Differential Evolution Algorithm Enhancement: A Novel Entropy Guided Parameter Adaptation Mechanism

Juncheng Guo, and Yonghong Zhang

Abstract—**This study presents an enhanced Differential Evolution (DE) framework, referred to as EPDE, which introduces a novel entropy-driven parameter adaptation strategy. The entropy guided mechanism in EPDE is designed to tackle key limitations of basic DE algorithms, including their sensitivity to parameter configurations and vulnerability to local optima. The entropy guided parameter adaptation adjusts the mutation factor and crossover rate according to the population entropy, maintaining population diversity and accelerating convergence. The mutation with double strategy selection provides an extra exploration direction. Benchmark functions and a practical application of parameter extraction for photovoltaic (PV) systems are used to test the performance of the original DE and several improved algorithms. Empirical findings demonstrate the superior performance of EPDE over DE and the other variants in terms of convergence speed and solution quality, with lower mean, standard deviation values, and better best scores. This indicates that EPDE is a more effective algorithm for solving optimization problems.**

Index Terms—**Differential Evolution, Entropy, Opposite based individual, Optimization**

I. Introduction

As a population based stochastic optimization technique, Differential Evolution (DE) has proven highly effective in solving global optimization problems [1]. Its applications span diverse domains, including neural network training [2], object tracking [3], and industrial control systems [4]. However, like other evolutionary algorithms, DE also faces challenges. One of the main issues is its sensitivity to parameter settings. The fixed parameter values in traditional DE may not be suitable for different optimization problems, leading to slow convergence and the possibility of getting trapped in local optima [5]. To overcome these limitations, substantial efforts have been devoted to DE's adaptability and convergence properties through methodological innovations, such as [6,7]. Among these DE variants, some studies focus on modifying the mutation strategies [8], while others aim at adapting the control parameters [9].

In this paper, we propose an enhanced DE algorithm EPDE that adapts the parameters based on the population entropy and introduces an opposite-based individual strategy.

Manuscript received Feb 18, 2025; revised Jul 31, 2025.

This work was supported in part by the Scientific and Technological Research Projects in Henan Province (242102210146); the Key Scientific Research Projects in Universities in Henan Province (24B110006); the Research Project of Henan Polytechnic (2024J046); the Basic Research Program of Natural Science of Shaanxi Province (2024JC-YBMS-003); the Key Cultivation Project of Xianyang Normal University (XSYK21044); the Research Project on Teaching Reform of Xianyang Normal University (2023C117); the 14th Five Year Plan for Education Science in Shaanxi Province (SGH23Y2506).

Juncheng Guo is an associate professor at the Basic Education Department, Henan Polytechnic, Zhengzhou, 450046, China (email: acheng2090@126.com)

Yonghong Zhang is an associate professor at the School of Mathematics and Statistics, Xianyang Normal University, Xianyang, 712000, China (email: zhangyonghong09@126.com)

II. Literature Review

A. Traditional Differential Evolution Algorithm

The DE algorithm was first introduced by Storn and Price [1], which initializes a population of candidate solutions randomly within a given search space. The algorithmic framework of DE primarily encompasses three core phases: perturbation through mutation, recombination via crossover, and competitive selection. In the mutation step, a mutant vector is generated through adding the weighted difference between two randomly selected vectors to a third vector. The crossover operation combines the mutant vector and the target vector to create a trial vector. Finally, the selection operation determines whether the trial vector replaces the target vector based on their fitness values.

B. Parameter Adaptation in DE

Brest and his colleagues introduced the JDE algorithm, in which the mutation factor $\boldsymbol{F}$ and the crossover rate $\boldsymbol{CR}$ are adaptively modified throughout the optimization procedure [9]. They employed a self adaptation approach, generating new values of $\boldsymbol{F}$ and $\boldsymbol{CR}$ randomly for every individual within each generation. In an effort to tackle unconstrained optimization challenges more effectively, Zhao et al. [10] continuously adjusted $\boldsymbol{F}$ and $\boldsymbol{CR}$ by leveraging the Cauchy distribution and the normal distribution. Aiming to simplify the selection of control parameters and improve an existing mutation strategy, Meng et al. adopted an adaptive learning mechanism known as PLAM to fine-tune the control parameters [11].

C. Mutation Strategy Improvements

Price et al. put forward multiple mutation strategies within the DE framework, including DE/rand/1, DE/best/1, and DE/rand-to-best/1 [8]. Each of these strategies exhibits distinct features in terms of their exploration and exploitation capabilities. For instance, DE/rand/1 demonstrates a robust exploration capacity, whereas DE/best/1 pays emphasis on

exploitation.

Zhang and Sanderson introduced the JaDE algorithm, which incorporates a memory-based mechanism for the adaptive selection of mutation strategies [12]. In attempt to enhance performance, Qin et al. devised a self-adaptive variant of DE, named SaDE [13]. Their approach involved constructing a candidate pool of mutation strategies to achieve this improvement.

Wang et al., on the other hand, took the DE/rand/2 strategy as a basis and designed a refined mutation scheme, which was accomplished by implementing an elite archive strategy [14].

D. Hybridization with Other Algorithms

Sethanan and Pitakaso proposed a hybrid variant of DE by integrating three distinct local search techniques: the shifting algorithm, the exchange algorithm, and the k-variable move algorithm. They then applied this hybrid DE algorithm to tackle the generalized assignment problem [15].

Cai et al. merged a one-step k-means clustering approach with the DE algorithm, giving rise to a novel DE variant named CDE [16].

Furthermore, Wang et al. developed a self-adaptive DE algorithm named DEPSO, which is grounded in the Particle Swarm Optimization (PSO) algorithm. In DEPSO, a modified mutation strategy of DE and an enhanced mutation strategy of PSO are combined. This combination aims to boost the overall performance of the DE algorithm [17].

E. The Main Contributions

The paper presents an enhanced DE algorithm with two main innovations.

