Scientific Research Publishing

International Journal of Intelligence Science, 2021, 11, 17-30
https://www.scirp.org/journal/ijis
ISSN Online: 2163-0356
ISSN Print: 2163-0283

Covariance Matrix Learning Differential Evolution Algorithm Based on Correlation

Sainan Yuan, Quanxi Feng*

Center for Data Analysis and Algorithm Technology, Guilin University of Technology, Guilin, China
Email: *fqx9904@163.com

How to cite this paper: Yuan, S.N. and Feng, Q.X. (2021) Covariance Matrix Learning Differential Evolution Algorithm Based on Correlation. *International Journal of Intelligence Science*, **11**, 17-30.
https://doi.org/10.4236/ijis.2021.111002

Received: October 27, 2020
Accepted: December 12, 2020
Published: December 15, 2020

Abstract

Differential evolution algorithm based on the covariance matrix learning can adjust the coordinate system according to the characteristics of the population, which makes the search move in a more favorable direction. In order to obtain more accurate information about the function shape, this paper proposes covariance matrix learning differential evolution algorithm based on correlation (denoted as RCLDE) to improve the search efficiency of the algorithm. First, a hybrid mutation strategy is designed to balance the diversity and convergence of the population; secondly, the covariance learning matrix is constructed by selecting the individual with the less correlation; then, a comprehensive learning mechanism is comprehensively designed by two covariance matrix learning mechanisms based on the principle of probability. Finally, the algorithm is tested on the CEC2005, and the experimental results are compared with other effective differential evolution algorithms. The experimental results show that the algorithm proposed in this paper is an effective algorithm.

Keywords

Differential Evolution Algorithm, Correlation, Covariance Matrix, Parameter Self-Adaptive Technique

1. Introduction

Optimization problems are everywhere, and how to better solve optimization problems has become a current research hotspot. Differential evolution algorithm performs well in solving optimization problems, so it is very popular. Differential Evolution (DE) is a random search method proposed by Storn and Price *et al.* [1] in 1995. As a member of intelligent optimization methods, it has

*Corresponding author.

become one of the commonly used methods to solve optimization problems because of its high search efficiency and strong robustness. However, the standard DE algorithm has the pressure to select control parameters, the search ability is inconsistent with the development ability, and the fixed coordinate system cannot be rotated according to the landscape of function to improve the search efficiency [2]. Many documents have improved the control parameters of the DE algorithm. For example, J. Zhang *et al.* [3] proposed the parameter adaptive differential evolution algorithm (JADE) in 2009, which uses an adaptive learning scheme to adjust the values of F and CR. Wang Yong *et al.* [4] proposed an unconstrained optimization compound differential evolution algorithm (CoDE) in 2011. The algorithm uses three different mutation strategies to generate trail vectors to improve the exploration and development capabilities of the DE algorithm.

Covariance matrix learning helps to improve the optimization performance of differential evolution algorithm. Wang Yong *et al.* [5] proposed a differential evolution algorithm based on covariance matrix learning (CoBiDE) in 2014. This algorithm combines eigenvectors with crossover operators to make the algorithm rotate and not deform. Yong Lili *et al.* [6] proposed the covariance matrix and crossover matrix guided differential evolution (CCDE) in 2016. The algorithm uses the optimal individual to construct the covariance matrix and Gaussian distribution function to randomly search the area around the optimal individual, and makes full use of the best individual information to improve the search ability of the algorithm. Noor H. Awad *et al.* [7] proposed a covariance matrix learning constraint optimization algorithm based on euclidean distance (LSHADE-cnEpSin) in 2017. These algorithms mainly screen individuals to construct covariance matrix according to their fitness values, and do not consider possible collinearity among individuals. Highly related individuals will transmit duplicate information, which may cause the population to ignore important distribution information during the search process.

In order to solve the problem of information duplication between individuals and reduce algorithm efficiency, this paper proposes a differential evolution algorithm for covariance matrix learning based on correlation research. The algorithm calculates the correlation coefficient between all individuals in the population, eliminates the individuals with strong correlation, uses the remaining individuals to construct a covariance matrix, realizes coordinate rotation, and improves the search efficiency of the algorithm.

The rest of this paper is organized as follows: Section 2 introduces the preparatory work, Section 3 introduces the algorithm proposed in this paper, Section 4 analyzes the numerical experiment results, and Section 5 summarizes this paper.

