Improving Clustering-based Differential Evolution with Chaotic Sequences and New Mutation Operator

Gang Liu, Yuanxiang Li, Xin Nie, Yu Sun

State Key Lab of Software Engineering, Computer School, Wuhan University, Wuhan, 430072, P.R. China, E-mail: lg0061408@126.com (Gang liu), yxli@whu.edu.cn (Yuanxiang Li), nix83@163.com (Xin Nie), sunyu1225@163.com (Yu Sun)
doi: 10.4156/ijact.vol3.issue6.32

Abstract

Differential evolution (DE) algorithms compose an efficient type of evolutionary algorithm (EA) for the global optimization domain. But DE is not completely free from the problems of slow and/or premature convergence. In this paper, an improving clustering-based differential evolution with chaotic sequences and new mutation operator (CCDE) is proposed for the unconstrained global optimization problems. In CCDE, the population is partitioned into k subsets by a clustering algorithm, and each subset is considered to be the cluster neighborhood. And these cluster centers and chaotic sequences using logistic equation are used to design the new differential evolution mutation operators. This method utilizes the concept of the cluster neighborhood of each population member. The CEC2005 benchmark functions are employed for experimental verification. Experimental results indicate that CCDE is highly competitive compared to the state-of-the-art DE algorithms.

Keywords: Differential Evolution, K-means Clustering, Chaotic Sequences, Self-adaptive, Global Optimization

1. Introduction

Differential evolution (DE), proposed by Storn and Price [1-3], is a very popular evolutionary algorithm (EA) and is a simple yet powerful algorithm for real parameter optimization. DE is a population-based stochastic search technique. It uses mutation, crossover, and selection operators at each generation to move its population toward the global optimum. It has successfully been applied to diverse domains of science and engineering, such as the design of digital filters [4], reduced-order/fixed-structure controllers [5], satellite image registration [6], and many others. DE has been shown to perform better than the genetic algorithm (GA) [7] or the particle swarm optimization (PSO) [8] over several numerical benchmarks [9].

The advantages of DE are simple structure, ease of use, speed and robustness. However, DE has been shown to have certain weaknesses. It may be trapped in a local optimum point and also be premature convergence in optimization. In addition, DE is good at exploring the search spaces and locating the region of global minimum, but it is slow at exploitation of the solution [10]. In recent years, a lot of researches have been proposed to improve the performance of DE. Recent studies of the improvement of different mutation operators show that these methods are effective and individuals in the population exhibit the tendency to gather around optimizers of the objective function. These methods are listed in the literature [11-13]. Other studies combine DE with chaotic sequences for improved DE approaches, such as the literature [14, 15]. Some other studies focus on the adaptive DE, such as the literature [16, 17].

Motivated by these findings, we propose an improving clustering-based differential evolution with chaotic sequences and new mutation operator (CCDE) in this paper. CCDE is the hybridization of the one-step k-means clustering, the self-adaptive crossover rate and chaotic sequences using logistic equation with DE. To combine the exploration and exploitation capabilities of DE, we propose a new hybrid mutation scheme that utilizes a local and a global mutation operator, with an objective of balancing their effects. These mutation operators are inspired by [11] and the concept of the cluster. The similar local mutation model and the similar global mutation model are defined in this paper. They are called the cluster neighborhood mutation model and the global mutation model in this paper. In the cluster neighborhood mutation model the population is partitioned into k subsets by a clustering
algorithm, and these cluster centers are used to yield the new mutation operator. The DE scheme in [11] is also different with the DE scheme in this paper. A new parameter, called the weight factor, is used to combine the two mutation operators. CCDE employed three trial vector generation strategies including the cluster neighborhood mutation strategy, the global mutation model strategy and both combination strategy. The best vector created by three trial vector generation strategies is selected into the next generation population. Chaotic sequences using logistic equation are used in the population initialization and generate the self-adaptive parameter settings. And the weight factor is also designed by chaotic sequences using logistic equation. The CCDE approach can provide the robust optimal solutions for the nonlinear functions with continuous variables. CCDE has a very simple structure and thus is very easy to implement. The CEC2005 benchmark functions are employed for experimental verification. It compares the performance of CCDE with several state-of-the-art DE variants as well as other evolutionary algorithms. Experimental results indicate that our approach performs better, or at least comparably, in terms of the quality of the final solutions.