First, an entropy-guided parameter adaptation mechanism adjusts the mutation factor F and crossover rate CR based on population entropy, maintaining diversity and accelerating convergence.

Second, a novel mutation with double-strategy selection, including generating individuals in a DE/best/1-like way and opposite-based individuals, provides an extra exploration direction to avoid local optima. The experiments show that the improved algorithm outperforms the traditional DE in convergence speed and solution quality, having lower mean and standard deviation values and better best scores.

III. Basic DE

A. Initialization

The DE algorithm starts by initializing a population of N candidate solutions in a n-dimensional search space. Each solution $X_i=(x_{i1},x_{i2},\cdots,x_{in})$ is randomly generated within the lower bound $L=(l_1,l_2,\cdots,l_n)$ and the upper bound $U=(u_1,u_2,\cdots,u_n)$:

$$x_{ik}=l_k+(u_k-l_k)\times rand, \tag{1}$$

where $rand$ is a random number in the range [0,1], i=1,2,⋯,N, and $k=1,2,\cdots,n$.

B. Mutation

The mutation operation creates a mutant vector v_i. The most common mutation strategy is DE/rand/1:

$$v_i=X_a+F\times(X_b-X_c), \tag{2}$$

where a, b, and c are randomly selected indices different from i, and F is the mutation factor, which controls the step size of the mutation.

C. Crossover

The crossover operation combines the mutant vector v_i and the target vector X_i to generate a trial vector u_i. The binomial crossover is often used:

$$u_{ij}=\begin{cases} v_{ij} & \text{if } rand\le CR \text{ or } j=j_{rand} \\ x_{ij} & \text{otherwise} \end{cases} \tag{3}$$

where CR is the crossover rate, j_{rand} is a randomly selected index in the range $[1,n]$, and $j=1,2,\cdots,n$.

D. Selection

The selection operation compares the fitness values of the trial vector u_i and the target vector X_i. If the fitness of u_i is better than that of X_i, then X_i is replaced by u_i in the next generation.

IV. Improved DE Algorithm

A. Entropy-guided Parameter Adaptation

We calculate the population entropy in each generation. The population entropy serves as a quantitative measure of solution diversity within the evolutionary algorithm. A lower entropy value indicates a more homogeneous population, while a higher entropy value means a more diverse. The details to compute the entropy of the population is given below.

First, we calculate the entropy of each dimension of the population. For a given dimension k:

- Discretize the data of the current dimension into nb bins (a predefined value).
- Calculate the probabilities of each bin:

$$p=\frac{h}{N}, \tag{4}$$

where h is the frequency distribution of the data in the current dimension and N is the population size.

- Calculate the entropy of the current dimension:

$$H_k=\sum_{p>0}-p\times\log_2(p). \tag{5}$$

Then, we calculate the average entropy of the population:

$$HV=\frac{\sum_{k=1}^{n}H_k}{n}. \tag{6}$$

Based on the entropy value, we adjust the mutation factor F and the crossover rate CR:

Case 1: If $HV<0.2$ (low entropy), we increase the adjustment amount of F and CR:

Set $F = F + \epsilon_1$, where ϵ_1 is a random value with zero mean and standard deviation $2\sigma_F$ (σ_F is a preset adjustment parameter for F).

Set $CR = CR + \epsilon_2$, where ϵ_2 is a random value with zero mean and standard deviation $2\sigma_{CR}$ (σ_{CR} is a preset adjustment parameter for CR).

Case 2: If $HV > 0.8$ (high entropy), we decrease the adjustment amount of F and CR:

Set $F = F + \epsilon_3$, where ϵ_3 is a random value with zero mean and standard deviation $0.5\sigma_F$.

Set $CR = CR + \epsilon_4$, where ϵ_4 is a random value with zero mean and standard deviation $0.5\sigma_{CR}$.

Case 3: If $0.2 \le HV \le 0.8$ (medium entropy), we perform standard adjustment:

Set $F = F + \epsilon_5$, where ϵ_5 is a random value with zero mean and standard deviation σ_F.

Set $CR = CR + \epsilon_6$, where ϵ_6 is a random value with zero mean and standard deviation σ_{CR}.

We also limit F and CR to be in the reasonable ranges: $F \in [0.1, 1]$ and $CR \in [0, 1]$.

B. Mutation with Double-strategy Selection

The traditional DE/best/1 uses the best solution of the population for mutation, which accelerates convergence but raises the risk of premature convergence to local optima. To overcome this drawback, in this section, we design a mutation with double-strategy selection.

Strategy 1: For each individual in the population, a mutant vector is designed using DE/best/1 mutation strategy:

$$v_i = bestX + F \times (X_b - X_c). \quad (7)$$

where X_b and X_c are two different solutions.

Strategy 2: For each individual in the population, we generate an opposite-based individual as follows:

$$v_i = L + (U - bestX), \quad (8)$$

where $bestX$ is the current best solution.

Based on strategies 1 and 2, we randomly select v_i from them. And then, If $f(v_i) < bestScore$ (where $bestScore$ is the current best fitness value), we update the best solution.

The pseudo-code of the Algorithm EPDE is given as follows:

Step 1: Initialize population X of size N in the search space. Set initial values $MaxGeneration$, F, CR, σ_F, σ_{CR}.

Step 2: Evaluate the fitness of each individual in X and find the best solution $bestX$ with fitness $bestScore$.

Step 3: For $t = 1 : MaxGeneration$

Step 4: Calculate the entropy HV of the population X by (4)-(6). Adjust F and CR based on H_V as described above.

Step 5: For $i = 1 : N$

Step 6: Compute u_i according to (3), (7) and (8), and update X_i with u_i.

Step 7: End for

Step 8: Update $bestX$.

Step 9: End for

Step 10: Output the final result.

The flow chart is given in Figure 1.

