Improving the JADE Algorithm by Clustering Successful Parameters

Zhijian Li
School of Computer Science, 
Central China Normal University, Wuhan 430079, China 
E-mail: lzj1769@163.com

Jinglei Guo*
School of Computer Science, 
Central China Normal University, Wuhan 430079, China 
E-mail: guojinglei@mail.ccnu.edu.cn 
*Corresponding author

Shengxiang Yang
Centre for Computational Intelligence (CCI) 
School of Computer Science and Informatics, De Montfort University, Leicester 
LE1 9BH, U.K. 
E-mail: syang@dmu.ac.uk

Abstract: Differential evolution (DE) is one of the most powerful and popular evolutionary algorithms for real parameter global optimization problems. However, the performance of DE highly depends on the selection of control parameters, e.g., the population size, scaling factor and crossover rate. How to set these parameters is a challenging task because they are problem dependent. In order to tackle this problem, a JADE variant, denoted CJADE, is proposed in this paper. In the proposed algorithm, the successful parameters are clustered with the K-means clustering algorithm to reduce the impact of poor parameters. Simulation results show that CJADE is better than, or at least comparable to, several state-of-the-art DE algorithms.

Keywords: differential evolution algorithm; k-means; successful parameters.

Biographical notes: Zhijian Li received the M.S. degree in Computer Science from the Central China Normal University in 2016. His current research interests focus on evolutionary computation, optimization algorithm and machine learning.

Jinglei Guo received the Ph.D. in Computer Science from Wuhan University in 2011. Currently, she is an Associate Professor at the School of Computer, Central China Normal University. Her research interests include artificial intelligence and evolutionary computation.

Shengxiang Yang received the B.Sc. and M.Sc. degrees in automatic control and the Ph.D. degree in system engineering from Northeastern University, China. He is currently a Professor in Computational Intelligence and Director of the Centre for Computational Intelligence, School of Computer Science and Informatics, De Montfort University, U.K. His current research interests include evolutionary and genetic algorithms, swarm intelligence, computational intelligence in dynamic and uncertain environments, artificial neural networks for scheduling, and relevant real-world applications.

1 Introduction

Differential evolution (DE), proposed in R. Storn and K. Price (1995), is a simple and efficient evolutionary algorithm (EA) for global optimization in the continuous space. However, unlike other EAs, DE perturbs the current population members with the scaled difference of randomly selected distinct population members (Das and Suganthan, 2011). DE has been successfully applied to various fields of real-world optimization problems, such as pattern recognition (Koloseni et al., 2012; Kim et al., 2008), image processing (Su et al., 2012), and engineering design (Ghosh et al., 2012; Guo et al., 2001). A detailed survey on DE can be found in Das and Suganthan (Das and Suganthan, 2011).
There are three main control parameters within the DE algorithm: the mutation scale factor $F$, the crossover constant $Cr$, and the population size $NP$. These parameters have significant impact on the performance of DE. In order to minimize the effects of these parameters, a number of enhanced DE variants have been proposed in the past decade. Most of those algorithms are based on adaptive or self-adaptive mechanisms and consider $F$ and $Cr$ separately. In this paper, we propose a new self-adaptive scheme, where $F$ and $Cr$ are considered as a point in the two-dimensional space and we utilize the cluster analysis to reduce the impact of poor parameters and to update the control parameters automatically.

The rest of this paper is organized as follows. In Section 2, we introduce the classical DE algorithm and related work. JADE is described in Section 3. The proposed approach is introduced in detail in Section 4. Experimental studies are reported in Section 5. Finally, Section 6 concludes this paper.

## 2 Background

### 2.1 Classical differential evolution

DE begins with a randomly initiated population of $NP$ individuals, each of which is a $D$-dimensional real parameter vector to represent a candidate solution, where $NP$ is the population size and $D$ is the number of dimensions of the problem. The $j$th dimension of the $i$th individual can be initialized according to:

$$x_{j,i} = x_{j,\text{min}} + \text{rand}_{i,j}(0,1) \cdot (x_{j,\text{max}} - x_{j,\text{min}})$$

where $x_{j,\text{min}}$ and $x_{j,\text{max}}$ are the predefined lower and upper bounds of the $j$th dimension, $\text{rand}_{i,j}(0,1)$ is a uniformly distributed random number in $(0,1)$.