2. Related Work

This section mainly introduces the preliminary preparations, mainly including the definition of optimization problems and the basic process of differential evolution algorithm, so as to facilitate the understanding of the algorithm pro-

posed in this paper.

2.1. Optimization Problem

Without loss of generality, many scientific and engineering optimization problems can be reduced to the following mathematical models. The optimization problem (minimization problem) can be defined as:

$$\min f(X) \quad X \in \Omega \tag{1}$$

where Ω represents the decision space, $X = [x_1, x_2, \cdots, x_D]$ is the decision variable, D is the size of the dimension, and the upper and lower boundaries of the j-th dimension x_j of the decision variable are L_j and U_j respectively:

$$L_j \le x_j \le U_j, \quad j = 1, 2, \cdots, D. \tag{2}$$

2.2. Differential Evolution

Differential evolution algorithm is a swarm intelligence optimization algorithm inspired by natural evolution. Its basic idea is to use a heuristic random search algorithm based on population individual differences. It mainly includes four component: initialization, mutation operator, crossover operator, and selection operator.

1) Initialization: The population size is expressed as NP, and each vector X_i represents a potential solution, that is, an individual. When g = 0, the initial population is $P_0 = \left\{X_1^0, X_2^0, \cdots, X_{NP}^0\right\}$ is generated as follows:

$$X_{i,j} = L_j + rand(0,1) * \left(U_j - L_j\right) \quad j = 1, 2, \cdots, D \tag{3}$$

where *rand*(0, 1) represents a random number that obeys [0, 1] uniform distribution.

2) Mutation: Mutation operator determines the degree of convergence and diversity of the algorithm. A mutation vector is created for each target individual by using paired difference information, where "*rand*" means that the base vector is selected randomly, and "1" means mutation. The number of random difference vectors in the equation. Let V denote the mutation vector, and the commonly used mutation strategies are as follows:

DE\rand\1:

$$V = X_{r1} + F * \left(X_{r2} - X_{r3}\right) \tag{4}$$

DE\rand-to-pbest\1:

$$V = X_{r1} + F * \left(X_{pbest} - X_{r1}\right) + F * \left(X_{r2} - X_{r3}\right). \tag{5}$$

The scaling factor $F \in [0,1]$, r_1, r_2, r_3 are randomly selected in $[1, NP]$ and are not equal to each other in the same mutation strategy. "*Best*" means the best fitness individual in the current population, and "*pbest*" means an individual randomly selected from the best p% individuals in the current population.

3) Crossover: There are two commonly used crossover methods: exponential crossover and binomial crossover. The crossover strategy used in this paper is binomial crossover. In the binomial crossover, the variation vector and the tar-

get vector generate the trail individual $U_{i,d}$ through formula (6):

$$U_{i,d} = \begin{cases} V_{i,d}, & \text{if } rand(0,1) < Cr \text{ or } j = j_{rand}; \\ X_{i,d}, & \text{otherwise.} \end{cases} \tag{6}$$

where $j_{rand} \in \{1,2,3,\cdots,D\}$ represents a random integer on [1, D], j_{rand} can ensure that at least one dimension of the D-dimensional variables of $U_{i,d}$ is contributed by V_i, and Cr is the crossover control parameter.

4) Selection: The selection operator compares the trail individual $U_{i,d}$ and the parent individual $X_{i,d}$ one to one, and selects the better individual as the $t+1$ generation individual X_i^{t+1}.

3. The Proposed Approach

This section mainly introduces the algorithm designed in this paper, the differential evolution algorithm based on the covariance matrix learning of correlation, denoted as RCLDE. This section mainly introduces: the correlation-based covariance matrix learning mechanism, the double mutation strategy coordination mechanism, the adaptive parameter control strategy and the general framework of the algorithm RCLDE. The details will be described below.

3.1. Covariance Matrix Learning Mechanism Based on Correlation

In the covariance matrix learning mechanism, the differences in the construction of the covariance matrix directly affect the performance of the mechanism. The correlation coefficient between variables can reflect the correlation between two indicators. The larger the correlation coefficient, the higher the correlation of the information reflected by the two individuals, and part of the information carried by the two individuals will overlap. The majority of scholars construct the covariance learning matrix mainly based on the optimality of individual fitness values, which is beneficial to speed up the optimization performance of the algorithm, but this method does not consider the correlation between individuals, which may affect the learning efficiency of the learning mechanism. Therefore, in order to improve the performance of the covariance matrix learning mechanism, this paper considers the correlation between individuals when constructing the covariance matrix, removes some individuals with strong correlation, and then constructs the covariance matrix, performs Eigen decomposition, and obtains the eigenvector The original coordinate system is rotated.