The remainder of this paper is organized as follows. In Section 2, we provide a brief outline of the DE algorithm and the k-means clustering used in this work is presented briefly and CCDE is presented in detail in Section 3. Experimental results are reported in Section 4. Finally, Section 5 concludes this paper.

2. Differential Evolution and K-means clustering

2.1. Differential Evolution and related work

Differential evolution is a population-based stochastic parallel direct search method that utilizes concepts borrowed from the broad class of EAs. It is a simple evolutionary algorithm that creates new candidate solutions by combining the parent individual and several other individuals of the same population. A candidate replaces the parent only if it has better fitness. This is a rather greedy selection scheme that often outperforms traditional EAs. The efficiency of differential evolution is very sensitive to the setting of control parameters. In general conditions, the control parameters depend on the results of preliminary tuning. The main procedure of DE is summarized as follows:

At generation $G = 0$, an initial population is randomly sampled from the feasible solution space. At each generation $G$, DE creates a mutant vector for each individual in the current population. Currently, there are 10 different mutation operators of DE [1-3]. Other well-known schemes are listed as follows:

$$
\tilde{V}_j = \tilde{X}_{r_1} + F^*(\tilde{X}_{r_2} - \tilde{X}_{r_3})
$$

$$
\tilde{V}_j = \tilde{X}_{r_1} + F^*(\tilde{X}_{r_2} - \tilde{X}_{r_3} + \tilde{X}_{r_4} - \tilde{X}_{r_5})
$$

$$
V_j = \tilde{X}_{\text{best}} + F^*(\tilde{X}_{r_1} - \tilde{X}_{r_2})
$$

where $r_1, r_2, r_3, r_4, r_5 \in \{1, 2, \ldots, \text{NP(population size)}\}$ are randomly chosen integers, which are different from each other and also different from the running index $i$. $\tilde{X}_{\text{best}}$ represents the best individual in the current generation. $F (>0)$ is a scaling factor which controls the amplification of the differential vector.

After mutation, DE performs a binomial crossover operator on $\tilde{X}_i$ and $\tilde{V}_j$ to generate a trial vector $\tilde{U}_j$.
Improving Clustering-based Differential Evolution with Chaotic Sequences and New Mutation Operator
Gang Liu, Yuanxiang Li, Xin Nie, Yu Sun
International Journal of Advancements in Computing Technology Volume 3, Number 6, July 2011

\[
    u_j = \begin{cases} 
    v_j & \text{if } \text{rand}(0,1) < Cr \text{ or } j = j_{\text{rand}} \\ 
    x_j & \text{otherwise}
    \end{cases}
\]  

(4)

where \(i= 1, 2, \ldots, NP, j = 1, 2, \ldots, D\) (D-dimensional), \(j_{\text{rand}}\) is a randomly chosen integer in \([1, D]\). \(\text{rand}(0, 1)\) is a uniformly distributed random number between 0 and 1 which is generated for each \(j\), and \(Cr\in[0, 1]\) is called the crossover rate. Due to the use of \(j_{\text{rand}}\), the trial vector \(\vec{U}_i\) differs from its target vector \(\vec{X}_i\). The selection operator is performed to select the better one from the target vector \(\vec{X}_i\) and the trial vector \(\vec{U}_i\) to enter the next generation.

Recently, many researchers are working the improvement of DE, and many new DEs are presented. In particular, the methods using k-means clustering are listed in the literature [18]. The neighborhood concept has been utilized in the context of the differential evolution algorithm. This method is listed in the literature [11].

2.2. K-means clustering and related work

The k-means clustering algorithm [19] partitions data points into \(k\) clusters \(S_i\) (\(i = 1, 2, \ldots, k\)) and the cluster \(S_i\) are associated with representatives (cluster centers) \(C_i\). The major process of k-means clustering is mapping a given set of representative vectors into an improved one through partitioning data points. It begins with an initial set of cluster centers and repeats this mapping process until a stopping criterion is satisfied. The k-means clustering can be described in [18]. The distance measure used in the clustering algorithm is a very important issue. The most widely used distance measure is the Euclidean distance. Other distance measures can be used. In this paper, the Euclidean distance is used. The k-means clustering algorithm is used in many aspects, such as internet traffic classification [20], optimal color image enhancement [21], wireless sensor network [22].