After initialization, DE generates a mutant individual $V_{i,g}$, called donor individual, for each population member or target individual $X_{i,g}$ in the current generation by mutation. The most frequently used strategy is defined as follows:

$$V_{i,g} = X_{r_1,g} + F \cdot (X_{r_2,g} - X_{r_3,g})$$

where the indices $r_1, r_2, r_3$ are mutually exclusive integers randomly chosen from the range $[1, NP]$, which are also different from the base individual index $i$. The scaling factor $F$ is a positive real number for scaling the difference vectors.

Then, DE employs crossover operator to enhance the potential diversity of the population. In the classical version, a binomial crossover operation is employed by DE as follows:

$$u_{i,j,g} = \begin{cases} u_{i,j,g}, & \text{if } (\text{rand}_{i,j}(0,1) \leq Cr \text{ or } j = j_{\text{rand}}) \\ x_{i,j,g}, & \text{otherwise} \end{cases}$$

where $j_{\text{rand}} \in [1, 2, \ldots, D]$ is a randomly chosen index, which ensures that $u_{i,j,g}$ dose not duplicate $x_{i,j,g}$.

Finally, a selection operator is used by DE to determine whether the target or trial vector survives to the next generation. The selection operation is described as follows:

$$X_{i,g+1} = \begin{cases} U_{i,g}, & \text{if } f(U_{i,g}) \leq f(X_{i,g}) \\ X_{i,g}, & \text{otherwise} \end{cases}$$

where $f(X)$ is the objective function (without loss of generality, we assume minimization problems here).

### 2.2 Related Work

Zhang and Sanderson (Zhang et al., 2009) proposed an adaptive DE with an optional external archive (JADE), where the scaling factor $F$ and the crossover rate $Cr$ are generated according to a normal distribution and a Cauchy distribution, respectively. Fei et al. (2009) applied two strategies together on the original JADE, to dedicatedly improve the reliability of it (rJADE). They first modified the control parameter adaptation strategy of JADE by adding a weighting strategy. Then, a restart with knowledge transfer strategy was applied by utilizing the knowledge obtained from previous failures to guide the subsequent search. Experimental studies showed that the proposed rJADE achieved significant improvements on a set of widely used benchmark functions. Li et al. (2015) proposed a predictive approach to the reproduction mechanism of new individuals for differential evolution (DE) algorithms utilizing cumulative correlation information already existing in an evolutionary process. DE uses a distributed model (DM) to generate new individuals, which is relatively exploitative, whilst evolution strategy (ES) uses a centralized model (CM) to generate offspring, which through adaptation retains a convergence momentum. Pan, X. and Li, R. (2015) proposed an improved differential evolution (DE) to solve parameter optimisation problems. The new approach is called ICBBDE, which is an enhanced version of bare bones DE (BBDE). The ICBBDE employs an adaptive strategy to dynamically adjust the crossover rate. Moreover, a Cauchy mutation is used to improve the exploration ability. Experiments are conducted on a set of benchmark functions and two real-world parameter optimisation problems. Simulation results demonstrate the efficiency and effectiveness of our approach. Zhang et al. (2015) presented a novel DE variant, Top-k elites-based Oppositional Differential Evolution (TEODE), which is based on a new opposition-based learning strategy using the top-k elites (TEOBL) in the current generation and employs similar schemes of ODE for population initialisation and generation jumping with TEOBL. Experiments are conducted on 17 benchmark functions. The results confirm that TEODE outperforms classical DE, ODE and COODE (opposition-based differential evolution using the current optimum). You et al. (2016) designed a new mutation operator to improve the exploitation ability of DE. Experiments are
carried out on 13 classical test functions. Simulation results show that the new mutation scheme can help DE to find better solutions than three other classical DE mutation strategies.

3 Introduction of JADE

JADE, proposed by Zhang and Sanderson Zhang et al. (2009), is a well-known adaptive DE, which uses a novel mutation strategy named DE/Current-to-pbest and an external archive for storing the inferior solutions. In addition, F and Cr are generated according to a normal distribution $N(\mu_{Cr}, 0.1)$ and a Cauchy distribution $C(\mu_F, 0.1)$, respectively.

3.1 DE/Current-to-pbest

In view of the fast convergence but less reliable performance of DE/current-to-best and DE/best/1 strategy, Zhang (Zhang et al. 2009) proposed a new mutation strategy, named DE/current-to-pbest, with an optional archive. In DE/current-to-pbest/1 (with archive), a mutation vector is generated in the following manner:

$$V_{i,g} = X_{i,g} + F_i(X^p_{\text{best},g} - X_{i,g}) + F_i(X_{i,1,g} - X_{i,2,g})$$

where $X^p_{\text{best},g}$ is randomly chosen as one of the top 100p% individuals in the current population with $p \in (0, 1]$, and $F_i$ is the mutation factor that is associated with $X_i$ and is re-generated at each generation by the adaptation process introduced later in Eq. (6).

