Hybrid Differential Evolution based on Fuzzy C-means Clustering

Wenyin Gong
School of Computer Science
China University of Geosciences
Wuhan, 430074 P.R. China
cug11100304@yahoo.com.cn

Zhihua Cai
School of Computer Science
China University of Geosciences
Wuhan, 430074 P.R. China
zhcai@cug.edu.cn

Charles X. Ling
Dept. of Computer Science
The University of Western Ontario, London,
ON, N6A 5B7, Canada
cling@csd.uwo.ca

Jun Du
Dept. of Computer Science
The University of Western Ontario, London,
ON, N6A 5B7, Canada
jdu42@csd.uwo.ca

ABSTRACT

In this paper, we propose a hybrid Differential Evolution (DE) algorithm based on the fuzzy C-means clustering algorithm, referred to as FCDE. The fuzzy C-means clustering algorithm is incorporated with DE to utilize the information of the population efficiently, and hence it can generate good solutions and enhance the performance of the original DE. In addition, the population-based algorithm-generator is adopted to efficiently update the population with the clustering offspring. In order to test the performance of our approach, 13 high-dimensional benchmark functions of diverse complexities are employed. The results show that our approach is effective and efficient. Compared with other state-of-the-art DE approaches, our approach performs better, or at least comparably, in terms of the quality of the final solutions and the reduction of the number of fitness function evaluations (NFFEs).

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems

General Terms

Algorithms

Keywords

Differential evolution, fuzzy C-means clustering, global optimization

1. INTRODUCTION

Without loss of generality, the global minimization problem can be formalized as a pair (S, f) , where $S \subseteq R^D$ is a bounded set on R^D and $f : S \rightarrow R$ is a D-dimensional real-valued function. The problem is to find a point $X^* \in S$ such that $f(X^*)$ is the global minimum on S [16]. More specifically, it is required to find an $X^* \in S$ such that

$$\forall X \in S : f(X^*) \leq f(X) \tag{1}$$

where f does not need to be continuous but it must be bounded. This paper only considers unconstrained function optimization. Generally, for each variable x_i it satisfies

$$l_i \leq x_i \leq u_i, i = 1, 2, \cdots, D \tag{2}$$

Global optimization problems are frequently arisen in almost every field of engineering design, applied sciences, molecular biology and other scientific applications.

Differential evolution (DE) [13] algorithm is a novel evolutionary algorithm (EA) for global optimization, where the mutation operator is based on the distribution of solutions in the population. It won the third place at the first International Contest on Evolutionary Computation on a real-valued function test-suite [14]. DE is a simple yet powerful population-based, direct search algorithm with the generation-and-test feature for globally optimizing functions using real-valued parameters. Among DE's advantages are its simple structure, ease of use, speed and robustness. Price & Storn [13] gave the working principle of DE with single scheme. Later on, they suggested ten different schemes of DE [14], [10]. The DE algorithm has been successfully applied to a whole host of engineering problems including the design of digital filters, mechanical design optimization, aerodynamic design and multiprocessor synthesis [14], [10], [2]. However, DE has been shown to have certain weaknesses, especially if the global optimum requires to be located using a limited number of fitness function evaluations (NFFEs). In addition, DE is good at exploring the search space and locating the region of global minimum, but it is slow at exploitation of the solution [9].

To remedy some weaknesses of DE, in this paper, we present a hybrid Differential Evolution (DE) algorithm, referred to as FCDE, which is based on the fuzzy C-means clustering algorithm. The

Table 1: Best error values of FCDE and DE on all test functions with $D = 30$, where "Mean" indicates the mean best error values found in the last generation, "Std Dev" stands for the standard deviation.