3. Improving clustering-based differential evolution with chaotic sequences and new mutation operator (CCDE)

The characteristics of the trial vector generation strategies and the control parameters of DE have a great impact on the performance of search. The appropriate mutation scheme makes DE have capacity of fulfilling global optimization tasks and accelerate the convergence rate. Based on the above considerations, this paper attempts to improve DE by integrating the one-step k-means clustering algorithm and chaotic sequences. The proposed DE algorithm is named CCDE. One-step k-means clustering algorithm is used to get the key information of the population. The cluster centers show the characteristics of the current population. In fact, each cluster is a sub-population and each sub-population can be regard as the neighborhood. In this paper, we propose two kinds of mutation models for DE. The first one is called the cluster neighborhood mutation model, where each vector is mutated using the cluster centers. The second one is called the global mutation model, using the globally best vector of the entire population at current generation for mutating a population member. The cluster neighborhood and the global model are combined by using a weight factor that appears as a new parameter in the algorithm like in the literature [11]. And there are 3 trial vectors in CCDE. They are the first vector created by the cluster neighborhood mutation model, the second vector created by the global mutation model and the third vector created by combination of both. The best vector in 3 trial vectors is selected to replace the target vector. Chaotic sequences are used to improve the control parameters and also used in the population initialization.

3.1. Chaotic sequences using logistic equation in CCDE

The performance of DE is sensitive to the choice of control parameters. The choice of these control parameters in DE is mainly based on empirical evidence and practical experience. However, it is
difficult to properly set control parameters in DE. In CCDE, chaotic sequences are used to dynamically adjust F, Cr and design the mutation operators. Due to the ergodicity property, chaos can be used to enrich the searching behavior and to avoid being trapped into local optimum in optimization problems. In this paper, chaotic sequences using logistic equation are incorporated into the DE.

The well-known logistic equation [14, 15], which exhibits the sensitive dependence on initial conditions, is used in the mutation operators and the population initialization. The logistic equation is defined as follows:

\[
y(k) = \mu \cdot y(k-1) \cdot [1 - y(k-1)]
\]

where \( k \) is the sample, and \( \mu \) is a control parameter, \( 0 < \mu \leq 4 \). The behavior of the system of (5) is greatly changed with the variation of \( \mu \). A very small difference in the initial value of \( y \) causes large differences in its long-time behavior. When \( \mu = 4 \) and \( y(1) \not\in \{0, 0.25, 0.50, 0.75, 1\} \), The equation (5) is deterministic, displaying chaotic dynamics. In this work, \( y(k) \) is distributed in the range \([0, 1]\) provided the initial \( y(1) \not\in \{0, 0.25, 0.50, 0.75, 1\} \). Fig. 1 shows the map graphic with 200 iterations.

![Logistic map beginning with a random number that is distributed in the range [0, 1].](image)

3.2. Chaotically initialized population

The population in DE can be guided towards the more promising areas if the initial population can be spread as much as possible over the objective function surface. So, sequences generated from chaotic systems substitute random numbers for the DE population. Chaotic sequences in DE ensure the individuals in population to be spread in the search spaces as much as possible for population diversity. The equation used in the population initialization is as follows:

\[
x_j = x_{\min} + c'_j \cdot (x_{\max} - x_{\min})
\]

where \( j=1, 2, \ldots, D \) and \( D \) is the number of problem decision variables. \( c'_j \) is the chaotic variable generated by the equation (5) for each decision variable. \( x_{\max} \) and \( x_{\min} \) are the upper bounds and lower bounds of the problem decision variables. This method is also used in [15].

Algorithm 1 shows the main steps of generating chaotically initialized population. NP is the population size.

**Algorithm 1: Algorithm for chaotically initialized population**

Step1: Set NP and D, randomly initialize chaotic variable \( c'_1 \).

Step2: For \( i=1 \) to NP do
For \( j = 1 \) to \( D \) do
\[
x_j = x_{\min} + c_j' \ast (x_{\max} - x_{\min})
\]
Generate next chaotic variable \( c_j' \).
End for

### Step3: End for

#### 3.3 The cluster neighborhood mutation model and the global mutation model

In this work, the one-step k-means clustering is chosen because of its simplicity and linear time complexity. \( K \) is the number of clusters and is generated from \([2, \sqrt{NP}]\) randomly. It is a rule of thumb used by many investigators in the literature [23]. According to the literature [18], the multi-step k-means clustering needs more computational time and it does not bring any important advantage. Other clustering approaches can also be employed as well.