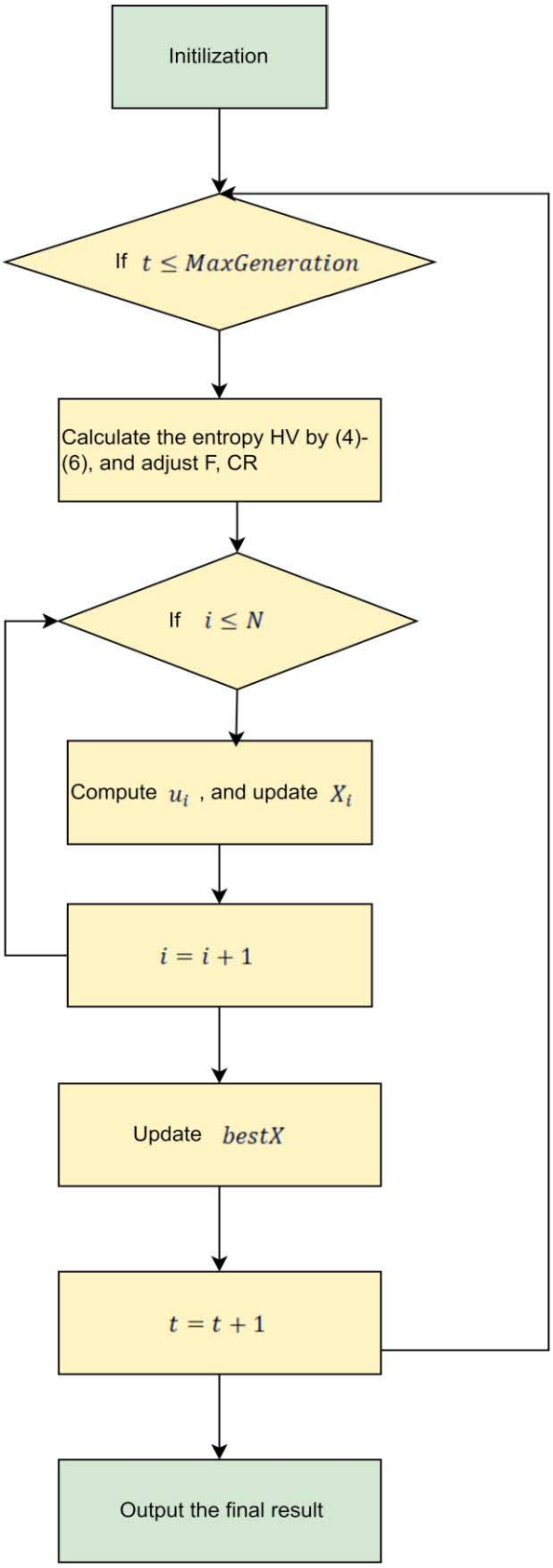

Figure 1:The flow chart of EPDE

V. Experimental Results

A. Experimental Setup

We use 14 benchmark functions to test the performance of the original DE algorithm and the proposed algorithm EPDE. The benchmark functions include Sphere function, Rosenbrock function, Rastrigin function, etc. (Suganthan et al., 2005). The parameters for the experiments are set as follows: the population size $N = 50$, the maximum number of generations $MaxGeneration = 3000$. Both F and CR are set to 0.5. The standard deviations for adjusting F and CR are $\sigma_F = 0.1$ and $\sigma_{CR} = 0.1$. When calculating the population entropy, the number of bins for data discretization in each dimension is set to $nb = 10$. The runtime is set to 30 independent runs for each algorithm. The performance of EPDE is compared with some DE variants, including DE [1], JaDE [12], SHADE [18], WED [19] and BESD [20].

B. Performance Metrics

We use the mean, standard deviation, best score, and global minimum values as performance metrics. The mean value represents the average performance of the algorithm, the standard deviation measures the stability of the algorithm, the best score represents the best solution found in all runs, and the global minimum value is the theoretical optimal value of the benchmark function.

C. Experimental Results and Analysis

1) Benchmark functions test: 12 benchmark functions are selected to test the performance of EPDE. The details of these benchmark functions are presented in TABLE I.

TABLE I
Benchmark test functions

Function	$f_1 = \sum_{i=1}^{n} x_i^2$
Range	[-100,100]
n	30
Optimal Value	0
Function	$f_2 = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
Range	[-10,10]
n	30
Optimal Value	0
Function	$f_3 = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$
Range	[-100,100]
n	30
Optimal Value	0
Function	$f_4 = \max\{\lvert x_i \rvert, 1 \le i \le n\}$
Range	[-100,100]
n	30
Optimal Value	0
Function	$f_5 = \sum_{i=1}^{n} i x_i^2$
Range	[-100,100]
n	30
Optimal Value	0
Function	$f_6 = \sum_{i=1}^{n} i x_i^4$
Range	[-100,100]
n	30
Optimal Value	0
Function	$f_7 = \sum_{i=1}^{n} \lvert x_i \rvert^{(i+1)}$
Range	[-100,100]
n	30
Optimal Value	0
Function	$f_8 = \sum_{i=1}^{n} (10^6)^{\frac{i-1}{n-1}} x_i^2$
Range	[-100,100]
n	30
Optimal Value	0
Function	$f_9 = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$
Range	[-100,100]
n	30
Optimal Value	0
Function	$f_{10} = \sum_{i=1}^{n} i x_i^4 + random[0,1)$
Range	[-100,100]
n	30
Optimal Value	0
Function	$f_{11} = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$
Range	[-100,100]
n	30
Optimal Value	0
Function	$f_{12} = -20\exp(-0.2\sqrt{\sum_{i=1}^{n} x_i^2/n}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$
Range	[-100,100]
n	30
Optimal Value	0
Function	$f_{13} = 0.5 + \frac{\sin(\sqrt{\sum_{i-1}^{n} x_i^2})^2 - 0.5}{(1 + 0.001\sum_{i-1}^{n} x_i^2)^2}$
Range	[-100,100]
n	30
Optimal Value	0
Function	$f_{14} = \sum_{i=1}^{n} (y_i^2 - 10\cos(2\pi y_i) + 10)$

	$\begin{cases} y_i = x_i, & \lvert x_i \rvert < \frac{1}{2} \\ y_i = \frac{\lvert 2x_i \rvert}{2}, & \lvert x_i \rvert \geq \frac{1}{2} \end{cases}$
Range	[-50,50]
n	30
Optimal Value	0

The comparison results of these algorithms are given in TABLE II, including Min, Mean and Std as the performance Metrics.