F	D	Max_NFFEs	FCDE			DE			FCDE-DE
			Mean	Std Dev	SR	Mean	Std Dev	SR	*t-test*
f01	30	150 000	1.84E-29	1.84E-29	50	1.24E-12	6.65E-13	50	-13.14†
f02	30	200 000	0.00E+00	0.00E+00	50	3.76E-09	1.76E-09	50	-15.14†
f03	30	500 000	6.36E-19	1.52E-18	50	8.04E-10	8.86E-10	50	-6.42†
f04	30	500 000	3.32E+00	1.42E+00	0	4.70E-01	1.09E+00	6	11.26†
f05	30	500 000	3.43E-11	2.04E-10	50	9.58E-09	1.89E-08	39	-3.58†
f06	30	150 000	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50	0
f07	30	300 000	1.57E-03	6.22E-04	50	5.13E-03	1.39E-03	50	-16.49†
f08	30	300 000	9.40E+02	3.93E+02	0	6.64E+03	5.47E+02	0	-59.83†
f09	30	300 000	1.30E+01	4.64E+00	0	1.50E+02	2.12E+01	0	-44.71†
f10	30	150 000	3.08E-15	1.64E-15	50	4.11E-07	1.54E-07	0	-18.85†
f11	30	200 000	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50	0
f12	30	150 000	2.95E-31	2.51E-31	50	1.58E-13	1.46E-13	50	-7.66†
f13	30	150 000	2.27E-24	1.51E-23	50	2.86E-11	2.81E-11	50	-7.18†

† The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by two-tailed test.

fuzzy C-means clustering algorithm acts as several multi-parent crossover operators to utilize the information of the population efficiently. In addition, the population-based algorithm-generator proposed in [5] is adopted to efficiently update the population with the clustering offspring. To validate the performance of our approach, 13 high-dimensional benchmark functions of a wide range of diversity complexities are employed. Experimental results indicate that our approach is effective and efficient. Compared with other state-of-the-art DE approaches, our approach performs better, or at least comparably, in terms of the quality of the final solutions and the reduction of the number of fitness function evaluations (NFFEs).

The rest of this paper is organized as follows. The DE is briefly introduced in Section 2. Section 3 briefly describes the fuzzy C-means clustering algorithm used in this work. In Section 4, the proposed fuzzy C-means clustering-based DE (FCDE) is described in detail. In Section 5, we verify our approach through 13 benchmark functions, and compare it with those of some state-of-the-art DE approaches. The last section, Section 6, is devoted to conclusions and future work.

2. DIFFERENTIAL EVOLUTION

The DE algorithm [13] is a simple EA 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 the traditional EAs. In addition, DE is a simple yet powerful population-based, direct search algorithm with the generation-and-test feature for globally optimizing functions using real-valued parameters. Among DE's advantages are its simple structure, ease of use, speed and robustness. Due to these advantages, it has many real-world applications, such as data mining [1], [4], pattern recognition, digital filter design, neural network training, etc. [10], [2].

The DE algorithm in pseudo-code is shown in Algorithm 1. D is the number of decision variables, NP is the size of the parent population P; F is the mutation scaling factor; CR is the probability of crossover operator; U^i is the offspring; rndint$(1, D)$ is a uniformly distributed random integer number between 1 and D; $X_j^{r_1}$ is the j-th variable of solution X^{r_1}; and $\mathrm{rnd}_j[0, 1)$ is a uniformly distributed random real number in $[0, 1)$. Many schemes of creation of a candidate are possible. We use the DE/rand/1/bin scheme (see lines 6 - 13 of Algorithm 1) described in Algorithm 1 (more details on DE/rand/1/bin and other DE schemes can be found in [14] and [10]).

