01**	1.39E+00	**9.98E-01**	**9.98E-01**	4.17E+00	1.27E+01	3.51E+00	1.27E+01	2.74E+00	-	**9.98E-01**
	Rank	**1**	**1**	6	**1**	**1**	9	10	8	10	7	12	**1**
F15	AVE	1.55E-03	3.33E-02	1.61E-03	1.66E-02	6.88E-04	6.24E-03	3.00E-02	1.01E-03	3.13E-04	2.35E-03	-	**3.12E-04**
	Rank	5	11	6	9	3	8	10	4	2	7	12	**1**
F16	AVE	**-1.03E+00**	-3.78E-01	**-1.03E+00**	-8.30E-01	**-1.03E+00**	**-1.03E+00**	-6.87E-01	**-1.03E+00**	**-1.03E+00**	**-1.03E+00**	-	**-1.03E+00**
	Rank	**1**	11	**1**	9	**1**	**1**	10	**1**	**1**	**1**	12	**1**
F17	AVE	**3.98E-01**	5.24E-01	4.00E-01	5.49E-01	**3.98E-01**	**3.98E-01**	**3.98E-01**	**3.98E-01**	**3.98E-01**	**3.98E-01**	-	**3.98E-01**
	Rank	**1**	10	9	11	**1**	**1**	**1**	**1**	**1**	**1**	12	**1**
F18	AVE	**3.00E+00**	**3.00E+00**	3.10E+00	**3.00E+00**	**3.00E+00**	**3.00E+00**	1.47E+01	**3.00E+00**	**3.00E+00**	**3.00E+00**	-	**3.00E+00**
	Rank	**1**	**1**	10	**1**	**1**	**1**	11	**1**	**1**	**1**	12	**1**
F19	AVE	-2.01E+00	-3.42E+00	**-3.86E+00**	-3.78E+00	**-3.86E+00**	**-3.86E+00**	-3.84E+00	**3.86E+00**	**3.86E+00**	**3.86E+00**	-	**-3.86E+00**
	Rank	11	10	**1**	9	**1**	**1**	8	**1**	**1**	**1**	12	**1**
F20	AVE	-3.27E+00	-1.61E+00	-3.11E+00	-2.71E+00	-3.30E+00	-3.26E+00	-3.25E+00	-3.28E+00	**-3.32E+00**	-3.24E+00	-	**-3.32E+00**
	Rank	5	11	9	10	3	6	7	4	**1**	8	12	**1**
F21	AVE	-5.05E+00	-6.66E+00	-4.15E+00	-8.32E+00	-5.22E+00	-8.64E+00	-4.27E+00	-7.67E+00	-5.06E+00	-6.89E+00	-	**-8.85E+00**
	Rank	9	6	11	3	7	2	10	4	8	5	12	**1**
F22	AVE	-5.08E+00	-5.58E+00	-6.01E+00	-9.38E+00	-5.34E+00	**-1.04E+01**	-5.61E+00	-9.64E+00	-5.09E+00	-8.26E+00	-	**-1.04E+01**
	Rank	11	8	6	4	9	**1**	7	3	10	5	12	**1**
F23	AVE	-5.12E+00	-4.70E+00	-4.72E+00	-6.24E+00	-5.29E+00	-1.01E+01	-3.97E+00	-9.75E+00	-5.13E+00	-7.66E+00	-	**-1.05E+01**
	Rank	8	10	9	5	6	2	11	3	7	4	12	**1**
No1 Rank		4	2	2	2	5	5	1	4	5	4	0	**10**
Average Rank		5.30	7.90	6.80	6.20	3.30	3.20	8.50	3.00	4.20	4.00	12.00	**1.00**

Note: No.1 Rank is the number of times of getting the first place; Average Rank is the average of the rankings.

As can be seen from Table 7~9, the Arithmetic Optimization Algorithm (AOA)[18] has the best average results of the three types of test functions. The Differential Evolution (DE) algorithm in this section can achieve the first place in the multimodal type test function for many times. Compared with other algorithms, the average performance is ranked in the upper middle of the performance in all types of test functions. Table 10 shows the number of No.1 rankings and the average ranking, and the results are shown in Table 10.