3.2 Parameter Adaptation

At each generation $g$, the mutation factor $F_i$ and crossover probability $Cr_i$ of each individual $X_i$ is independently generated:

$$F_i = \text{randc}(\mu_F, 0.1)$$

$$Cr_i = \text{randn}(\mu_{Cr}, 0.1)$$

where $\text{randc}(\mu_F, 0.1)$ and $\text{randn}(\mu_{Cr}, 0.1)$ are the random number generated according to Cauchy distribution of $\mu_F$ and normal distribution of $\mu_{Cr}$, respectively. $F_i$ is truncated as 1 when $F_i > 1$ and re-generated when $F_i < 0$. $F_i$ is set as 1 or 0. $Cr_i$ is set as 1 or 0. Denote $S_F$ and $S_{Cr}$ as the set of all successful mutation factors and crossover probabilities at generation $g$. $\mu_F$ and $\mu_{Cr}$ are initialized to be 0.5 and then updated at the end of each generation as follows:

$$\mu_F = (1-c) \cdot \mu_F + c \cdot \text{mean}_L(S_F)$$  \hspace{1cm} (8)

$$\mu_{Cr} = (1-c) \cdot \mu_{Cr} + c \cdot \text{mean}_A(S_{Cr})$$  \hspace{1cm} (9)

where $c$ is a positive constant between 0 and 1, $\text{mean}_A(\cdot)$ is the arithmetic mean, and $\text{mean}_L(\cdot)$ is the Lehmer mean, defined as follows:

$$\text{mean}_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F}.$$  \hspace{1cm} (10)

Algorithm 1 CJADE algorithm

1: Initialization
2: $g := 0; \{\mu_F, \mu_{Cr}\} := \{0.5\}$
3: $\{\mu^1_{Cr}, \mu^2_{Cr}, \ldots, \mu^K_{Cr}\} := \{0.5\}; p := 0.05; A := 0;$
4: Initialize population $P_g = \{x_{1,g}, \ldots, x_{NP,g}\}$ randomly;
5: while The termination criteria are not met do
6: $S_F := \emptyset, S_{Cr} := \emptyset$;
7: for $i := 1$ to $NP$ do
8: Randomly choose $\mu_F, \mu_{Cr}$ from $\{\mu^1_F, \mu^2_F, \ldots, \mu^K_F\}$ and $\{\mu^1_{Cr}, \mu^2_{Cr}, \ldots, \mu^K_{Cr}\}$
9: $U_i := \text{randc}(\mu_F, 0.1)$;
10: $S_{Cr} := \text{random}(\mu_{Cr}, 0.1)$;
11: Generate $V_{i,g}$ according to Eq. (5);
12: Generate $U_{i,g}$ according to Eq. (3);
13: if $f(U_{i,g}) \leq f(X_{i,g})$ then
14: $X_{i,g+1} := U_{i,g}, X_{i,g} \rightarrow A$;
15: $F_i \rightarrow S_F, Cr_i \rightarrow S_{Cr}$;
16: else
17: $x_{i,g+1} := x_{i,g}$;
18: end if
19: end for
20: Remove solutions randomly from A so that $|A| \leq NP$;
21: if $S_F \neq \emptyset$ and $S_{Cr} \neq \emptyset$ then
22: Select randomly K initial cluster centers from points set $A := \{(F_i, Cr_i) | F_i \in S_F, Cr_i \in S_{Cr}\}$;
23: for $i := 1$ to $K$ do
24: for $j := 1$ to $|A|$ do
25: For each point $I_j$, find its closest cluster center and assign point $I_j$ to cluster;
26: end for
27: Update the cluster centers to be the average of points contained within them;
28: end for
29: K clusters $\{(C^1_F, C^1_{Cr}), (C^2_F, C^2_{Cr}), \ldots, (C^K_F, C^K_{Cr})\}$
30: for $i := 1$ to $K$ do
31: $\mu^1_F = (1-c) \cdot \mu^1_F + c \cdot \text{mean}_L(C^i_F)$
32: $\mu^1_{Cr} = (1-c) \cdot \mu^1_{Cr} + c \cdot \text{mean}_A(C^i_{Cr})$
33: end for
34: end if
35: $g := g + 1$
36: end while

4 THE PROPOSED APPROACH

As mentioned above, the performance of DE highly depends on the control parameters $F$ and $Cr$. Therefore, it appears increasing to the DE community to study adaptive or self-adaptive schemes to select suitable parameter values during the evolution process. A good volume of research work has been undertaken so far to improve the ultimate performance of DE by tuning its control parameters Das and Suganthan (2011). To the best of our knowledge, most of these work consider $F$ and $Cr$ separately. The trial vector, however, is a result of the combination of $F$ and $Cr$. Hence, in this paper, we consider each $F$ and $Cr$ as a point in the
two dimensional vector space and update the control parameters automatically by clustering the successful parameters with K-means method.