Algorithm 1 DE algorithm with DE/rand/1/bin

1: Generate the initial population P
2: Evaluate the fitness for each individual in P
3: **while** The halting criterion is not satisfied **do**
4:   **for** $i = 1$ to NP **do**
5:     Select uniform randomly $r_1 \neq r_2 \neq r_3 \neq i$
6:     j_{rand} = rndint$(1, D)$
7:     **for** $j = 1$ to D **do**
8:       **if** $\mathrm{rnd}_j[0, 1) > CR$ or $j == j_{rand}$ **then**
9:         $U_j^i = X_j^{r_1} + F \times (X_j^{r_2} - X_j^{r_3})$
10:       **else**
11:         $U_j^i = X_j^i$
12:       **end if**
13:     **end for**
14:     Evaluate the offspring U^i
15:     **if** U^i is better than P^i **then**
16:       $P^i = U^i$
17:     **end if**
18:   **end for**
19: **end while**

From Algorithm 1, we can see that there are only three control parameters in this algorithm. These are NP, F and CR. As for the terminal conditions, one can either fix the maximum NFFEs *Max_NFFEs* or the precision of a desired solution *VTR* (value to reach).

3. FUZZY C-MEANS CLUSTERING ALGORITHM

Clustering algorithms proposed in literature can be divided into two main categories: crisp (or hard) clustering procedures where

Table 2: NFFEs Required to obtain accuracy levels less than VTR. "NA" indicates the accuracy level is not obtained after the Max_NFFEs.

F	D	Max_NFFEs	FCDE			DE			FCDE-DE
			Mean	Std Dev	SR	Mean	Std Dev	SR	*t-test*
f01	30	150 000	5.52E+04	1.67E+03	50	1.14E+05	1.98E+03	50	-161.01†
f02	30	200 000	8.46E+04	1.48E+03	50	1.91E+05	3.15E+03	50	-216.67†
f03	30	500 000	2.50E+05	1.58E+04	50	4.53E+05	1.70E+04	50	-61.57†
f04	30	500 000	NA	NA	0	3.70E+05	7.70E+03	6	NA
f05	30	500 000	4.19E+05	4.40E+04	50	4.80E+05	1.18E+04	39	NA
f06	30	150 000	1.88E+04	8.58E+02	50	4.25E+04	1.33E+03	50	-105.86†
f07	30	300 000	4.32E+04	1.67E+04	50	1.48E+05	3.22E+04	50	-20.48†
f08	30	300 000	NA	NA	0	NA	NA	0	NA
f09	30	300 000	NA	NA	0	NA	NA	0	NA
f10	30	150 000	8.66E+04	1.70E+03	50	NA	NA	0	NA
f11	30	200 000	5.71E+04	1.33E+03	50	1.20E+05	3.89E+03	50	-107.75†
f12	30	150 000	4.62E+04	1.84E+03	50	1.05E+05	3.37E+03	50	-107.77†
f13	30	150 000	5.88E+04	7.22E+03	50	1.25E+05	3.31E+03	50	-59.02†

† The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by two-tailed test.

each data point belongs to only one cluster, and fuzzy clustering techniques where every data point belongs to every cluster with a specific degree of membership [8].

Fuzzy C-means (FCM) clustering algorithm [8] is based on a fuzzy extension of the least-square error criterion. The advantage of FCM over K-means is that FCM assigns each pattern to each cluster with some degree of membership (i.e. fuzzy clustering). This is more suitable for real applications where there are some overlaps between the clusters in the data set. The objective function that the FCM optimizes is

$$J_m = \sum_{i=1}^{N} \sum_{j}^{C} u_{ij}^m \parallel \boldsymbol{x}_i - \boldsymbol{c}_j \parallel^2 \quad (3)$$

where N is the number of patterns; C is the number of cluster centers; $\boldsymbol{x}_i$ is the i-th pattern; $\boldsymbol{c}_j$ is the j-th center; m is the fuzziness exponent, with $m \geq 1$ ($m = 2$ used in this workd); and $\parallel \bullet \parallel$ is any norm expressing the similarity between any measured data and the center. Increasing the value of m will make the algorithm more fuzzy; $u_{i,j}$ is the membership value for the i-th pattern in the j-th cluster.

Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above, with the update of membership u_{ij} and the cluster centers $\boldsymbol{c}_j$ by:

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\|\boldsymbol{x}_i - \boldsymbol{c}_j\|}{\|\boldsymbol{x}_i - \boldsymbol{c}_k\|} \right)^{\frac{2}{m-1}}} \quad (4)$$

$$\boldsymbol{c}_j = \frac{\sum_{i=1}^{N} (u_{i,j} \cdot \boldsymbol{x}_i)}{\sum_{i=1}^{N} u_{i,j}} \quad (5)$$

4. FUZZY C-MEANS CLUSTERING BASED-DE: FCDE

As mentioned above, the DE algorithm is good at exploring the search space and locating the region of global minimum, however, it is slow at exploitation of the solution [9]. In order to accelerate the convergence rate and balance the exploration and exploitation of the original DE, in this work, we attempt to improve DE by integrating the one-step fuzzy C-means clustering algorithm. Our proposed DE algorithm is referred to as FCDE. The pseudocode of FCDE is described in Algorithm 2, where t is the generation counter, *cp* is the *clustering period*, *NP* is the population size, and $\text{rndint}[2, \sqrt{NP}]$ is a random integer number from $[2, \sqrt{NP}]$. Compared with the original DE, three crucial issues of FCDE will be discussed as follows.

Algorithm 2 Fuzzy C-means Clustering-based DE: FCDE

```
Generate the initial population P randomly
Evaluate the fitness for each individual in P
Initialize the generation counter t = 1
while The halting criterion is not satisfied do
    Use DE to update the population (see lines 4 - 18 in Algorithm 1)
    if t%cp == 0 then
        Randomly generate C = rndint[2, √NP]
        Adopt the one-step fuzzy C-means clustering to create C offspring (the set A)
        Choose C parents (the set B) randomly from the population P
        From the combined set A ∪ B, choose C best solutions and put them in B'. Update P as P = (P\B) ∪ B'
    end if
    t = t + 1
end while
```

4.1 One-step Fuzzy C-Means Clustering

In this study, one-step fuzzy C-means clustering is used to enhance the performance of DE. It acts as several multi-parent crossover operators to utilize the information of the population efficiently, and hence it can balance the exploration and exploitation in the evolutionary process. The one-step fuzzy C-means clustering is described as follows.

1) Choose C individuals as the initial cluster centers $\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_C$ randomly from the current population $\{X_1, X_2, \cdots, X_{NP}\}$.

2) Calculate the $U = [u_{ij}]$ matrix with Eqn. (4).

Table 3: Comparison of the best error values of DE, FCDE, DEahcSPX, and ODE on all test functions with $D = 30$.

F	FCDE	DE	DEahcSPX	ODE
f01	1.84E-29 ± 1.84E-29 (50)	1.24E-12 ± 6.65E-13 (50)†	9.93E-14 ± 9.21E-14 (50)†	4.53E-26 ± 9.27E-26 (50)†
f02	0.00E+00 ± 0.00E+00 (50)	3.76E-09 ± 1.76E-09 (50)†	6.01E-10 ± 2.74E-10 (50)†	8.16E-11 ± 7.14E-11 (50)†
f03	6.36E-19 ± 1.52E-18 (50)	8.04E-10 ± 8.86E-10 (50)†	7.44E-11 ± 1.05E-10 (50)†	2.68E-10 ± 3.10E-10 (50)†
f04	3.32E+00 ± 1.42E+00 (0)	4.70E-01 ± 1.09E+00 (6)‡	5.43E-01 ± 1.08E+00 (2)‡	3.44E-15 ± 1.01E-14 (50)‡
f05	3.43E-11 ± 2.04E-10 (50)	9.58E-09 ± 1.89E-08 (39)†	5.88E-08 ± 3.64E-07 (46)†	2.57E+01 ± 8.19E-01 (0)†
f06	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)
f07	1.57E-03 ± 6.22E-04 (50)	5.13E-03 ± 1.39E-03 (50)†	3.84E-03 ± 1.10E-03 (50)†	9.69E-04 ± 3.04E-04 (50)‡
f08	9.40E+02 ± 3.93E+02 (0)	6.64E+03 ± 5.47E+02 (0)†	5.92E+03 ± 8.31E+02 (0)†	6.79E+03 ± 3.72E+02 (0)†
f09	1.30E+01 ± 4.64E+00 (0)	1.50E+02 ± 2.12E+01 (0)†	1.15E+02 ± 2.13E+01 (0)†	5.60E+01 ± 1.84E+01 (0)†
f10	3.08E-15 ± 1.64E-15 (50)	4.11E-07 ± 1.54E-07 (0)†	8.52E-08 ± 3.24E-08 (0)†	1.26E-13 ± 1.27E-13 (50)†
f11	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)
f12	2.95E-31 ± 2.51E-31 (50)	1.58E-13 ± 1.46E-13 (50)†	8.36E-15 ± 9.15E-15 (50)†	3.00E-27 ± 5.66E-27 (50)†
f13	2.27E-24 ± 1.51E-23 (50)	2.86E-11 ± 2.81E-11 (50)†	1.14E-12 ± 1.11E-12 (50)†	4.35E-23 ± 1.36E-22 (50)†