The covariance matrix learning mechanism includes two steps: Eigen decomposition and coordinate transformation of the covariance matrix. First, perform eigendecomposition on the covariance matrix to obtain the eigenvectors, and use the Eigen vectors as the Eigen coordinate system, which is the axial direction of the new coordinate system. Then, according to the distribution information of the population, the original coordinate system is rotated through the Eigen coordinate system to adjust the search direction of the population and improve the search efficiency of the global optimal solution. The specific steps of the cor-

relation-based covariance matrix learning mechanism used in this paper are as follows:

Step 1 Find the correlation coefficients of all individuals in the current population, remove one of the two individuals with a correlation coefficient greater than α, use the remaining individuals with weak correlations to construct the covariance matrix C, and perform Eigen-decomposition on C:

$$C = BD^2B^T \tag{7}$$

where is the orthogonal matrix B, each column of B is the eigenvector of the covariance matrix C, and D is a diagonal matrix composed of eigenvalues.

Step 2 Use the orthogonal matrix B^T to map the target vector and mutation vector to the Eigen coordinate system:

$$X_i' = B^{-1}X_i = B^T X_i \tag{8}$$

$$V_i' = B^{-1}V_i = B^T V_i. \tag{9}$$

Step 3 Cross operation of target vector X_i' and mutation vector V_i' in the Eigen coordinate system to generate trail vector U_i':

$$U_{i,j}' = \begin{cases} V_{i,j}', & \text{if } rand_j(0,1) \le CR \text{ or } j = j_{rand} \\ X_{i,j}', & \text{otherwise.} \end{cases} \tag{10}$$

Step 4 Use the orthogonal matrix B to convert the trail vector U_i' back to the original coordinate system:

$$U_i = BU_i' \tag{11}$$

here U_i is the trail vector in the original coordinate system.

3.2. Mutation Strategy

In order to reduce the influence of mutation strategy on differential evolution algorithm, this paper draws on the probability calculation method of [8] and proposes a double mutation strategy cooperative coevolution mechanism based on probability selection mechanism. **Figure 1** shows the pseudocode of the mutation strategy.

In the early stage of evolution, in order to maintain the diversity and search ability of the population, the probability of executing the mutation strategy DE/rand/1 is greater. In the later stage of evolution, in order to improve the convergence speed of the algorithm and adjust the diversity of the population, the probability of executing the mutation strategy DE/rand-to-pbest/1is greater, and the information of the optimal individual is fully utilized. The mutation strategy can not only maintain the diversity of the population, but also adjust the diversity of the population according to the iterative process, and improve the convergence speed of the algorithm.

3.3. Adaptive Parameter Control Strategy

The setting of the parameter size directly affects the solution effect of the algorithm. In this paper, the scaling factor F and crossover control parameter CR

1) **Input**: *FES*: Current evaluation times
F: The scaling factor
X_i : Target individual in the current population
$X_{r1} \backslash X_{r2} \backslash X_{r3}$: Individuals randomly selected from the current popularion (r_1, r_2, r_3 is not equal to each other and not equal to i)
X_{pbest} : Individuals randomly selected from outstanding individuals in the current population
2) **For** $i = 1$: NP
3) **If** $rand < \frac{1}{1+e^{1-(FES_{MAX}/FES)^2}}$
4) $V = X_{r1} + F*(X_{r2} - X_{r3})$
5) **Else**
6) $V = X_{r1} + F*(X_{pbest} - X_{r1}) + F*(X_{r2} - X_{r3})$
7) **End If**
8) **End For**

Figure 1. Pseudocode of double mutation collaborative strategy.