Table 10. The number of first rankings and average rankings of each algorithm.

Algorithm	Total Average Rank	Total No1 Rank

 doi:10.20944/preprints202504.0563.v1

AOA	1.5217	15
GWO	3.1739	7
FA	4.1739	4
CS	4.2174	5
DE	4.6522	9
BBO	5.8696	3
FPA	6.3478	5
MFO	6.5217	4
GA	7.5217	2
PSO	8.7391	2
GSA	9.6957	0
BAT	10.0870	1

Note: No.1 Rank is the number of times of getting the first place; Average Rank is the average of the rankings.

From the above table, it can be seen that the differential evolution algorithm can perform well compared with other well-known algorithms. It can be seen why most scholars want to use the differential evolution algorithm as the basis, and take advantage of its short steps and simple formulas to make changes to the algorithm.

4.4. Comparison of the Results of Twelve Improved Differential Evolution Algorithms Applied to the Test Functions

In this subsection, the improved differential evolution algorithm is applied to the three types of test functions described in Section 4.1. It can be seen how each algorithm and the Twelve improved differential evolution algorithms proposed in this study perform in various types of test functions. The comparison results are shown in Tables 11~13, in which the gray bottom part is the best value.

In Table 11, it can be seen that the algorithm of cross-step and variant solution-based variation, IHDE-mbi performed best in all seven optimization test questions. It is better than the original differential evolution algorithm (DE) and the memory hybrid differential evolution algorithm (MBDE).

Table 11. Twelve improved differential evolution algorithms, DE and MBDE are applied to the Unimodal test functions.

F	M	DE	MBDE	MBDE2	IHDE2	IHDE-mbi	IHDE-mbm	IHDE-2cross	IHDE	IHDE-BPSO3	IHDE-BPSO4	IHDE-HPSO3	IHDE-HPSO4	IHDE-BPSO5	IHDE-HPSO5
F1	AVE	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Rank	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F2	AVE	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Rank	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F3	AVE	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Rank	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F4	AVE	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Rank	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F5	AVE	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Rank	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F6	AVE	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Rank	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F7	AVE	7.72E-02	1.20E-04	5.84E-05	1.48E-04	**1.92E-05**	7.46E-04	3.53E-04	2.42E-04	1.46E-04	7.42E-04	3.46E-04	2.32E-04	7.66E-04	3.23E-04
	Rank	14	3	2	5	**1**	12	10	7	4	11	9	6	13	8
No1 Rank		6	6	6	6	7	6	6	6	6	6	6		6	6
Average Rank		2.85	1.28	1.14	1.57	1.00	2.57	2.28	1.85	1.42	2.42	2.14	1.71	2.71	2.00

Note: No.1 Rank is the number of times of getting the first place; Average Rank is the average of the rankings.

Table 12 shows that although the original differential evolution algorithm (DE) performed poorly in F9, it ranked first the most in the six multimodal test functions. However, IHDE-BPSO3, which is based on different algorithms of particle swarm optimization and improved, has the best

 doi:10.20944/preprints202504.0563.v1

average performance. Most of the twelve new methods proposed are better than the original differential evolution algorithm and the hybrid differential evolution algorithm based on memory improvement.

Table 12. Twelve improved differential evolution algorithms, DE and MBDE are applied to the Multimodal test functions.