TABLE II
COMPARISON RESULTS OF DIFFERENT ALGORITHMS ON TEST FUNCTIONS

Function		f_1
DE	Min	2.44E-42
	Mean	6.78.E-42
	SD	5.79E-42
JaDE	Min	0
	Mean	8.82E-135
	SD	2.79E-134
SHADE	Min	0
	Mean	7.76E-125
	SD	2.45E-124
WDE	Min	4.74E+01
	Mean	7.73E+01
	SD	1.97E+01
BESD	Min	3.45E-65
	Mean	1.94E-05
	SD	6.14E-05
EPDE	Min	0
	Mean	0
	SD	0
Function		f_2
DE	Min	2.24E-26
	Mean	5.56E-26
	SD	1.76E-26
JaDE	Min	0
	Mean	1.73E-66
	SD	5.49E-66
SHADE	Min	0
	Mean	1.36E-64
	SD	4.32E-64
WDE	Min	3.17E+00
	Mean	4.04E+00
	SD	4.35E-01
BESD	Min	6.80E-36
	Mean	3.99E-04
	SD	1.26E-03
EPDE	Min	0
	Mean	0
	SD	0
Function		f_3
DE	Min	1.20E+04
	Mean	1.73E+04
	SD	2.91E+04
JaDE	Min	1.03E-38
	Mean	1.35E-11
	SD	4.29E-11
SHADE	Min	6.29E-298
	Mean	3.01E-27
	SD	9.54E-27
WDE	Min	3.23E+03
	Mean	5.11E+03
	SD	1.08E+03
BESD	Min	2.30E-17
	Mean	3.75E-01
	SD	1.16E+00
EPDE	Min	0
	Mean	5.29E-111
	SD	1.67E-110
Function		f_4
DE	Min	6.34E-04
	Mean	1.08E-03
	SD	3.26E-04
JaDE	Min	1.74E-03
	Mean	1.74E-03
	SD	2.61E-12
SHADE	Min	2.88E-165
	Mean	1.41E-16
	SD	4.46E-16
WDE	Min	2.76E+01
	Mean	3.36E+01
	SD	2.99E+00
BESD	Min	2.19E+00
	Mean	4.09E-02
	SD	1.16E-01
EPDE	Min	0
	Mean	3.63E-170
	SD	0
Function		f_5
DE	Min	1.11E-43
	Mean	8.54E-43
	SD	6.74E-43
JaDE	Min	0
	Mean	3.19E-135
	SD	1.01E-134
SHADE	Min	1.62E-127
	Mean	5.12E-127
	SD	3.21E-128
WDE	Min	5.31E+00
	Mean	9.72E+00
	SD	2.38E+00
BESD	Min	1.03E-65
	Mean	3.31E-01
	SD	1.04E-05
EPDE	Min	0
	Mean	1.71E-185
	SD	0
Function		f_6
DE	Min	9.20E-68
	Mean	2.74E-66
	SD	3.18E-66
JaDE	Min	0
	Mean	2.27E-237
	SD	0

SHADE	Min	0
	Mean	2.27E-237
	SD	0
WDE	Min	4.43E-04
	Mean	8.87E-04
	SD	3.45E-04
BESD	Min	1.49E-111
	Mean	2.17E-14
	SD	6.86E-14
EPDE	Min	0
	Mean	0
	SD	0
Function	f_7	
DE	Min	1.09E-114
	Mean	2.65E-112
	SD	5.20E-112
JaDE	Min	2.40E-39
	Mean	5.32E-33
	SD	1.68E-32
SHADE	Min	0
	Mean	2.95E-261
	SD	0
WDE	Min	3.73E-11
	Mean	1.69E-10
	SD	1.06E-10
BESD	Min	7.15E-173
	Mean	9.88E-29
	SD	3.12E-28
EPDE	Min	0
	Mean	0
	SD	0
Function	f_8	
DE	Min	1.81E-39
	Mean	5.69E-39
	SD	3.60E-39
JaDE	Min	0
	Mean	2.08E-131
	SD	6.59E-131
SHADE	Min	0
	Mean	5.48E-121
	SD	1.73E-120
WDE	Min	3.34E+04
	Mean	5.98E+04
	SD	1.38E+04
BESD	Min	1.58E-60
	Mean	7.30E-02
	SD	2.31E-01
EPDE	Min	0
	Mean	0
	SD	0
Function	f_9	
DE	Min	0
	Mean	0
	SD	0
JaDE	Min	0
	Mean	0
	SD	0
SHADE	Min	0
	Mean	0
	SD	0
WDE	Min	0
	Mean	0
	SD	0
BESD	Min	0
	Mean	0
	SD	0
EPDE	Min	0
	Mean	0
	SD	0
Function	f_{10}	
DE	Min	6.37E-03
	Mean	8.09E-03
	SD	1.07E-03
JaDE	Min	1.55E-02
	Mean	2.57E-02
	SD	1.39E-02
SHADE	Min	7.32E-04
	Mean	1.04E-03
	SD	3.88E-04
WDE	Min	2.94E-01
	Mean	4.11E-01
	SD	6.05E-02
BESD	Min	5.47E-04
	Mean	8.42E-04
	SD	5.93E-04
EPDE	Min	1.05E-04
	Mean	2.81E-04
	SD	1.56E-04
Function	f_{11}	
DE	Min	3.72E+01
	Mean	4.93E+01
	SD	6.44E+00
JaDE	Min	1.99E+00
	Mean	2.58E+01
	SD	2.89E+01
SHADE	Min	0
	Mean	0
	SD	0
WDE	Min	4.45E+01
	Mean	5.04E+01
	SD	3.55E+00
BESD	Min	8.28E-02
	Mean	9.54E+00
	SD	1.28E+01
EPDE	Min	0
	Mean	0
	SD	0
Function	f_{12}	
DE	Min	2.66E-15
	Mean	5.15E-15
	SD	1.72E-15
JaDE	Min	2.66E-15
	Mean	2.66E-15
	SD	0
SHADE	Min	2.66E-15
	Mean	2.66E-15
	SD	0
WDE	Min	5.09E+00