$^{\dagger, \ddagger}$ The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by two-tailed test.
‡ means that the corresponding algorithm is better than our proposed FCDE method.

3) Compute new cluster centers $\boldsymbol{c}'_1, \cdots, \boldsymbol{c}'_C$ using Eqn. (5).

4) Replace each $\boldsymbol{c}_i$ with $\boldsymbol{c}'_i, i = 1, 2, \cdots, C$, and evaluate these C individuals. The process is terminated.

We choose the one-step fuzzy C-means clustering for its simplicity. Other clustering approaches can also be employed as well. Note that C is generated from $[2, \sqrt{NP}]$ randomly. Here, the upper bound of the number of clusters is taken to be $\sqrt{NP}$, which is a rule of thumb used by many investigators in the literature [12]. In addition, the Euclidean distance is used as the distance measure.

4.2 Population Update

After using one-step fuzzy C-means clustering to create C offspring, the population needs to be updated. Deb [5] proposed a generic population-based algorithm-generator for real-parameter optimization, where the optimization task is divided into four independent plans: i) selection plan, ii) generation plan, iii) replacement plan, and iv) update plan. In lines 8 - 10 of Algorithm 2, our improvement can also be described with the population-update-algorithm proposed in [5].

- **Selection plan**: Choose C individuals from current population randomly as the C initial cluster centers.
- **Generation plan**: Create C offspring (the set $\boldsymbol{A}$) using the one-step fuzzy C-means clustering.
- **Replacement plan**: Choose C solutions (the set $\boldsymbol{B}$) from current population randomly for replacement.
- **Update plan**: From the combined set $\boldsymbol{A} \cup \boldsymbol{B}$, choose C best solutions and put them in $\boldsymbol{B}'$. Update P as $P = (P \backslash \boldsymbol{B}) \cup \boldsymbol{B}'$.

The population-update-algorithm used in this work is similar to the G3 model in [5]. In the update plan, the C best solutions are chosen from the combined set $\boldsymbol{A} \cup \boldsymbol{B}$, thereby the elite-preservation is ensured.

4.3 Clustering Period

In order to exploit the search space efficiently, the clustering is performed periodically in our proposed hybrid DE. It is similar to the method used in [3]. The reason for performing the clustering periodically is that DE needs time to explore the search place and form clusters. An attempt to perform the clustering very early will lead to a false identification of clusters [3]. Consequently, it is important to choose a clustering period that is large enough so that DE has time to completely form stable clusters.

It is worth to point out that the clustering period used in FCDE approach is similar to Damavandi's technique proposed in [3]. Compared with Damavandi's technique, our approach has two main differences: i) We don't use the deterministic method to refine the cluster centers; and ii) We propose a population update method to update the population after the clustering technique is conducted.