F	M	DE	MBDE	MBDE2	IHDE2	IHDE-mbi	IHDE-mbm	IHDE-2cross	IHDE	IHDE-BPSO3	IHDE-BPSO4	IHDE-HPSO3	IHDE-HPSO4	IHDE-BPSO5	IHDE-HPSO5
F8	AVE	-1.26E+04	-1.26E+04	-1.26E+04	-1.26E+04	-1.26E+04	-1.26E+04	-1.26E+04	-1.26E+04	-1.26E+04	-1.26E+04	-1.26E+04	-1.26E+04	-1.26E+04	-1.26E+04
	Rank	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F9	AVE	2.03E+01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Rank	14	1	1	1	1	1	1	1	1	1	1	1	1	1
F10	AVE	8.91E-02	8.91E-02	8.91E-02	8.91E-02	8.91E-02	8.91E-02	8.91E-02	8.91E-02	8.91E-02	8.91E-02	8.91E-02	8.91E-02	8.91E-02	8.91E-02
	Rank	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F11	AVE	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Rank	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F12	AVE	1.60E-36	4.14E-29	1.41E-31	3.93E-31	2.64E-14	3.70E-16	2.26E-30	5.65E-31	6.28E-32	1.27E-30	1.41E-31	8.55E-05	1.41E-31	5.65E-31
	Rank	1	11	3	6	13	12	10	7	2	9	3	14	3	7
F13	AVE	8.39E-32	6.50E-29	6.21E-30	1.89E-29	1.39E-29	1.25E-28	1.63E-29	9.66E-30	1.59E-30	1.63E-29	5.29E-07	9.53E-08	2.03E-29	2.22E-06
	Rank	1	10	3	8	5	11	6	4	2	6	13	12	9	14
No1 Rank		5	4	4	4	4	4	4	4	4	4	4	4	4	4
Average Rank		3.16	4.16	1.66	3.00	3.66	4.50	3.33	2.50	1.33	3.16	3.33	5.00	2.66	4.16

Note: No.1 Rank is the number of times of getting the first place; Average Rank is the average of the rankings.

In the fixed-dimensional multimodal test problem, the memory mixed difference evolution algorithm (MBDE) can be used to obtain the first place in all of the problems in this part of the problem at the same time as the seven new methods proposed in this study. The seven methods are MBDE2, IHDE2, IHDE-2cross, IHDE-BPSO3, IHDE-BPSO4, IHDE-BPSO5, and IHDE-HPSO5. Therefore, it can be found that most of the proposed algorithms and MBDE are more suitable for fixed-dimensional multimodal optimization problems. The results are presented in Table 13.

Table 13. Twelve improved differential evolution algorithms, DE and MBDE are applied to fixed-dimensional multimodal test functions.

F	M	DE	MBDE	MBDE2	IHDE2	IHDE-mbi	IHDE-mbm	IHDE-2cross	IHDE	IHDE-BPSO3	IHDE-BPSO4	IHDE-HPSO3	IHDE-HPSO4	IHDE-BPSO5	IHDE-HPSO5
F14	AVE	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01
	Rank	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F15	AVE	1.17E-03	3.08E-04	3.08E-04	3.08E-04	3.08E-04	4.32E-04	3.08E-04	3.08E-04	3.08E-04	3.08E-04	5.42E-04	3.08E-04	3.08E-04	3.08E-04
	Rank	14	1	1	1	1	12	1	1	1	1	13	1	1	1
F16	AVE	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00
	Rank	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F17	AVE	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01
	Rank	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F18	AVE	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00
	Rank	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F21	AVE	-1.02E+01	-1.02E+01	-1.02E+01	-1.02E+01	-4.65E+00	-1.02E+01	-1.02E+01	-2.68E+00	-1.02E+01	-1.02E+01	-5.06E+00	-5.06E+00	-1.02E+01	-1.02E+01
	Rank	1	1	1	1	13	1	1	14	1	1	12	11	1	1
F22	AVE	-1.04E+01	-1.04E+01	-1.04E+01	-1.04E+01	-1.03E+01	-1.04E+01	-1.04E+01	-2.77E+00	-1.04E+01	-1.04E+01	-4.97E+00	-1.04E+01	-1.04E+01	-1.04E+01
	Rank	1	1	1	1	12	1	1	14	1	1	13	1	1	1
F23	AVE	-1.05E+01	-1.05E+01	-1.05E+01	-1.05E+01	-2.92E+00	-1.04E+01	-1.05E+01	-1.05E+01	-1.05E+01	-1.05E+01	-1.05E+01	-5.13E+00	-1.05E+01	-1.05E+01
	Rank	1	1	1	1	14	12	1	1	1	1	1	13	1	1
No1 Rank		7	8	8	8	5	6	8	6	8	8	5	6	8	8
Average Rank		2.62	1.00	1.00	1.00	5.50	3.75	1.00	4.25	1.00	1.00	5.37	3.75	1.00	1.00

Note: No.1 Rank is the number of times of getting the first place; Average Rank is the average of the rankings.