	Mean	5.85E+00
	SD	4.24E-01
BESD	Min	6.22E-15
	Mean	3.32E-04
	SD	1.05E-03
EPDE	Min	-8.88E-16
	Mean	-1.78E-16
	SD	1.50E-15
Function	f_{13}	
DE	Min	3.72E-02
	Mean	3.72E-02
	SD	7.07E-07
JaDE	Min	3.72E-02
	Mean	3.72E-02
	SD	1.15E-11
SHADE	Min	7.82E-02
	Mean	7.82E-02
	SD	1.63E-15
WDE	Min	4.94E-01
	Mean	4.95E-01
	SD	1.30E-03
BESD	Min	7.82E-02
	Mean	9.45E-02
	SD	2.82E-02
EPDE	Min	9.70E-03
	Mean	9.70E-03
	SD	1.05E-15
Function	f_{14}	
DE	Min	3.76E+01
	Mean	3.99E+01
	SD	2.09E+00
JaDE	Min	3.17E+01
	Mean	3.60E+01
	SD	3.78E+00
SHADE	Min	0
	Mean	0
	SD	0
WDE	Min	3.25E+01
	Mean	3.89E+01
	SD	6.27E+00
BESD	Min	1.38E+01
	Mean	2.16E+01
	SD	8.82E+00
EPDE	Min	0
	Mean	0
	SD	0

For most of the 14 benchmark functions, the EPDE algorithm achieved the best (lowest) minimum values. For example, in functions $f_1 - f_9$, f_{11} and f_{14}, EPDE reached 0 as the Min value, while traditional DE and some other algorithms like WDE had much larger Min values. These results indicate that EPDE has a strong ability to find the optimal or near optimal solutions in these functions.

JaDE and SHADE demonstrated competitive performance in achieving minimum values, but not as comprehensively as EPDE. For instance, in f_3, SHADE had a very low Min value of 6.29E-29, but in other functions, its performance was not as outstanding as EPDE did.

The EPDE algorithm generally had lower Mean values compared to the traditional DE algorithm across the 14 test functions. In the function f_2, the mean fitness value of EPDE was significantly lower than that of the traditional DE algorithm. This implies that, on average, EPDE can find better solutions than the traditional DE.

Among all the algorithms, WDE consistently exhibited higher mean values, suggesting that it may not be as effective as other algorithms in finding good quality solutions on average. For example, in f_4, the Mean value of WDE was 3.36E+01, much higher than those of EPDE, JaDE, and SHADE.

A lower Std indicates that the algorithm's performance is less variable, and it can consistently find solutions close to the mean value. EPDE usually had lower Std values, which means it was more stable in different runs. In f_1, the Std value of EPDE was 0, while that of traditional DE was 5.79E-42. Some algorithms like BESD had relatively high Std values in some functions, indicating that its performance varied greatly in different runs. For example, in f_2, the Std value of BESD was 1.26E-03, which was higher than that of EPDE, showing that BESD's solutions were less consistent.

Overall, among the six algorithms tested on 14 functions, EPDE demonstrated excellent performance. It outperformed the traditional DE algorithm in terms of convergence speed and solution quality, as shown by its lower Min, Mean, and Std values in most cases. JaDE and SHADE also showed good performance in some aspects, but EPDE was more consistent across different functions. WDE and BESD generally had worse performance, with higher Mean and Std values in many functions.

In conclusion, EPDE is a more effective algorithm for solving the optimization problems represented by these benchmark functions, with strong abilities in finding optimal solutions, obtaining good quality solutions on average, and maintaining stability in different runs.

The average rankings of six algorithms (DE, JaDE, SHADE, WDE, BESD, EPDE) across fourteen test function are shown in Figure 2.

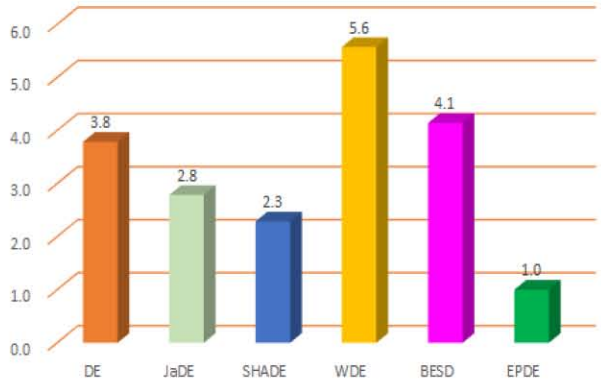

Figure 2 The average rank of different algorithms

From Figure 2, we can see that, EPDE has the lowest average ranking of 1.0, meaning it performs the best among these algorithms on average. SHADE follows closely with an average ranking of 2.3, also demonstrating excellent performance. JaDE has an average ranking of 2.8, indicating relatively good performance as well.

DE has an average ranking of 3.8, showing moderate performance. BESD has an average ranking of 4.1, with performance that is not as strong as the previous ones. WDE has the highest average ranking of 5.6, suggesting it performs the worst among these six algorithms on average.

In summary, EPDE is the top performing algorithm, while WDE lags behind. SHADE, JaDE, DE, and BESD have performance levels that decrease in the order mentioned based on their average rankings across the test functions.