For each member of the cluster, a local donor vector is created by employing the cluster center and any two other vectors chosen from the population. Similarly, the global donor vector is created. The two models may be expressed as:

\[
\bar{L}_{r,G} = \bar{X}_{r1,G} + F \ast (\bar{C}_{i,G} - \bar{X}_{r2,G} + \bar{X}_{r3,G})
\]
(7)

\[
\bar{g}_{i,G} = \bar{X}_{r,G} + F \ast (\bar{X}_{best,G} - \bar{X}_{r4,G} + \bar{X}_{r5,G})
\]
(8)

where \( \bar{C}_{i,G} \) is the cluster center of the cluster of \( \bar{X}_{i,G} \), \( \bar{X}_{best,G} \) is the best vector in the entire population at generation \( G \). And \( r1 \neq r2 \neq r3 \neq i, r \neq r4 \neq r5 \neq i, r, r1, r2, r3, r4, r5 \in [1, NP] \). \( F \) is randomly selected in 0.5 and the chaotic variable generated by the equation (5). \( F \) in CCDE is as follows:

\[
F = \begin{cases} 
0.5 & \text{if } \text{rand}_f = 0 \\
1 & \text{if } \text{rand}_f = 1 
\end{cases}
\]
(9)

where \( \text{rand}_f \) is a random integer and it is selected in 0 and 1. The fitness of \( \bar{X}_{r1} \) and \( \bar{X}_r \) are not worse than the fitness of \( \bar{X}_i \). This will help maintain population diversity while maintaining convergence rate. It is an eclectic selecting method [3].

The local and global donor vectors are combined by using a weight factor \( w \in (0,1) \). It forms the new donor vector of the proposed algorithm. It is as follows:

\[
\bar{V}_{i,G} = w \ast \bar{g}_{i,G} + (1 - w) \ast \bar{L}_{i,G}
\]
(10)

Optimal values of the weight factor will always depend on the problem at hand. For simplicity, chaotic sequences are used to dynamically adjust \( w \). It is as follows:

\[
\bar{V}_{i,G} = c \ast \bar{g}_{i,G} + (1 - c) \ast \bar{L}_{i,G}
\]
(11)

where \( c \) is the chaotic variable generated by the equation (5).
It is worth pointing out that 3 new trial vectors are generated in CCDE. The equation (7), (8) and (11) generate 3 new trial vectors respectively. And then the best vector in 3 new trial vectors is selected to replace the target vector in the population. This method is effective to improve the search efficiency of DE. The advantages of cluster neighborhood do not need to assign the neighborhood size and do not need to consider population topology.

3.4 Self-adaptive crossover rate

In CCDE, the crossover rate is randomly selected in 0.9 and the chaotic crossover rate. This chaotic crossover rate is as follows:

\[
0.1 + 0.9 \times c
\]

It means this crossover rate is a random number from [0.1, 1] and it is similar to the equation (6). This is a flexible approach. 0.9 can maintain a high probability of generating new individuals and it is used in many papers. But it is not applied in all cases. The new chaotic crossover rate can remedy this problem. The crossover rate in CCDE is as follows:

\[
cr = \begin{cases} 
0.9 & \text{if } rand_{cr} = 0 \\
0.1 + 0.9 \times c & \text{if } rand_{cr} = 1 
\end{cases}
\]

where \( rand_{cr} \) is a random integer and it is selected in 0 and 1.

The CCDE algorithm is described in Algorithm 2, where \( t \) is the iteration counter; \( \text{rndint}[2, \sqrt{NP}] \) is a random integer from [2, \( \sqrt{NP} \)].

Algorithm 2: The CCDE algorithm

Step1: Generate the initial population \( P \) using the algorithm 1. Evaluate each individual in \( P \), \( t = 1 \).

Step2: While the halting criterion is not satisfied do

Step3: Randomly generate \( k = \text{rndint}[2, \sqrt{NP}] \), one-step k-means clustering is used to create \( k \) the cluster centers.

Step4: For \( i = 1 \) to \( NP \) do

Randomly generate \( F \) and \( Cr \) using the equation (9), (13). The equation (7), (8) and (10) generate 3 new trial vectors respectively. The best vector in 3 new trial vectors is selected to replace the target vector in the population. Continue to generate different chaotic variables according to the equation (5).