refer to Wang *et al.* [4] proposed the bimodal distribution parameter setting in CoBiDE, which adaptively generates the corresponding value for each target vector X_i^t. Scaling factor F and crossover control parameter CR. If the trail vector U_i^t generated by the mutation and crossover of X_i^t successfully enters the next generation, then $F_i^t = F_i^{t+1}$, $CR_i^t = CR_i^{t+1}$, otherwise F and CR will be regenerated for the next generation X_i^{t+1} as follows:

$$F_i = \begin{cases} randc_i(0.65, 0.1), & \text{if } rand(0,1) < 0.5 \\ randc_i(1.0, 0.1), & \text{otherwise} \end{cases} \tag{12}$$

$$CR_i = \begin{cases} randc_i(0.1, 0.1), & \text{if } rand(0,1) < 0.5 \\ randc_i(0.95, 0.1), & \text{otherwise.} \end{cases} \tag{13}$$

Among them, $rand(0,1)$ is a uniformly distributed random number between 0 and 1, and $randc_i(a,b)$ is a Cauchy distributed random number with a location parameter of a and a scale parameter of b.

3.4. Algorithm Main Framework

This section will specifically introduce the algorithm RCLDE proposed in this paper. The algorithm is mainly composed of the above mentioned double mutation cooperative strategy, adaptive parameter control, and covariance matrix learning mechanism based on correlation. First, based on the probability formula, the two mutation strategies DE\rand\1 and DE\rand-to-pbest\1 with different functions are combined to generate mutant individuals. DE\rand\1 has strong randomness, which can improve population diversity and search ability. In the early stage of evolution, extensive searches are required, so the probability of setting DE\rand\1 is greater. DE\rand-to-pbest\1 contains the information of excellent individuals in the current population, which provides a favorable direc-

tion for search and speeds up the convergence speed. Therefore, the probability of setting DE\rand-to-pbest\1 in the later stage of evolution is greater. Secondly, in order to eliminate the collinearity between individuals and cause the problem of repetitive information transmission, according to the correlation coefficient between individuals in the population, the individuals with strong correlation are eliminated, and the remaining individuals are used to construct the covariance matrix for coordinate rotation, which improves the search efficiency of algorithm. Finally, the crossover process is based on probability selection under the covariance matrix learning based on correlation. The algorithm pseudocode is shown in **Figure 2**.

Lines 1 - 4 in the RCLDE pseudocode are the process of initializing the population and parameters, and lines 5 - 42 are the process of the algorithm looping to search for the optimal solution. Lines 6 - 10 are mutation operations on each target vector. Lines 11 - 29 are crossover operations. First, calculate the correlation coefficient matrix of all individuals in the current population. If it is less than the random number generated between 0.5 and 1 and the elements in the correlation coefficient matrix are not all 1, then all two individuals are correlated One individual with a coefficient greater than α is eliminated, and the remaining linearly independent individuals are used to construct a covariance matrix to perform coordinate rotation. If the random number generated between 0 and 1 is less than 0.5, all individuals are sorted according to the function value from small to large, and the top 50% of the individuals in the current population are taken to construct a covariance matrix, and then coordinate rotation and crossover operations are performed, otherwise in the original coordinates perform crossover operations under the department to generate trail vectors. 30 - 39 compares the target vector and the trail vector according to the function value, and selects the smaller function value between the two vector to enter the next generation, and the successful F and CR will also enter the next generation. Lines 40 - 42 represent repeated cycles until the optimal solution is found.

4. Experimental Study

In order to verify the effectiveness of the algorithm RCLDE, this paper will test RCLDE on the CEC2005 standard test function [9], and compare it with other differential evolution algorithms jDE [10], SaDE [11], COBiDE [5], etc. Select the function g01 - g20 in CEC2005 as the test function. Among them, g01 - g05 are unimodal functions, g06 - g12 are basic multimodal functions, g13 - g14 are expanded multimodal functions, and g15 - g20 are hybrid composition functions.

In this experiment, the population size NP of the RCLDE algorithm is set to 60, and the dimension D is 30. The algorithm runs 25 times independently, and the mean (Mean) and variance (STD) of the results of the 25 runs are used as indicators for evaluating performance. According to the numerical results obtained from the experiment, the Mann-Whitney rank sum test [12] and Frideman test [13] are used for statistical analysis.