2) The practical application: In this subsection, a more substantial and practical real-world engineering application is utilized to show the superiority of EPDE in comparison to other highly regarded rivals. Photovoltaic (PV) systems play a crucial role in the realm of new energy, given their capacity to directly transform solar energy into electrical power. Consequently, devising a precise and efficient model for PV systems through the extraction of their parameters from measured current-voltage data represents a vital undertaking. The principal parameters of PV systems are extracted using EPDE, along with six other methods. There exist three conventional PV models: the single diode model (SDM), the double diode model (DDM), and the PV module model (PVMM). The corresponding circuit configurations for these three models are depicted in Figure 3.

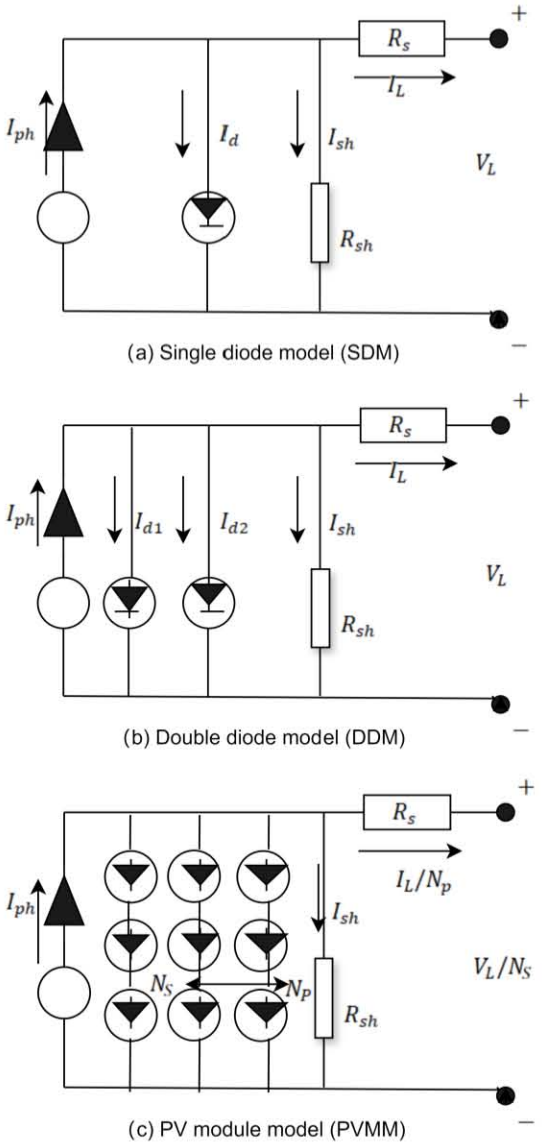

Figure 3 Equivalent circuit diagrams for photovoltaic cells

As depicted in Figure 3(a), the extraction of SDM hinges on five essential parameters: the photocurrent source (I_{ph}), shunt resistance (R_{sh}), series resistance (R_s), ideal factor of diodes (n), and reverse saturation current (I_{sd}). The mathematical relationships these parameters adhere to can be summarized below.

The output current I_L is defined as:

$$I_L = I_{ph} - I_{sh} - I_d, \tag{9}$$

Here, the diode current is computed using:

$$I_d = I_{sd} \cdot [\exp(\frac{(I_L \cdot R_s + V_L) \cdot q}{T \cdot n \cdot k}) - 1],$$

and the shunt current I_{sh} is calculated as:

$$I_{sh} = \frac{I_L \cdot R_s + V_L}{R_{sh}},$$

Combining these equations, the expression for I_L in the context of SDM is:

$$I_L = I_{ph} - \frac{I_L R_s + V_L}{R_{sh}} - I_{sd}[\exp(\frac{(I_L \cdot R_s + V_L) \cdot q}{T \cdot n \cdot k}) - 1] \tag{10}$$

Unlike the SDM, the DDM accounts for recombination losses occurring within semiconductor depletion regions. Its corresponding circuit diagram is presented in Figure 3(b). The formula for computing I_L in DDM is:

$$\begin{aligned} I_L &= I_{ph} - I_{sh} - I_{d1} - I_{d2} \\ &= I_{ph} - \frac{I_L \cdot R_s + V_L}{R_{sh}} - I_{sd1}[\exp(\frac{(I_L \cdot R_s + V_L) \cdot q}{T \cdot n_1 \cdot k}) - 1] \end{aligned} \tag{11}$$

In (11), I_{sd1} corresponds to the diffusion current and I_{sd2} represents the saturation current. The ideality factors of the diodes are denoted by n_1 and n_2. Notably, DDM aims to extract seven key parameters: $(I_{ph}, I_{sd1}, R_s, R_{sh}, n_1, I_{sd2}, n_2)$.

Unlike SDM and DDM, the PVMM configuration uses multiple identical PV cells arranged in parallel or series (see Figure 3(c)), which incorporates the expression for the photocurrent I_L:

$$\begin{aligned} I_L = I_{ph} N_p - I_{sd} N_p \exp[(\frac{(I_L N_s R_s / N_p + V_L) \cdot q}{T \cdot n \cdot k \cdot N_s}) - 1] \\ - \frac{I_L R_s N_s / N_p + V_L}{N_s R_{sh} / N_p}, \end{aligned} \tag{12}$$

where N_p and N_s denote the number of parallel and serial connected cells, respectively. This model aims to extract five important parameters: $(I_{ph}, I_{sd}, R_s, R_{sh}, n)$.