End for

Step5: \( t = t + 1 \).

Step6: End while

Step7: Output the best vector.

Compared with the technique in [11], our approach has five main characteristics: (1) the one-step k-means clustering algorithm is used to create the cluster neighborhood and it is not need to define a neighborhood of radius. In [11], it need to define a neighborhood of radius; (2) Two kinds of mutation models for DE are different with the ones in [11]; (3) Chaotic sequences are used to design the self-adaptive scaling factor, the self-adaptive weight factor and the self-adaptive crossover rate in the CCDE. In [11], chaotic sequences are not used; (4) Chaotic sequences are used for population initialization. In [11], it is random population initialization; (5) 3 new trial vectors are generated respectively and the best vector in 3 new trial vectors is selected to replace the target vector in the population. In [11], only one trial vector is generated.

Hence, the CCDE algorithm can improve search efficiency and the ability to find the optimal solution. And CCDE is simplicity, efficiency and flexibility.
4. Experimental results and discussions

In this section, we perform an extensive experimental evaluation of the proposed algorithm. We employ the CEC 2005 benchmark suite which consists of 25 scalable benchmark functions [24]. A thorough description of this test set is provided in [24]. The number of variables of all the functions is scalable (up to 50), and we have considered in this paper dimensions $D=30$ for all the problems.

CCDE was compared with four other state-of-the-art DE variants and DE. They are DEGL [11], DE-GPBX-$\alpha$ [25], SaDE [16] and EPSDE [26]. In our experiments, we used the same parameter settings for these methods as in their original papers. The number of function evaluations (NFEs) in all these methods was set to 300,000, as the same as in CCDE. The population size in CCDE was set to 100, as the same as in DE. In DE, $F=0.5$ and $Cr=0.9$. Each algorithm was executed independently 50 times, to obtain an estimate of the mean solution error and its standard deviation.

To evaluate the statistical significance of the observed performance differences a two-sided Wilcoxon rank sum test at a 0.05 significance level was conducted on the experimental results. At the bottom of each table, for each pair, we also show the total number of the aforementioned statistical significant cases (+/=/−). “+”, “−”, and “=” denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of CCDE, respectively. Figures in bold font represent the best results we found for every problem.

The experimental results are given in Table I and Table II. In order to study the impact of different crossover rates, CCDE with the fixed crossover rate is also listed in Table I for comparison. The fixed crossover rate is 0.9. For simplicity, CCDE with the self-adaptive crossover rate is called CCDE.

For unimodal functions F1-F5, we can see that two kinds of CCDE performs better than DE, DE-GPBX-$\alpha$, EPSDE, DEGL and SaDE on three (i.e. F2-F4), five (i.e.F1-F5), three (i.e. F3-F5), two (i.e.F4-F5) and four (i.e.F2-F5) test functions, respectively. DEGL is better than other algorithms on F2-F3. The outstanding performance of DEGL should be due to its neighborhood strategy, which leads to get better results on F2-F3. DE-GPBX-$\alpha$ cannot outperform CCDE on any test function. CCDE with the fixed crossover rate outperforms CCDE with the self-adaptive crossover rate on three functions (i.e.F2-F4). It shows that the fixed crossover rate is more suitable for unimodal functions.

For basic multimodal functions F6-F12, CCDE is significantly better than DE, DE-GPBX-$\alpha$, EPSDE, DEGL and SaDE on three (i.e.F9-F11), five (i.e.F6-F7, F10-F12), five (i.e.F6-F7, F10-F12), six (i.e.F6-F12) and five (i.e.F6-F7, F10-F12) test functions, respectively. For F8, the results obtained by all algorithms are not very different. For F12, the results obtained by CCDE and DE are the same. Thus, CCDE is the winner on these seven test functions. This can be because CCDE could balance exploration and exploitation on these test functions by combining different trial vector generation strategies with different control parameter settings. CCDE with the self-adaptive crossover rate performs better than CCDE with the fixed crossover rate on four functions (i.e.F6, F9-F11). For F7 and F8, the results obtained by two kinds of CCDE are similar.

For expanded multimodal functions F13–F14, two kinds of CCDE exhibit similar performance and outperform other methods. DE-GPBX-$\alpha$ and DEGL are not very different with CCDE on F13, F14, respectively.