1) **Input:** *NP*: Number of individuals in the population
FES: current function evaluation times
2) $G = 0$;
3) Generate initial population $P_0 = \{X_1^0, X_2^0, \cdots, X_{NP}^0\}$;
4 Evaluate the function value of each individual in the population, $FES = NP$;
5) According to formulas (12) and (13), calculate the initial scaling factor F and CR of each individual respectively;
6) **While** $FES < MaxFEs$
7) $P = \varnothing$;
8) For $i = 1 : NP$
9) Apply mutation strategy to generate V_i;
10) **End For**
11) **Handling boundary constraints;**
12) Find the correlation coefficient matrix of all individuals in the current population R;
13) **If** $rand(0,1) < 0.5$
14) **For** $i = 1 : NP$
15) **If** $R = 1$
16) Cross operator in the original coordinate system to generate trail vectors U_i;
17) **Else**
18) Covariance matrix learning based on correlation to generate trail vector U_i;
19) **End If**
20) **End For**
21) **else**
22) **If** $rand(0,1) < 0.5$
23) **For** $i = 1 : NP$
24) Covariance matrix learning based on the size of the function value to generate test vectors trail vector U_i;
25) **End For**
26) **Else**
27) Cross operation in the original coordinate system to generate trail vectors U_i;
28) **End If**
29) **End If**
30) **Handling boundary constraints;**
31) **For** $i = 1 : NP$
32) Evaluation of the function value of U_i;
33) **If** $f(X_i) < f(U_i)$
34) $P = P \cup X_i$;
35) $F = F \cup F_i$ and $CR = CR \cup CR_i$;
36) **Else**
37) $P = P \cup U_i$;
38) Generate F_{i+1} and CR_{i+1} for the next generation according to equations (13) and (14);
39) **End If**
40) **End For**
41) $FES = FES + NP$;
42) $G = G + 1$;
43) **End While**

Figure 2. Pseudocode of RCLDE.

4.1. Parameter Setting Experiment Analysis

In order to verify the validity of the parameter setting in the learning of the correlation covariance matrix, an experiment in this section is set up. Generally, the correlation coefficient is greater than or equal to 0.6, indicating that the two variables are correlated. In order to test what the correlation threshold is set to, the algorithm works best. In this section, four correlation coefficient thresholds of 0.6, 0.7, 0.8, and 0.9 are selected for experiments. The experiment in this section is run 25 times independently on the 30D CEC2005, and the population size is set to 60. The numerical results obtained by the algorithm under different thresholds are shown in Table 1. Table 2 is the Frideman test numerical results of the average of the optimal values obtained by running 25 times. The black bold font is the rank sum of the optimal parameters.

Table 1. Comparison of experimental results of correlation coefficient threshold.

Relevance threshold	0.6		0.7		0.8		0.9	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
g01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
g02	2.11E−22	5.01E−22	7.61E−23	1.40E−22	9.25E−23	2.10E−22	3.35E−23	3.53E−23
g03	4.86E+04	2.75E+04	4.25E+04	2.70E+04	3.98E+04	3.17E+04	4.71E+04	2.63E+04
g04	3.06E−07	4.74E−07	1.90E−06	7.13E−06	2.62E−07	3.81E−07	3.46E−07	6.48E−07
g05	1.04E+02	1.59E+02	1.15E+02	1.53E+02	6.35E+01	1.30E+02	8.52E+01	1.54E+02
g06	2.01E−19	6.89E−19	4.29E−18	2.13E−17	6.38E−01	1.49E+00	4.78E−01	1.32E+00
g07	9.06E−03	9.37E−03	1.30E−02	1.19E−02	6.40E−03	7.71E−03	5.62E−03	7.34E−03
g08	2.00E+01	7.09E−03	2.00E+01	1.19E−02	2.00E+01	1.26E−05	2.00E+01	3.28E−05
g09	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.58E−01	6.96E−01
g10	4.45E+01	1.38E+01	4.19E+01	1.30E+01	4.58E+01	1.23E+01	4.68E+01	1.23E+01
g11	1.03E+01	3.29E+00	1.04E+01	3.20E+00	9.33E+00	2.11E+00	8.70E+00	3.02E+00
g12	2.58E+03	3.06E+03	4.23E+03	5.10E+03	5.06E+03	9.40E+03	5.52E+03	7.74E+03
g13	1.95E+00	3.55E−01	2.09E+00	4.34E−01	2.63E+00	7.62E−01	2.39E+00	6.80E−01
g14	1.25E+01	5.27E−01	1.27E+01	3.01E−01	1.24E+01	3.66E−01	1.26E+01	3.72E−01
g15	4.28E+02	4.58E+01	4.24E+02	4.36E+01	4.20E+02	4.08E+01	4.12E+02	6.00E+01
g16	6.91E+01	1.42E+01	6.69E+01	1.30E+01	7.22E+01	1.95E+01	8.59E+01	6.89E+01
g17	7.85E+01	3.42E+01	7.14E+01	2.07E+01	6.63E+01	1.58E+01	8.70E+01	7.21E+01
g18	9.04E+02	9.46E−01	9.04E+02	4.08E−01	9.04E+02	1.07E+00	9.00E+02	2.09E+01
g19	9.04E+02	8.90E−01	9.04E+02	9.04E−01	9.04E+02	5.90E−01	9.04E+02	1.05E+00
g20	9.04E+02	7.37E−01	9.04E+02	6.43E−01	9.04E+02	7.89E−01	9.00E+02	2.09E+01