TABLE III presents the detailed parameter ranges for the three PV models. The objective is to minimize the disparity between experimental and measured data. Consequently, the root mean square error (RMSE) is frequently employed as the fitness function, with its mathematical formulation provided in (13):

$$\mathrm{RMSE}(X) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} f_i(X, I_L, V_L)}. \tag{13}$$

TABLE III
BOUNDS OF DIFFERENT PARAMETERS IN THREE PV MODELS

Parameter	SDM		DDM		PVMM	
	LB	UB	LB	UB	LB	UB
$I_{ph}(A)$	0	1	0	2	0	2

I_{sd1}, I_{sd2}	0	1	0	1	0	50
n_1, n_2, n	1	2	1	2	1	50
$R_{sh}(\Omega)$	0	100	0	100	0	2000
$R_s(\Omega)$	0	0.5	0	0.5	0	2

Let N be the number of measured data, and X be the solution vector incorporating the unknown core parameters. The objective function of each model is then stated as follows:

Model SDM:

$$f_i(X,I_L,V_L)=I_{ph}-I_{sd}[\exp(\frac{(I_L\cdot R_s+V_L)\cdot q}{T\cdot n\cdot k})-1]$$
$$-\frac{I_L\cdot R_s+V_L}{R_{sh}}-I_L$$
$$X=\{I_{ph},I_{sd},R_s,R_{sh},n\}$$

Model DDM:

$$f_i(X,I_L,V_L)=I_{ph}-I_{sd1}[\exp(\frac{(I_L\cdot R_s+V_L)\cdot q}{T\cdot n_1\cdot k})-1]$$
$$-I_{sd2}\cdot[\exp(\frac{(I_L\cdot R_s+V_L)\cdot q}{T\cdot n_2\cdot k})-1]-\frac{I_L\cdot R_s+V_L}{R_{sh}}-I_L$$
$$X=\{I_{ph},I_{sd1},R_s,R_{sh},n_1,I_{sd2},n_2\}$$

Model PVMM:

$$f_i(X,I_L,V_L)=N_pI_{ph}-N_pI_{sd}\exp[(\frac{(I_LN_sR_s/N_p+V_L)\cdot q}{T\cdot n\cdot k\cdot N_s})$$
$$-1]-\frac{I_LR_sN_s/N_p+V_L}{N_sR_{sh}/N_p}-I_L$$
$$X=\{I_{ph},I_{sd},R_s,R_{sh},n\}$$

In this experiment, a commercial RTC France silicon solar cell with a diameter of 57 mm (exposed to an irradiance of less than 1000 W/m^2 at a temperature of 33 °C) was utilized. This particular benchmark data set has been extensively adopted for assessing the efficacy of algorithms designed for parameter extraction purposes [21].

As presented in Tables IV to IX, the optimal outcomes achieved by six distinct algorithms are shown. These results are derived from 30 independent executions of the algorithms on SDM, DDM, and PVMM.

TABLE IV
THE BEST OUTCOMES ATTAINED BY SIX METHODS OVER 20 INDEPENDENT RUNS ON SDM

Variables	Methods		
	DE	JaDE	SHADE
$I_{ph}(A)$	7.6E-01	7.6E-01	7.6E-01
$I_{sd}(\mu A)$	3.2E-07	3.8E-07	3.2E-07
$R_s(\Omega)$	3.6E-02	3.5-02	3.6E-02
$R_{sh}(\Omega)$	5.3E+01	5.8E+01	5.3E+01
n	1.4E+00	1.4E+00	1.4E+00
RMSE	1.0E-3	9.8E-04	9.8E-04

TABLE V
THE BEST OUTCOMES ATTAINED BY SIX METHODS OVER 20 INDEPENDENT RUNS ON SDM

Variables	Methods		
	WDE	BESD	EPDE
$I_{ph}(A)$	7.6E-01	7.6E-01	7.6E-01
$I_{sd}(\mu A)$	3.2E-07	3.3E-07	3.3E-07
$R_s(\Omega)$	3.6E-02	3.6E-02	3.6E-02
$R_{sh}(\Omega)$	5.3E+01	5.3E+01	5.4E+01
n	1.4E+00	1.4E+00	1.4E+00
RMSE	9.9E-04	9.9E-04	9.8E-04

From TABLE IV and TABLE V, we can see that, for SDM, the best outcomes attained by DE, JaDE, SHADE, WDE, BESD, and EPDE are 1.0E-03, 9.8E-04, 9.8E-04, 9.9E-04, 9.9E-04, and 9.8E-04, respectively. As a result, EPDE outperforms DE, WDE, and BESD.

TABLE VI
THE BEST OUTCOMES ATTAINED BY SIX METHODS OVER 20 INDEPENDENT RUNS ON DDM

Variables	Methods		
	DE	JaDE	SHADE
$I_{ph}(A)$	7.6E-01	7.6E-01	7.6E-01
$I_{sd1}(\mu A)$	3.4E-02	3.6E-02	3.6E-02
$R_s(\Omega)$	8.4E+01	5.3E+01	6.0E+01
$R_{sh}(\Omega)$	2.5E-07	2.7E-07	2.5E-07
n_1	1.5E+00	1.4E+00	1.8E+00
$I_{sd2}(\mu A)$	2.3E-07	1.8E-07	3.7E-07
n_2	1.5E+00	1.8E+00	1.7E+00
RMSE	1.4E-03	9.8E-04	1.0E-03

TABLE VII
THE BEST OUTCOMES ATTAINED BY SIX METHODS OVER 20 INDEPENDENT RUNS ON DDM

Variables	Methods		
	WDE	BESD	EPDE
$I_{ph}(A)$	7.6E-01	7.6E-01	7.6E-01
$I_{sd1}(\mu A)$	3.5E-02	3.6E-02	3.6E-02
$R_s(\Omega)$	6.5E+01	5.6E+01	5.5E+01
$R_{sh}(\Omega)$	2.3E-07	1.9E-07	2.3E-07
n_1	1.8E+00	1.7E+00	1.4E+00
$I_{sd2}(\mu A)$	3.5E-07	2.3E-07	6.5E-07
n_2	1.4E+00	1.4E+00	1.9E+00
RMSE	1.1E-03	1.1E-03	9.8E-04

From TABLE VI and TABLE VII, we can see that, for DDM, the best outcomes attained by DE, JaDE, SHADE, WDE, BESD and EPDE are 1.4E-03, 9.8E-03, 1.0E-03, 1.1E-03, 1.1E-03, and 9.8E-04, respectively. As a result, EPDE outperforms DE, SHADE, WDE, and BESD.