The parameters that generate individuals survived to the next generation are called successful parameters, which tend to generate individuals that are more likely to survive and thus should be propagated to the following generations. As with JADE, $S_F$ and $S_{cr}$ record the successful $F$ and $Cr$ in the current generation and a set of points is constructed as follows:

$$I = \{(F_i, Cr_i) | F_i \in S_F, Cr_i \in S_{cr}\}. \quad (11)$$

We utilize the $K$-means cluster algorithm to divide the set $I$ into $K$ clusters $\{(C_{r1}^{F},C_{cr1}^{F}), (C_{r2}^{F},C_{cr2}^{F}), \ldots, (C_{rK}^{F},C_{crK}^{F})\}$. For the $i$th cluster, $\mu_{r}^{F}$ and $\mu_{cr}^{F}$ are updated as follows:

$$\mu_{r}^{F} = (1-c) \cdot \mu_{r}^{F} + c \cdot \text{mean}_{L}(C_{r}^{F}) \quad (12)$$

$$\mu_{cr}^{F} = (1-c) \cdot \mu_{cr}^{F} + c \cdot \text{mean}_{A}(C_{cr}^{F}) \quad (13)$$

where $\mu_{r}^{F}$ and $\mu_{cr}^{F}$ are initialized to 0.5 at the beginning according to Zhang et al. (2009), and $\text{mean}_{L}(C_{r}^{F})$ and $\text{mean}_{A}(C_{cr}^{F})$ are calculated according to Eqs. (8) and (9) respectively. The overall CJADE algorithm is shown in Algorithm 1. Here, the number of iterations is set to a fixed value to simplify the $K$-means algorithm, and we use the Euclidean metric to measure the distance between two points. Obviously, JADE is a special case of CJADE with $K = 1$.

\section{EXPERIMENTAL STUDY}

\subsection{Benchmark Functions}

In order to verify the performance of our proposed approach, we employ the benchmark functions used for the 2014 IEEE Congress on Evolutionary Computation (IEEE CEC) Competition on Single Objective Real-parameter Numerical Optimization, which consists of 30 single objective benchmark functions Liang et al. (2013) and is denoted $\text{IEECEC2014 benchmark functions}$ in this paper. Based on their characteristics, the IEEE CEC 2014 benchmark functions can be divided into the following four classes. Functions $cf_{1} - cf_{3}$ are unimodal functions, functions $cf_{4} - cf_{16}$ are simple multimodal functions, functions $cf_{17} - cf_{22}$ are hybrid, and functions $cf_{23} - cf_{30}$ are composition functions with a huge number of local minima. A thorough description of this suit is provided in Liang et al. (2013).

\subsection{Effect of the Number of Iterations and Cluster Centers}

CJADE introduces a new parameter that need to be predefined: the number of clusters $K$ in the cluster analysis. We compare the performance of $K = 3$ with the performance of $K = 2$ in Table 1. It can be seen from Table 1 that the number of cluster centers has significant impact on the performance of CJADE on CEC2014 benchmark functions. Specifically, the results of $K = 2$ are better than those of $K = 3$ on 17 functions, while worse than those of $K = 3$ on 4 functions.

\subsection{Comparison of CJADE with Other DE Algorithms}

In this section, we compare CJADE with four other state-of-the-art DE algorithms. For fair comparison, we set the parameters of CJADE to be fixed: $p = 0.05$, $\text{Iterations} = 10$, $K = 2$ and $NP = 100$. We follow the parameter settings in the original paper of JADE (Zhang et al., 2009), EPSDE (Mallipeddi et al., 2011), CoDE (Wang et al., 2011), and MDE$_{pBX}$ (Islam et al., 2012). For all algorithms, the maximum number of function evaluations was set to 10000D. In addition, we judge the results by performing the Wilcoxon’s rank-sum test at the 0.05 significance level.