Table 2. The frideman test result of the mean of the experimental results of the correlation threshold.

Relevance threshold	0.6	0.7	0.8	0.9
Rank sum	2.63	2.58	**2.28**	2.53

It can be seen from Table 1 that when the correlation coefficient threshold is 0.8, the average of the optimal value performs best overall, and the variance is relatively stable relative to other parameters. Specifically, when the correlation coefficient threshold is set to 0.8, the numerical results of the optimal values of functions g01, g04, g08, g09, g11, g14, g15, g17, and g19 are better than those of other groups. It can be seen more intuitively from Table 2 that in the Frideman test results of the optimal mean value, the rank sum is the smallest when the correlation coefficient threshold is 0.8, indicating that when the algorithm RCLDE correlation threshold parameter is set to 0.8, and the algorithm effect is the best and the correlation is when the threshold parameter is set to 0.9, the algorithm effect is second. The effect is not good when the correlation threshold is set to 0.6. It means that the correlation threshold is too large or too small, and 0.8 is the most suitable. The effect is not good when the correlation threshold is set to 0.6. The relevance threshold is set too small, and there are many related individuals, which causes overlap of information and affects the search efficiency of the population. If the correlation threshold is set too large, more individuals may be screened out, reducing the search information of the population, and it is easy to cause the population to fall into a local optimal solution.

4.2. Comparative Analysis with Other Differential Evolution Algorithms

In order to prove the effectiveness of this algorithm RCLDE, the algorithm was tested 25 times on the 20 standard tests of 30D CEC2005, and compared with algorithms such as jDE, SaDE, COBiDE, etc. The experimental results of jDE, SaDE and COBiDE are directly taken from the literature [5].

The experimental numerical results of the RCLDE algorithm and the three evolutionary algorithms are shown in Table 3. In the data in Table 3: 1) Numerical results slanted bold font indicates the solution result of the RCLDE algorithm. 2) Regular font indicates that the numerical result of the RCLDE solution is worse than the comparison algorithm. 3) Bold font indicates the value of RCLDE The result is better than or similar to other comparison algorithms.

The Mann-Whitney rank sum test is used in Table 4 to compare the degree of difference of different algorithms in the same problem. The significance level of the Mann-Whitney rank sum test in the algorithm is set to 0.05. The comparison results are shown in Table 4, where "+", "≈", and "−" respectively indicate that the numerical results of PIMDE are better than the number of functions of the comparison algorithm, approximation The number of functions that are similar to the comparison algorithm result is worse than the number of functions of the comparison algorithm.

Table 3. Comparison table of experimental results between RCLDE and other evolutionary algorithm test functions.