5. EXPERIMENTAL RESULTS AND ANALYSIS

13 high-dimensional benchmark functions ($D = 30$) from [16] were chosen to test the performance of our proposed FCDE algorithm. Functions f01 - f05 are unimodal. Function f06 is the step function, which has one minimum and is discontinuous. Function f07 is a noisy quartic function, where *random* [0,1) is a uniformly distributed random variable in [0,1). Functions f08 - f13 are multimodal functions where the number of local minima increases exponentially with the problem dimension. They appear to be the most difficult class of problems for many optimization algorithms. Note that, in the experiment, all functions have a small modification, i.e., to make a linear shift as follows

$$Y = X - \exp(1) \tag{6}$$

The new problem is to solve

$$\text{argmin} f(Y) \tag{7}$$

rather than $\text{argmin} f(X)$. This modification does not change the difficulty in degree of optimization problems. However, it can avoid the influence of the symmetrical initialization and the centroid calculation of the clustering [6].

5.1 Experimental Setup

For FCDE, there are four control parameters. Three of them belong to the original DE, namely, population size *NP*, scaling factor

Table 4: **Comparison of the required NFFEs to obtain accuracy levels less than VTR for the four algorithms. "NA" indicates the accuracy level is not obtained after the Max_NFFEs.**

F	FCDE	DE	DEahcSPX	ODE
f01	5.52E+04 ± 1.67E+03 (50)	1.14E+05 ± 1.98E+03 (50)†	1.06E+05 ± 2.31E+03 (50)†	6.18E+04 ± 2.02E+03 (50)†
f02	8.46E+04 ± 1.48E+03 (50)	1.91E+05 ± 3.15E+03 (50)†	1.78E+05 ± 3.15E+03 (50)†	1.64E+05 ± 4.49E+03 (50)†
f03	2.50E+05 ± 1.58E+04 (50)	4.53E+05 ± 1.70E+04 (50)†	4.18E+05 ± 1.61E+04 (50)†	4.36E+05 ± 1.93E+04 (50)†
f04	NA ± NA (0)	3.70E+05 ± 7.70E+03 (6)	3.35E+05 ± 8.36E+03 (2)	7.15E+04 ± 2.77E+03 (50)
f05	4.19E+05 ± 4.40E+04 (50)	4.80E+05 ± 1.18E+04 (39)	4.73E+05 ± 1.69E+04 (46)	NA ± NA (0)
f06	1.88E+04 ± 8.58E+02 (50)	4.25E+04 ± 1.33E+03 (50)†	3.89E+04 ± 2.10E+03 (50)†	2.43E+04 ± 1.81E+03 (50)†
f07	4.32E+04 ± 1.67E+04 (50)	1.48E+05 ± 3.22E+04 (50)†	1.22E+05 ± 2.92E+04 (50)†	3.32E+04 ± 1.12E+04 (50)‡
f08	NA ± NA (0)	NA ± NA (0)	NA ± NA (0)	NA ± NA (0)
f09	NA ± NA (0)	NA ± NA (0)	NA ± NA (0)	NA ± NA (0)
f10	8.66E+04 ± 1.70E+03 (50)	NA ± NA (0)	NA ± NA (0)	9.99E+04 ± 3.31E+03 (50)†
f11	5.71E+04 ± 1.33E+03 (50)	1.20E+05 ± 3.89E+03 (50)†	1.10E+05 ± 2.52E+03 (50)†	7.08E+04 ± 7.12E+03 (50)†
f12	4.62E+04 ± 1.84E+03 (50)	1.05E+05 ± 3.37E+03 (50)†	9.71E+04 ± 2.89E+03 (50)†	5.63E+04 ± 2.22E+03 (50)†
f13	5.88E+04 ± 7.22E+03 (50)	1.25E+05 ± 3.31E+03 (50)†	1.15E+05 ± 3.22E+03 (50)†	7.27E+04 ± 3.48E+03 (50)†

†,‡ The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by two-tailed test.
‡ means that the corresponding algorithm is better than our proposed FCDE method.