For hybrid composition functions F15–F25, these test functions are much harder than others since each of them is composed of 10 sub-functions. Overall, the performance of CCDE is better than that of the five competitors. It outperforms DE, DE-GPBX-$\alpha$, EPSDE, DEGL and SaDE on seven (i.e. F15-F20, F22), seven (i.e.F16-F20, F22, F25), six (i.e.F16-F17, F21, F23, F25), eleven (i.e.F15-F25) and four (i.e. F15, F21, F22, F25) test functions, respectively. DEGL cannot perform better than CCDE even on one test function. For F18-F20, SaDE is significantly better than all methods according to the Wilcoxon’s rank sum test. But for F15-F17, F21-F25, SaDE is not significantly better than CCDE according to the Wilcoxon’s rank sum test. CCDE with the self-adaptive crossover rate performs better than CCDE with the fixed crossover rate on eight functions (i.e.F15-F19, F21, F23, F25). For hybrid composition functions, the fixed crossover rate is not suitable and the self-adaptive crossover rate is better than it.

In summary, CCDE is the best among the five methods in comparison on basic multimodal functions, expanded multimodal functions, and hybrid composition functions. The last one row in Table I and Table II indicates that, overall, CCDE is better than the five competitors according to the
Wilcoxon’s rank sum test. CCDE with the fixed crossover rate is dominant for unimodal functions. But CCDE with the self-adaptive crossover rate is dominant in the overall.

The evolution of the mean function error values derived from DE, DE-GPBX-α, DEGL, EPSDE, SaDE, and CCDE versus the number of NFEs is plotted in Figure. 2 for some typical test functions.