IEEE CEC 2005	jDE		SaDE		CoBiDE		RCLDE	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
g01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
g02	**1.11E−06**	**1.96E−06**	**8.26E−06**	**1.65E−05**	**1.60E−12**	**2.90E−12**	**9.25E−23**	**2.10E−22**
g03	**1.98E+05**	**1.10E+05**	**4.27E+05**	**2.08E+05**	**7.26E+04**	**5.64E+04**	**3.98E+04**	**3.17E+04**
g04	**4.40E−02**	**1.26E−01**	**1.77E+02**	**2.67E+02**	**1.16E−03**	**2.74E−03**	**2.62E−07**	**3.81E−07**
g05	**5.11E+02**	**4.40E+02**	**3.25E+03**	**5.90E+02**	**8.03E+01**	**1.51E+02**	**6.35E+01**	**1.30E+02**
g06	**2.35E+01**	**2.50E+01**	**5.31E+01**	**3.25E+01**	**4.13E−02**	**9.21E−02**	**6.38E−01**	**1.49E+00**
g07	**1.18E−02**	**7.78E−03**	**1.57E−02**	**1.38E−02**	1.77E−03	3.73E−03	**6.40E−03**	**7.71E−03**
g08	**2.09E+01**	**4.86E−02**	**2.09E+01**	**4.95E−02**	**2.07E+01**	**3.75E−01**	**2.00E+01**	**1.26E−05**
g09	**0.00E+00**	**0.00E+00**	**2.39E−01**	**4.33E−01**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
g10	**5.54E+01**	**8.46E+00**	**4.72E+01**	**1.01E+01**	4.41E+01	1.29E+01	**4.58E+01**	**1.23E+01**
g11	**2.79E+01**	**1.61E+00**	**1.65E+01**	**2.42E+00**	5.62E+00	2.19E+00	**9.33E+00**	**2.11E+00**
g12	**8.63E+03**	**8.31E+03**	3.02E+03	2.33E+03	2.94E+03	3.93E+03	**5.06E+03**	**9.40E+03**
g13	**3.94E+00**	**1.35E−01**	**3.94E+00**	**2.81E−01**	**2.64E+00**	**1.13E+00**	**2.63E+00**	**7.62E−01**
g14	**1.26E+01**	**2.00E−01**	**1.26E+01**	**2.83E−01**	1.23E+01	4.90E−01	**1.24E+01**	**3.66E−01**
g15	3.77E+02	8.02E+01	3.76E+02	7.83E+01	**4.04E+02**	**5.03E+01**	**4.20E+02**	**4.08E+01**
g16	**7.94E+01**	**2.96E+01**	8.57E+01	6.94E+01	**7.38E+01**	**3.66E+01**	**7.22E+01**	**1.95E+01**
g17	**1.37E+02**	**3.80E+01**	**7.83E+01**	**3.76E+01**	**7.25E+01**	**2.02E+01**	**6.63E+01**	**1.58E+01**
g18	**9.04E+02**	**1.08E+01**	8.68E+02	6.23E+01	9.03E+02	1.05E+01	**9.04E+02**	**1.07E+00**
g19	**9.04E+02**	**1.11E+00**	8.74E+02	6.22E+01	9.03E+02	1.04E+01	**9.04E+02**	**5.90E−01**
g20	**9.04E+02**	**1.10E+00**	8.78E+02	6.03E+01	9.04E+02	5.95E−01	**9.04E+02**	**7.89E−01**

Table 4. Mann-whitney rank sum test results of each differential evolution algorithm under 30D.

RCLDE vs	jDE	SaDE	CoBiDE
+	14	13	9
≈	5	1	3
−	1	6	8

Judging from the numerical results in Table 3, the numerical results of the algorithm RCLDE perform well in the unimodal function and the basic multimodal function. Specifically, the algorithm RCLDE solves the function g01 - g06, g08, g09, g13, g17 to obtain the optimal value of the average value is better than or similar to the numerical results of the algorithm jDE, SaDE, COBiDE; The numerical result of the algorithm RCLDE to solve the optimal value of the function g07, g10, g11 is better than or similar to jDE, SaDE, and slightly worse than COBiDE.

It can be seen from Table 4 that the algorithm RCLDE is generally better or not worse than the other three differential evolution algorithms jDE, SaDE, COBiDE. The numerical results of the algorithm RCLDE are better than, similar to, and the number of functions worse than jDE is 14, 5, and 1, respectively; the numerical results of the algorithm RCLDE are better than, similar to, and the number of functions worse than SaDE is 13, 1, respectively. 6; The numerical results of algorithm RCLDE are better than, similar to, and worse than SaDE, the number of functions is 9, 3, and 8, respectively.

5. Conclusions

This paper first selects two mutation strategies with different performances to form a double mutation strategy cooperative coevolution mechanism. The probability of the two mutation strategies being used changes correspondingly with the evolution of the population, which improves the search efficiency of the population; secondly, this paper proposes based on correlation. The learning of the covariance matrix takes into account the correlation between individuals in the population, eliminates the individuals with collinearity based on the strength of the correlation, and uses the individuals with weak correlation to construct the covariance matrix, which avoids information duplication and other problems, and improve the search efficiency of the algorithm. Finally, based on the principle of probability, the covariance matrix learning mechanism of constructing the covariance matrix according to the fitness value is combined with the correlation-based covariance matrix learning mechanism proposed in this paper to generate test individuals.