TABLE VIII
THE BEST OUTCOMES ATTAINED BY SIX METHODS OVER 20 INDEPENDENT RUNS ON PVMM

Variables	Methods		
	DE	JaDE	SHADE
$I_{ph}(A)$	2.1E-01	2.1E-01	2.1E-01
$I_{sd}(\mu A)$	2.3E-06	7.0E-07	7.0E-07
$R_s(\Omega)$	1.7E+00	2.0E+00	2.0E+00
$R_{sh}(\Omega)$	2.0E+03	1.6E+03	1.6E+03
n	1.8E+01	1.6E+01	1.6E+01
RMSE	5.3E-03	2.4E-03	2.4E-03

TABLE IX
THE BEST OUTCOMES ATTAINED BY SIX METHODS OVER 20 INDEPENDENT RUNS ON PVMM

Variables	Methods		
	WDE	BESD	EPDE
$I_{ph}(A)$	2.1E-01	7.6E-01	7.6E-01
$I_{sd}(\mu A)$	1.9E-06	2.4E-06	7.0E-07
$R_s(\Omega)$	1.8E+00	1.7E+00	2.0E+00
$R_{sh}(\Omega)$	1.6E+03	1.6E+03	1.7E+03
n_1	1.6E+01	1.7E+01	1.6E+01
RMSE	5.1E-03	6.0E-03	2.4E-03

From TABLE VIII and TABLE IX, we can see that, for PVMM, the best outcomes attained by DE, JaDE, SHADE, WDE, BESD and EPDE are 5.3E-03, 2.4E-03, 2.4E-03, 5.1E-03, 6.0E-03, and 2.4E-03, respectively. As a result, EPDE outperforms DE, WDE, and BESD.

VI. Conclusion

In this paper, we propose an improved DE algorithm featuring an entropy guided parameter adaptation mechanism and a double-strategy selection mutation scheme. Based on population entropy, the proposed mechanism preserves population diversity and accelerates convergence by adjusting F and CR. The dual-strategy mutation operator introduces orthogonal exploration directions, dynamically balancing global search diversification and local search intensification to mitigate premature convergence. The experimental results on benchmark functions and the practical application show the effectiveness of the proposed algorithm.

References

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

[2] W.M. Ali, and P.N. Suganthan, "Real-parameter unconstrained optimization based on enhanced fitness-adaptive differential evolution algorithm with novel mutation," *Soft Computing*, vol. 22, no. 10, pp. 3215-3235, 2018.

[3] H. Nenavath, and R.K. Jatoth, "Hybridizing sine cosine algorithm with differential evolution for global optimization and object tracking," *Applied Soft Computing,* vol. 62, pp. 1019-1043, 2018.

[4] T. Marcic, B. Stumberger, and G. Stumberger, "Differential evolution based parameter identification of a line-start IPM synchronous motor," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 11, pp. 5921-5929, 2014.

[5] S. Das, and P.N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation,* vol. 15, no. 1, pp. 4-31, 2011.

[6] Q. Qu, Y.H. Huang, and X.L. Wang, et al., "Complementary differential evolution-based whale optimization algorithm for function optimization," *IAENG International Journal of Computer Science,* vol. 47, no. 4, pp. 805-815, 2020.

[7] D. Qu, H.Y. Li, and H.F. Chen, "Multi-objective differential evolution algorithm based on affinity propagation clustering," *IAENG International Journal of Applied Mathematics,* vol. 53, no. 4, pp. 1408-1417, 2023.

[8] K.V. Price, R.M. Storn, and J.A. Lampinen,"Differential evolution: a practical approach to global optimization," Springer Science Business Media, 2005.

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

[10] Z.W. Zhao, J.M. Yang, Z.Y. Hu, and H.J. Che, "A differential evolution algorithm with self-adaptive strategy and control parameters based on symmetric Latin hypercube design for unconstrained optimization problems," *European Journal of Operational Research,* vol. 250, no. 1, pp. 30-45, 2016.

[11] Z.Y. Meng, J.S. Pan, and L.P. Kong, "Parameters with adaptive learning mechanism (PALM) for the enhancement of differential evolution," *Knowledge-Based Systems,* vol. 141, pp. 92-112, 2018.

[12] J.Q. Zhang, and A.C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation,* vol. 13, no. 5, pp. 945-958, 2009.

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

[14] S.H. Wang, Y.Z. Li, H.Y. Yang, and H. Liu, "Self-adaptive differential evolution algorithm with improved mutation strategy," *Soft Computing*, vol. 22, no. 10, pp. 3433-3447, 2018.

[15] K. Sethanan, and R. Pitakaso, " Improved differential evolution algorithms for solving generalized assignment problem," *Expert Systems with Applications*, vol. 45, pp. 450-459, 2016.

[16] Z.H. Cai, W.Y. Gong, C.X. Ling, and H. Zhang, "A clustering-based differential evolution for global optimization," *Applied Soft Computing*, vol. 11, no. 1, pp. 1363-1379, 2011.

[17] S.H. Wang, Y.Z. Li, and H.Y. Yang, "Self-adaptive mutation differential evolution algorithm based on particle swarm optimization," *Applied Soft Computing,* vol. 81, 105496, 2019.

[18] R. Tanabe, and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," In: 2013 IEEE Congress on Evolutionary Computation, 2013.

[19] P. Civicioglu, and E. Besdok, et al., "Weighted differential evolution algorithm for numerical function optimization: a comparative study with cuckoo search, artificial bee colony, adaptive differential evolution, and backtracking search optimization algorithms," *Neural Computing and Applications,* vol. 32, pp. 3923-3937, 2020.

[20] C. Pinar, and B. Erkan, "Bezier search differential evolution algorithm for numerical function optimization:a comparative study with CRMLSP, MVO, WA, SHADE and LSHADE," *Expert Systems with Applications,* vol. 165, 113875, 2021.

[21] Y. Kharchouf, R. Herbazi, and A. Chahboun, "Parameter's extraction of solar photovoltaic models using an improved differential evolution algorithm," *Energy Conversion and Management,* vol. 251, 114972, 2022.