In summary, IHDE-mgm can achieve the best performance in DE and MBDE, as well as twelve improved differential evolution algorithms in this study, when applied to the unimodal test functions. It can be seen that for the unimodal test functions, the solution obtained from the variation step of

 doi:10.20944/preprints202504.0563.v1

the original MBDE is more suitable for the unimodal optimization problem when applied to the crossover step of the double variation.

However, the algorithm IHDE-BPSO3 can obtain better average performance compared with other algorithms in the multimodal test functions. It indicates that it has a memory property variation step with adaptive acceleration and inertia weight, and has a good adaptability to the multimodal test functions.

In the fixed-dimensional multimodal test function, most of the twelve algorithm algorithms proposed in this study can achieve the best solution of this type of optimization problem. The MBDE proposed by Parouha and Das [16] can achieve the same performance for the fixed-dimensional multimodal test functions, which means that its memory parameters can have a very good impact and performance on the algorithm.

The multimodal optimization problem can test the ability of the algorithm to jump out of the local optima. It can be seen that the twelve improve DE algorithm proposed in this study has a good ability to deal with complex problems. Table 14 summarizes the overall results.

Table 14. The number of first places and the average ranking of twelve improved differential evolution algorithms and DE and MBDE.

Algorithm	Total Average Rank	Total No1 Rank
IHDE-BPSO3	1.2381	18
MBDE2	1.2381	18
IHDE2	1.7619	18
MBDE	2.0000	18
IHDE-BPSO5	2.0476	18
IHDE-2cross	2.0952	18
IHDE-BPSO4	2.0952	18
IHDE-HPSO5	2.2381	18
DE	2.8571	18
IHDE	2.9524	16
IHDE-HPSO4	3.4286	16
IHDE-mbi	3.4762	16
IHDE-mbm	3.5714	16
IHDE-HPSO3	3.7143	15

Note: No.1 Rank is the number of times of getting the first place; Average Rank is the average of the rankings.

From the above table, the test performance of MBDE2 and IHDE-BPSO3 proposed in this study is very good for the overall 21 optimization problems. Out of 21 questions, almost every question was ranked first. It shows that the proposed twelve improved differential evolution algorithms have great efficiency for the algorithm as a whole by adding the concept of memory and adaptive parameter setting. The performance of the multimodal optimization problem can be stably exerted, which also indicates that the addition of the concept is helpful for complex and difficult to search problems. The proposed twelve improved differential evolution algorithms can also find the differences between the BPSO concept and the HPSO concept, and the adaptive algorithms for the acceleration constants also show obvious differences in performance.

5. Conclusions

Optimization problems have always arisen with the rapid development of information technology and engineering problems. Solving problems quickly, accurately and effectively has always been a desirable goal. However, each algorithm has some drawbacks for different problems, such as long executing time and poor results. Moreover, if the parameters are set in a random way or by user-definition, then the problems will be not stable, which will affect the effect of solving the problem.

In this study, an improved differential evolution algorithm is proposed to solve the optimization problem. The main steps and formulas are changed, and the concept of memory

 doi:10.20944/preprints202504.0563.v1

properties and parametric adaptive algorithms are considered. It enables it to solve the problem of solution stability during the solution process. This improved differential evolution algorithm can retain the simplicity of the original differential evolution algorithm. It outperforms the original algorithm and other well-known algorithms in terms of results.