The analysis of the experimental results shows that the algorithm performs best when the correlation coefficient threshold in the algorithm RCLDE proposed in this paper is set to 0.8. The algorithm RCLDE is compared with other advanced differential evolution algorithms on the CEC2005 standard test function. The experimental results show that the algorithm proposed in this paper performs well and has certain competitiveness.

Acknowledgements

The authors would like to thank anonymous reviewers for their valuable comments and suggestions for this paper. This work is proudly supported in part by National Natural Science Foundation of China (No. 61763008, 11661030, 11401357,

11861027), Natural Science Foundation of Guangxi (No. 2018GXNSFAA138095), Fangchenggang Scientific research and technology development plan 2014 No. 42, and Project supported by Program to Sponsor Teams for Innovation in the Construction of Talent Highlands in Guangxi Institutions of Higher Learning ([2011]47) and and Doctoral Research Fund of Guilin University of Technology.

Conflicts of Interest

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

References

[1] Storn, R. and Price, K. (1997) Differential Evolution: A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, **11**, 341-359. https://doi.org/10.1023/A:1008202821328

[2] Ding, Q.F. and Xin, X.Y. (2017) Research Survey of Differential Evolution Algorithms. *CAAI Transactions on Intelligent Systems*, **4**, 431-442.

[3] Zhang, J. (2009) Jade: Adaptive Differential Evolution with Optional External Archive. *IEEE Transactions on Evolutionary Computation*, **13**, 945-958. https://doi.org/10.1109/TEVC.2009.2014613

[4] Wang, Y., Cai, Z. and Zhang, Q. (2011) Differential Evolution with Composite Trial Vector Generation Strategies and Control Parameters. *IEEE Transactions on Evolutionary Computation*, **15**, 55-66. https://doi.org/10.1109/TEVC.2010.2087271

[5] Wang, Y., Li, H.X., Huang, T. and Li, L. (2014) Differential Evolution Based on Covariance Matrix Learning and Bimodal Distribution Parameter Setting. *Applied Soft Computing Journal*, **18**, 232-247. https://doi.org/10.1016/j.asoc.2014.01.038

[6] Li, Y.L., Feng, J.F. and Hu, J.H. (2016) Covariance and Crossover Matrix Guided Differential Evolution for Global Numerical Optimization. *Springerplus*, **5**, 1176. https://doi.org/10.1186/s40064-016-2838-5

[7] Awad, N.H., Ali, M.Z. and Suganthan, P.N. (2017) Ensemble Sinusoidal Differential Covariance Matrix Adaptation with Euclidean Neighborhood for Solving CEC2017 Benchmark Problems. 2017 *IEEE Congress on Evolutionary Computation* (*CEC*), San Sebastian, 5-8 June 2017, 372-379. https://doi.org/10.1109/CEC.2017.7969336

[8] Li, Y., Wang, S. and Yang, B. (2020) An Improved Differential Evolution Algorithm with Dual Mutation Strategies Collaboration. *Expert Systems with Applications*, **153**, 113-451. https://doi.org/10.1016/j.eswa.2020.113451

[9] Suganthan, P.N., Hansen, N., Liang, J.J., *et al.* (2005) Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. *Natural Computing*, 341-357.

[10] Brest, J., Greiner, S., Boskovic, B., Mernik, M. and Zumer, V. (2006) Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Transactions on Evolutionary Computation*, **10**, 646-657. https://doi.org/10.1109/TEVC.2006.872133

[11] Brest, J., Zumer, V. and Maucec, M.S. (2006) Self-Adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization. *IEEE Congress on Evolutionary Computation* (*CEC* 2006), Vancouver, BC, 16-21 July 2006, 215-222. https://doi.org/10.1109/CEC.2006.1688311

[12] Mann, H.B. and Whitney, D.R. (19476) On a Test of Whether One of Two Random

Variables Is Stochastically Larger Than the Other. *Annals of Mathematical Statistics*, **18**, 50-60. https://doi.org/10.1214/aoms/1177730491

[13] Schach, S., *et al.* (1979) An Alternative to the Friedman Test with Certain Optimality Properties. *The Annals of Statistics*, **7**, 537-550. https://doi.org/10.1214/aos/1176344675