F, and crossover probability *CR*. These parameters are problem dependent [7]. Another parameter is the clustering period cp, which will be discussed later. For all experiments, we use the following parameters unless a change is mentioned.

- Population size: $NP = 100$;
- Scaling factor: $F = 0.5$;
- Crossover probability: $CR = 0.9$;
- Clustering period: $cp = 10$;
- DE scheme: DE/rand/1/bin;
- Value to reach: VTR = 10^{-8}, except for f07 of VTR = 10^{-2};
- Maximum NFFEs[1]: For f01, f06, f10, f12, and f13, Max_NFFEs = 150000; for f03 - f05, Max_NFFEs = 500000; for f02 and f11, Max_NFFEs = 200000; For f07 - f09, Max_NFFEs = 300000.

Moreover, in our experiments, each function is optimized over 50 independent runs. We also use the same set of initial random populations to evaluate different algorithms in a similar way done in [9]. All the algorithms are implemented in standard C++.

5.2 Performance Criteria

Four performance criteria are selected from [15] to evaluate the performance of the algorithms. These criteria are also used in [9] and described as follows.

- **Error**: The error of a solution X is defined as $f(X) - f(X^*)$, where X^* is the global optimum of the function. The minimum error is recorded when the Max_NFFEs is reached in 50 runs and the average and standard deviation of the error values are calculated.
- **NFFEs**: The number of fitness function evaluations (NFFEs) is also recorded when the VTR is reached. The average and standard deviation of the NFFEs values are calculated.

[1]The function evaluations required to process the cluster points are added in the Maximum NFFEs.

- **Number of successful runs (SR)**: The number of successful runs is recorded when the VTR is reached before the max_NFFEs condition terminates the trial.
- **Convergence graphs**: The convergence graphs show the mean error performance of the total runs, in the respective experiments.

5.3 Comparison of DE and FCDE

In this experiment, we compare the performance between the original DE and FCDE to show the superiority of FCDE. Table 1 shows the best error values of DE and FCDE on all test functions. The average and standard deviation of NFFEs are shown in Table 2. Additionally, some representative convergence graphs of DE and FCDE are shown in Fig. 1 and Fig. 2.

With respect to the best error values of DE and FCDE, from Table 1, it can be seen that FCDE is significantly better than DE on 10 out of 13 functions. For function f04, FCDE is worse than DE due to the premature of FCDE shown in Fig. 1 (d). The reason might be that the unreasonable setting of cp makes FCDE converge prematurely. For the rest two functions (f06 and f11), both FCDE and DE can obtain the global optimum over all 50 runs.

From Table 2, we can see that FCDE requires less NFFEs to reach the accuracy levels on ten functions. Only for function f04, DE is better than FCDE. For f08 and f09, both FCDE and DE fail to obtain the accuracy levels.

Considering the convergence rate of FCDE and DE, Fig. 1 and Fig. 2 show that FCDE is faster than DE on 11 functions out of the selected 12 functions. For function f04, FCDE is faster in the early evolution process, however, it stagnates after 20000 NFFEs. The reason might be the loss of the diversity of the population.

5.4 Comparison with Other DE Hybrids

In this section, we make a comparison with other DE hybrids. Since there are many variants of DE, we only compare our approach with DEahcSPX proposed in [9] and ODE proposed in [11]. In DEahcSPX, a crossover-based adaptive local search operation to accelerate the original DE. The authors concluded that DEahcSPX outperforms the original DE in items of convergence rate in