Author Contributions: Conceptualization, S.-K.C. and G.-H.W.; methodology, S.-K.C. and G.-H.W.; software, Y.-H.W.; validation, S.-K.C., G.-H.W. and Y.-H.W.; formal analysis, G.-H.W. and Y.-H.W.; investigation, S.-K.C.; resources, G.-H.W. and Y.-H.W.; data curation, G.H.W. and Y.-H.W.; writing—original draft preparation, S.-K.C.; writing—review and editing, S.-K.C.; visualization, G.-H.W.; supervision, G.-H.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts 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(4), 341–359.
2. Abbass, H. A.; Sarker, R. The pareto differential evolution algorithm. *Int. J. Artif. Intell. Tools* 2002, 11(4), 531–552.
3. Das, S.; Abraham, A.; Konar, A. Automatic clustering using an improved differential evolution algorithm. *IEEE Trans. Syst. Man Cybern. A: Syst. Hum.* 2007, 38(1), 218–237.
4. Das, S.; Konar, A.; Chakraborty, U. K. Two improved differential evolution schemes for faster global search. *In Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, Washington DC, U.S.A., June 25-29, 2005, 991–998.
5. Das, S.; Mullick, S. S.; Suganthan, P. N. Recent advances in differential evolution–an updated survey. *Swarm Evol. Comput.* 2016, 27, 1–30.
6. Draa, A.; Bouzoubia, S.; Boukhalfa, I. A sinusoidal differential evolution algorithm for numerical optimisation. *Appl. Soft Comput.* 2015, 27, 99–126.
7. Eltaeib, T.; Mahmood, A. Differential evolution: a survey and analysis. *Appl. Sci.* 2018, 8(10), 1945.
8. Tao, S.; Yang, Y.; Zhao, R.; Todo, H.; Tang, Z. Competitive elimination improved differential evolution for wind farm layout optimization problems. *Mathematics* 2024, 12(23), 3762.
9. Nguyen, V.-T.; Tran, V.-M.; Bui, N.-T. Self-adaptive differential evolution with gauss distribution for optimal mechanism design. *Appl. Sci.* 2023, 13(10), 6284.
10. Chao, M.; Zhang, M.; Zhang, Q.; Jiang, Z.; Zhou, L. A two-stage adaptive differential evolution algorithm with accompanying populations. *Mathematics* 2025, 13(3), 440.
11. Sui, X.; Chu, S.-C.; Pan, J.-S.; Luo, H. Parallel compact differential evolution for optimization applied to image segmentation. *Appl. Sci.* 2020, 10(6), 2195.
12. Kennedy, J.; Eberhart, R. Particle swarm optimization. *In Proceedings of ICNN'95-International Conference on Neural networks*, Perth, WA, Australia, Nov. 27-Dec 1, 1995, 1942–1948.
13. Hizarci, H.; Demirel, O.; Turkay, B. E. Distribution network reconfiguration using time-varying acceleration coefficient assisted binary particle swarm optimization. *Eng. Sci. Technol. Int. J.* 2022, 35, 10C30.
14. Qin, A. K.; Suganthan, P. N. Self-adaptive differential evolution algorithm for numerical optimization. *In Proceedings of 2005 IEEE Congress on Evolutionary Computation*, Edinburgh, Scotland, U.K., Sep. 2-5, 2005, 1785–1791.
15. Huang, F. Z.; Wang, L.; He, Q. An effective co-evolutionary differential evolution for constrained optimization. *Appl. Math. Comput.* 2007, 186(1), 340–356.

 doi:10.20944/preprints202504.0563.v1

16. Parouha, R. P.; Das, K. N. A robust memory based hybrid differential evolution for continuous optimization problem. *Knowl.-Based Syst.* 2016, 103, 118–131.
17. Chen, K.; Zhou, F.; Yin, L.; Wang, S.; Wang, Y.; Wan, F. A hybrid particle swarm optimizer with sine cosine acceleration coefficients. *Inf. Sci.* 2018, 422, 218–241.
18. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A. H. The arithmetic optimization algorithm. *Comput. Methods Appl. Mech. Eng.* 2021, 376, 113609.
19. Rashedi, E.; Rashedi, E.; Nezamabadi-Pour, H. A comprehensive survey on gravitational search algorithm. *Swarm Evol. Comput.* 2018, 41, 140–158.