| Table 1. Experimental Results of CCDE, DE, DE-GPBX-α Over 50 Independent Runs on 25 Test Functions of 30 Variables With 300 000 NFEs |
|-----------------|-----------------|-----------------|-----------------|-----------------|
|                 | CCDE (Self-adaptive cr) | CCDE (cr=0.9)   | DE/rand/1/bin   | DE-GPBX-α       |
|                 | Mean | St.D | Mean | St.D | Mean | St.D | Mean | St.D |
| F1               | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 2.84E-15 | 1.25E-14 |
| F2               | 5.67E-10 | 6.17E-10 | 1.15E+16 | 3.41E+16 | 5.39E-05 | 1.35E-04 | 3.34E+01 | 5.17E+01 |
| F3               | 1.88E+05 | 1.24E+05 | 1.06E+05 | 5.60E+04 | 3.89E+05 | 2.14E+05 | 8.70E+06 | 6.21E+06 |
| F4               | 2.73E-05 | 3.27E-05 | 7.66E-08 | 1.96E-07 | 1.82E-02 | 2.14E-02 | 7.01E+02 | 7.26E+02 |
| F5               | 2.23E+02 | 2.25E+02 | 3.20E+02 | 2.81E+02 | 7.97E+01 | 1.06E+02 | 2.03E+03 | 6.26E+02 |
| F6               | 3.19E-01 | 1.09E+00 | 8.77E-01 | 1.67E+00 | 1.19E-01 | 2.49E+01 | 5.04E+01 | 4.34E+01 |
| F7               | 1.11E-02 | 1.03E-02 | 1.12E-02 | 8.95E-03 | 1.05E-03 | 3.23E-03 | 4.70E+03 | 4.09E-13 |
| F8               | 2.09E+01 | 6.13E-02 | 2.09E+01 | 7.67E-02 | 2.10E+01 | 5.74E-02 | 2.09E+01 | 5.54E-02 |
| F9               | 2.11E+01 | 7.53E+00 | 4.33E+01 | 1.20E+01 | 1.39E+02 | 2.38E+01 | 8.12E+00 | 4.37E+00 |
| F10              | 4.50E+01 | 1.33E+01 | 5.11E+01 | 1.65E+01 | 1.83E+02 | 9.61E+01 | 1.01E+02 | 3.95E+01 |
| F11              | 1.45E+01 | 3.99E+00 | 1.70E+01 | 5.05E+00 | 3.95E+01 | 1.07E+00 | 2.71E+01 | 5.66E+00 |
| F12              | 1.70E+03 | 1.95E+03 | 1.40E+03 | 2.14E+03 | 1.70E+03 | 1.78E+03 | 7.11E+03 | 6.34E+03 |
| F13              | 2.58E+00 | 6.41E-01 | 2.82E+00 | 6.75E+01 | 1.52E+01 | 1.11E+01 | 2.56E+01 | 1.29E+00 |
| F14              | 1.25E+01 | 3.76E+01 | 1.22E+01 | 5.70E+01 | 1.33E+01 | 1.35E+01 | 1.31E+01 | 2.28E+01 |
| F15              | 3.37E+02 | 1.06E+02 | 3.60E+02 | 1.04E+02 | 4.02E+02 | 5.53E+01 | 3.13E+02 | 7.43E+01 |
| F16              | 9.04E+01 | 8.12E+01 | 1.31E+02 | 1.26E+02 | 2.04E+02 | 1.04E+01 | 1.58E+02 | 9.11E+01 |
| F17              | 8.61E+01 | 7.12E+01 | 1.13E+02 | 9.88E+01 | 2.30E+02 | 1.43E+01 | 2.17E+02 | 8.20E+01 |
| F18              | 8.93E+02 | 4.11E+01 | 8.96E+02 | 4.25E+01 | 9.02E+02 | 1.48E+01 | 9.06E+02 | 1.84E+00 |
| F19              | 8.91E+02 | 3.06E+01 | 9.03E+02 | 3.49E+01 | 9.02E+02 | 1.48E+01 | 9.06E+02 | 1.85E+00 |
| F20              | 8.96E+02 | 3.65E+01 | 8.88E+02 | 4.76E+01 | 9.02E+02 | 1.48E+01 | 9.06E+02 | 1.81E+00 |
| F21              | 5.06E+02 | 4.24E+01 | 5.21E+02 | 1.05E+02 | 5.00E+02 | 3.19E+03 | 5.03E+02 | 3.00E+01 |
| F22              | 8.92E+02 | 1.32E+01 | 8.92E+02 | 1.33E+01 | 9.08E+02 | 9.29E+00 | 9.06E+02 | 1.84E+00 |
| F23              | 5.42E+02 | 5.71E+01 | 5.52E+02 | 8.01E+01 | 5.34E+02 | 2.64E+04 | 5.34E+02 | 3.31E-03 |
| F24              | 2.00E+02 | 2.87E+14 | 2.00E+02 | 4.46E+13 | 2.00E+02 | 2.87E+14 | 2.00E+02 | 8.92E+13 |
| F25              | 2.10E+02 | 3.92E-01 | 2.16E+02 | 4.10E+01 | 2.09E+02 | 3.35E+01 | 1.39E+03 | 3.03E+01 |

From Figure.2, it can be seen that for the test functions CCDE converges faster than the five competitors.

In view of the above discussion it can be concluded that the overall performance of CCDE is better than the five competitors for the CEC2005 benchmark functions.
5. Conclusion

In this paper, we proposed the cluster neighborhood mutation operator, the global mutation operator and a linear combination of both operators, in an attempt to balance their effects. CCDE employed three trial vector generation strategies and self-adaptive parameter settings. The chaotic sequences using logistic equation are used to design the mutation operators and to generate chaotically initialized population. And the chaotic sequences are also designed the self-adaptive crossover rate.

The experimental studies in this paper were carried out on 25 global numerical optimization problems used in the CEC2005 special session on real-parameter optimization. CCDE was compared with four other state-of-the-art DE variants and DE/rand/1/bin. These DE variants are DEGL, DE-GPBox-α, SaDE and EPSDE. The experimental results suggested that its overall performance was better than the five competitors. In addition, the effectiveness of fixed crossover rate for the trial vector generation strategies was experimentally studied. CCDE with the self-adaptive crossover rate is dominant in the overall.

In the future, we will generalize our work to other EAs for other hard optimization problems. The self-adaptive parameter settings will be further studied. Furthermore, the idea of coevolutionary algorithm [27] will be also used for the improvement of DE.