Table 2 summarizes the average error results of 51 independent runs for each algorithm on each 30-dimensional function. Error value smaller than $10^{-8}$ will be taken as zero (Liang et al., 2013). For each function, the best value of the results obtained by all the algorithms is shown in \textbf{bold font}. $b/n/w$ summarizes the statistical results: $b$, $n$, and $w$ denote the number of functions for which CJADE performs significantly better, not significantly different and significantly worse than its peer, respectively.

1. \textbf{Unimodal Functions} $cf_{1} - cf_{3}$: Clearly, CJADE is the best among the five methods on these three unimodal functions. It outperforms JADE, EPSDE, CoDE and MDE$_{pBX}$ on one, one, one and three test functions, respectively.

2. \textbf{Multimodal Functions} $cf_{4} - cf_{16}$: On these thirteen test functions, CJADE is significantly better than JADE, EPSDE, CoDE, and MDE$_{pBX}$ on four, twelve and ten test functions, respectively. MDE$_{pBX}$ outperforms CJADE on one test function, and JADE, EPSDE, and CoDE cannot be significantly better than CJADE on any test function. Hence, CJADE is the winner on these thirteen test functions.

3. \textbf{Hybrid Functions} $cf_{17} - cf_{22}$: For these six functions, CJADE is slightly better than JADE and CoDE. CJADE outperforms EPSDE on one function ($cf_{17}$) and is better than MDE$_{pBX}$ on five functions ($cf_{17} - cf_{19}, cf_{21}, cf_{22}$).

4. \textbf{Composition Functions} $cf_{23} - cf_{30}$: These composition functions are much harder than others because they can have different properties for different variables subcomponents Liang et al. (2013). The performance of JADE and EPSDE is slightly better than that of CJADE. In contrast,
Table 1  CJADE Results on CEC2014 benchmarks using various K

<table>
<thead>
<tr>
<th>Prob.</th>
<th>CJADE(K=2)</th>
<th>CJADE(K=3)</th>
<th>CJADE(K=2)</th>
<th>CJADE(K=3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cF1</td>
<td>2.15e+03 2.40e+03</td>
<td>4.39e+06 1.96e+06</td>
<td>= cF16 9.31e+00 3.18e-01</td>
<td>9.78e+02 2.72e-01</td>
</tr>
<tr>
<td>cF2</td>
<td>0.00e+00 0.00e+00</td>
<td>0.00e+00 0.00e+00</td>
<td>= cF17 1.40e+03 4.59e+02</td>
<td>4.60e+05 3.12e+05</td>
</tr>
<tr>
<td>cF3</td>
<td>0.00e+00 0.00e+00</td>
<td>0.00e+00 0.00e+00</td>
<td>= cF18 9.51e-01 4.05e+01</td>
<td>2.75e+02 2.46e-02</td>
</tr>
<tr>
<td>cF4</td>
<td>0.00e+00 0.00e+00</td>
<td>2.28e+01 3.02e+01</td>
<td>= cF19 4.83e+00 8.65e-01</td>
<td>6.58e+00 1.18e+00</td>
</tr>
<tr>
<td>cF5</td>
<td>2.03e+01 5.04e-02</td>
<td>2.02e+01 9.17e-02</td>
<td>= cF20 1.00e+03 1.94e-03</td>
<td>3.36e+01 2.92e+01</td>
</tr>
<tr>
<td>cF6</td>
<td>7.39e+00 3.38e+03</td>
<td>1.06e+01 1.76e+00</td>
<td>= cF21 1.77e+03 9.41e+03</td>
<td>6.70e+04 3.15e+04</td>
</tr>
<tr>
<td>cF7</td>
<td>5.32e-04 2.15e-03</td>
<td>4.92e-03 9.46e-03</td>
<td>= cF22 1.57e+02 7.01e+01</td>
<td>1.65e+02 6.11e+01</td>
</tr>
<tr>
<td>cF8</td>
<td>0.99e+00 0.00e+00</td>
<td>0.00e+00 0.00e+00</td>
<td>= cF23 3.15e+02 0.00e+00</td>
<td>3.15e+02 0.00e+00</td>
</tr>
<tr>
<td>cF9</td>
<td>2.26e+01 3.96e+00</td>
<td>3.23e+01 3.84e+00</td>
<td>= cF24 2.27e+02 4.55e+00</td>
<td>2.27e+02 3.18e+00</td>
</tr>
<tr>
<td>cF10</td>
<td>5.30e-03 9.06e-03</td>
<td>1.99e-01 7.71e-01</td>
<td>= cF7 2.08e+02 2.55e+01</td>
<td>2.10e+02 1.42e+00</td>
</tr>
<tr>
<td>cF11</td>
<td>1.58e+03 2.34e+02</td>
<td>1.88e+03 1.53e+02</td>
<td>= cF25 3.50e+02 4.65e+01</td>
<td>4.05e+02 6.09e+01</td>
</tr>
<tr>
<td>cF12</td>
<td>2.46e-01 4.02e+02</td>
<td>2.49e-01 4.31e+02</td>
<td>= cF26 7.99e+02 3.67e+01</td>
<td>8.52e+02 2.96e+01</td>
</tr>
<tr>
<td>cF13</td>
<td>2.13e-01 3.62e+02</td>
<td>2.43e-01 2.71e+02</td>
<td>= cF27 7.99e+02 3.67e+01</td>
<td>8.52e+02 2.96e+01</td>
</tr>
<tr>
<td>cF14</td>
<td>2.29e-01 3.53e+02</td>
<td>2.25e-01 3.62e+02</td>
<td>+ cF28 7.99e+02 3.67e+01</td>
<td>8.52e+02 2.96e+01</td>
</tr>
<tr>
<td>cF15</td>
<td>2.99e+00 4.29e+01</td>
<td>3.84e+00 4.31e+01</td>
<td>= cF29 1.64e+03 3.66e+02</td>
<td>2.30e+03 5.98e+02</td>
</tr>
<tr>
<td>Total number of (+/=/-)</td>
<td>4/9/17</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2  The experimental results of 30-dimensional problems cF1 – cF50