Table 2. Experimental Results of CCDE, DEGL, EPSDE and SaDE Over 50 Independent Runs on 25 Test Functions of 30 Variables With 300 000 NFES

<table>
<thead>
<tr>
<th>CCDE</th>
<th>EPSDE</th>
<th>DEGL</th>
<th>SaDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>St.D</td>
<td>Mean</td>
<td>St.D</td>
</tr>
<tr>
<td>F1</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>F2</td>
<td>5.67E-10</td>
<td>6.17E-10</td>
<td>4.23E-26</td>
</tr>
<tr>
<td>F3</td>
<td>1.88E+05</td>
<td>1.24E+05</td>
<td>8.74E+05</td>
</tr>
<tr>
<td>F4</td>
<td>2.73E-05</td>
<td>3.27E-05</td>
<td>3.49E+02</td>
</tr>
<tr>
<td>F5</td>
<td>2.23E+02</td>
<td>2.25E+02</td>
<td>1.40E+03</td>
</tr>
<tr>
<td>F6</td>
<td>3.19E-01</td>
<td>1.09E+00</td>
<td>6.38E-01</td>
</tr>
<tr>
<td>F7</td>
<td>1.11E-02</td>
<td>1.03E-02</td>
<td>1.77E-02</td>
</tr>
<tr>
<td>F8</td>
<td>2.09E+01</td>
<td>6.13E-02</td>
<td>2.09E+01</td>
</tr>
<tr>
<td>F9</td>
<td>2.11E+01</td>
<td>7.53E+00</td>
<td>3.99E-02</td>
</tr>
<tr>
<td>F10</td>
<td>4.50E+01</td>
<td>1.33E+01</td>
<td>5.36E+01</td>
</tr>
<tr>
<td>F11</td>
<td>1.45E+01</td>
<td>3.99E+00</td>
<td>3.56E+01</td>
</tr>
<tr>
<td>F12</td>
<td>1.70E+03</td>
<td>1.95E+03</td>
<td>3.58E+04</td>
</tr>
<tr>
<td>F13</td>
<td>2.58E+00</td>
<td>6.41E-01</td>
<td>1.94E+00</td>
</tr>
<tr>
<td>F14</td>
<td>1.25E+01</td>
<td>3.76E+00</td>
<td>1.35E+01</td>
</tr>
<tr>
<td>F15</td>
<td>3.37E+02</td>
<td>1.06E+02</td>
<td>2.12E+02</td>
</tr>
<tr>
<td>F16</td>
<td>9.04E+01</td>
<td>8.12E+01</td>
<td>1.22E+02</td>
</tr>
<tr>
<td>F17</td>
<td>8.61E+01</td>
<td>7.12E+01</td>
<td>1.69E+02</td>
</tr>
<tr>
<td>F18</td>
<td>8.93E+02</td>
<td>4.11E+01</td>
<td>8.20E+02</td>
</tr>
<tr>
<td>F19</td>
<td>8.91E+02</td>
<td>3.06E+01</td>
<td>8.21E+02</td>
</tr>
<tr>
<td>F20</td>
<td>8.96E+02</td>
<td>3.65E+01</td>
<td>8.22E+02</td>
</tr>
<tr>
<td>F21</td>
<td>5.66E+02</td>
<td>4.24E+01</td>
<td>8.33E+02</td>
</tr>
<tr>
<td>F22</td>
<td>8.92E+02</td>
<td>1.32E+02</td>
<td>5.07E+02</td>
</tr>
<tr>
<td>F23</td>
<td>5.42E+02</td>
<td>5.71E+01</td>
<td>8.15E+02</td>
</tr>
<tr>
<td>F24</td>
<td>2.00E+02</td>
<td>2.87E+01</td>
<td>2.13E+02</td>
</tr>
<tr>
<td>F25</td>
<td>2.10E+02</td>
<td>3.92E+01</td>
<td>2.13E+02</td>
</tr>
</tbody>
</table>

Total number of (f−/f−/): 8/2/15 3/2/20 4/8/13
Improving Clustering-based Differential Evolution with Chaotic Sequences and New Mutation Operator
Gang Liu, Yuanxiang Li, Xin Nie, Yu Sun
International Journal of Advancements in Computing Technology Volume 3, Number 6, July 2011

Figure 2. Evolution of the mean function error values derived from CCDE, DE, DE-GPBX-α, EPSDE, DEGL, and SaDE versus the number of FES on ten test functions.

6. References

Improving Clustering-based Differential Evolution with Chaotic Sequences and New Mutation Operator
Gang Liu, Yuanxiang Li, Xin Nie, Yu Sun
International Journal of Advancements in Computing Technology Volume 3, Number 6, July 2011