<table>
<thead>
<tr>
<th>b/n/w</th>
<th>CJADE</th>
<th>JADE</th>
<th>EPISODE</th>
<th>CoDE</th>
<th>MDE,pBX</th>
</tr>
</thead>
<tbody>
<tr>
<td>9/16/5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16/7/7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21/7/2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26/2/2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

†CJADE performs better than the algorithm at a 0.05 level of significance by the Wilcoxon rank-sum test.
‡CJADE performs worse than the algorithm at a 0.05 level of significance by the Wilcoxon rank-sum test.

CJADE outperforms CoDE and MDE,pBX on five and eight functions, respectively. simultaneously, which shows that CJADE is better than the four competitors.

Fig. 1 shows the convergence graphs for cF1, cF9, cF10 and cF12 in detail. Obviously, CJADE maintains faster convergence speed and higher solution precision.
As mentioned in Section 4, JADE is a special case of CJADE with $K = 1$. The procedure of CJADE is the same as JADE except for the adaptation of parameters. In this section, we will study the improved adaptation of parameters and show the advantage of CJADE compared with JADE. According to the results of Table 2, we take $c_{f3}$ as example. The evolutionary curves of solution error and parameters of CJADE and JADE are plotted in Fig. 2. When the objective function is separable, a value for $Cr$ from the range $(0.0, 0.2)$ is best because then each trial vector frequently competes with a target vector from which it differs by a single parameter (Ronkkonen et al., 2005). In contrast, a large value of $Cr$ is very suitable for non-separable problems, since in this case all parameters may have to be adjusted simultaneously for the search to remain efficient. $c_{f3}$ is an unimodal and non-separable functions (Liang et al., 2013). As shown in Fig. 2, CJADE has a larger mean value of $Cr$ of all individuals than JADE, which is why CJADE performs better than JADE on $c_{f3}$.

6 CONCLUSION

DE is an efficient population-based optimization algorithm. However, its performance highly depends on its control parameters. In this paper, a variant of JADE algorithm based on the clustering of successful parameters, denoted CJADE, is proposed. It considers each $F$ and $Cr$ as a point of the two-dimensional vector space and utilizes the K-means to cluster the successful parameters and to update the control parameters automatically.
The experimental studies are carried on 30 benchmark functions used for the 2014 IEEE CEC Competition on Single Objective Real-parameter Numerical Optimization, which includes simple unimodal and multimodal function, hybrid and composition functions. Compared with four DE variants, namely, JADE, jDE, MGBDE and RADE, CJADE shows good performance on the majority of the test functions.

Further research work includes the analysis of control parameters during the evolution process, as well as experiments with some other methods of cluster analysis, such as distribution-based and density-based clustering.

ACKNOWLEDGMENT

This work is supported by the self-determined research funds of CCNU from the colleges’ basic research and operation of MOE (No. CCNU15A05063).

References