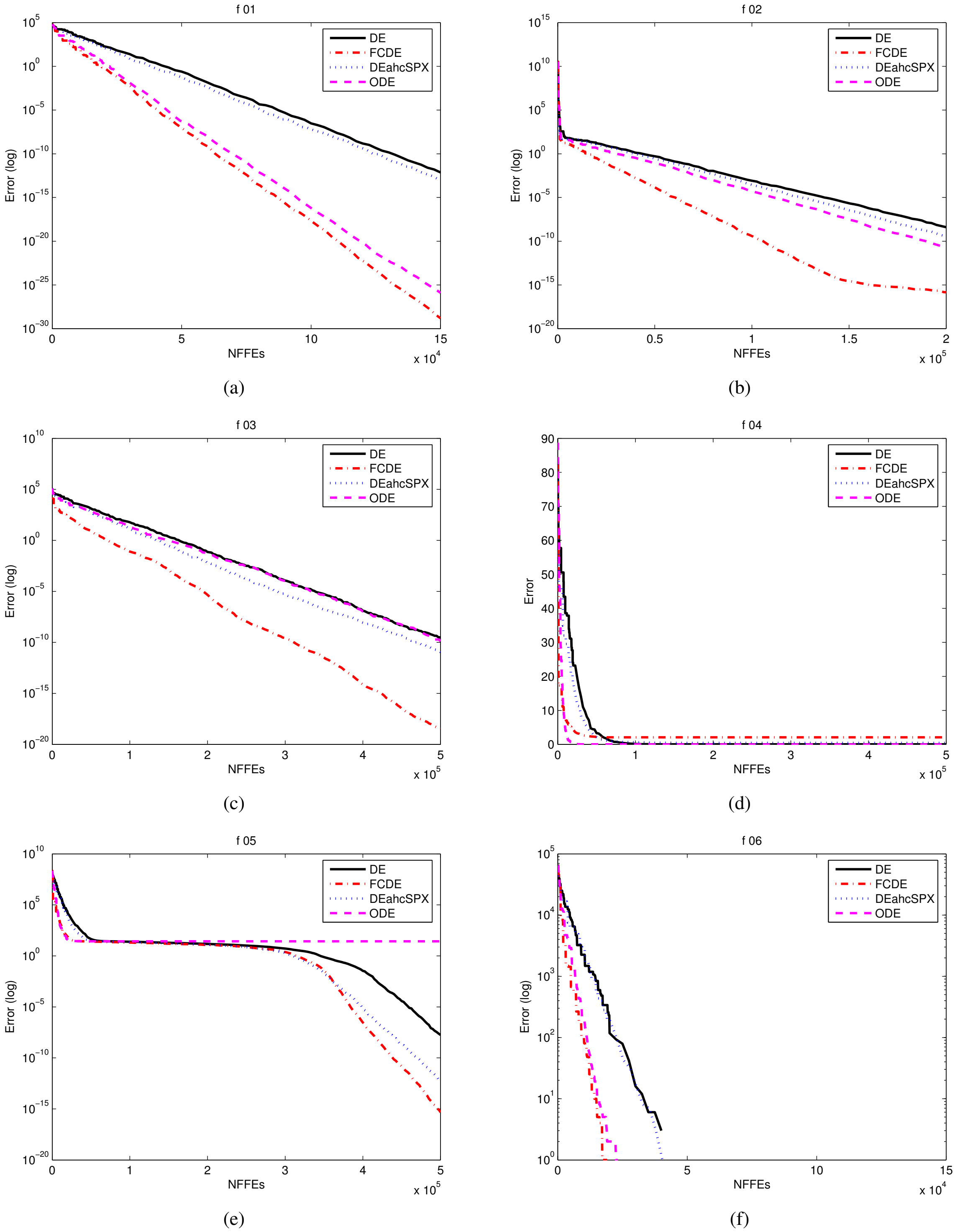

Figure 1: Mean error curves of DE, FCDE, DEahcSPX, and ODE for six unimodal functions. (a) f01. (b) f02. (c) f03. (d) f04. (e) f05. (f) f06. Log-scale of Y axis is to make the comparison more clearly for some functions.

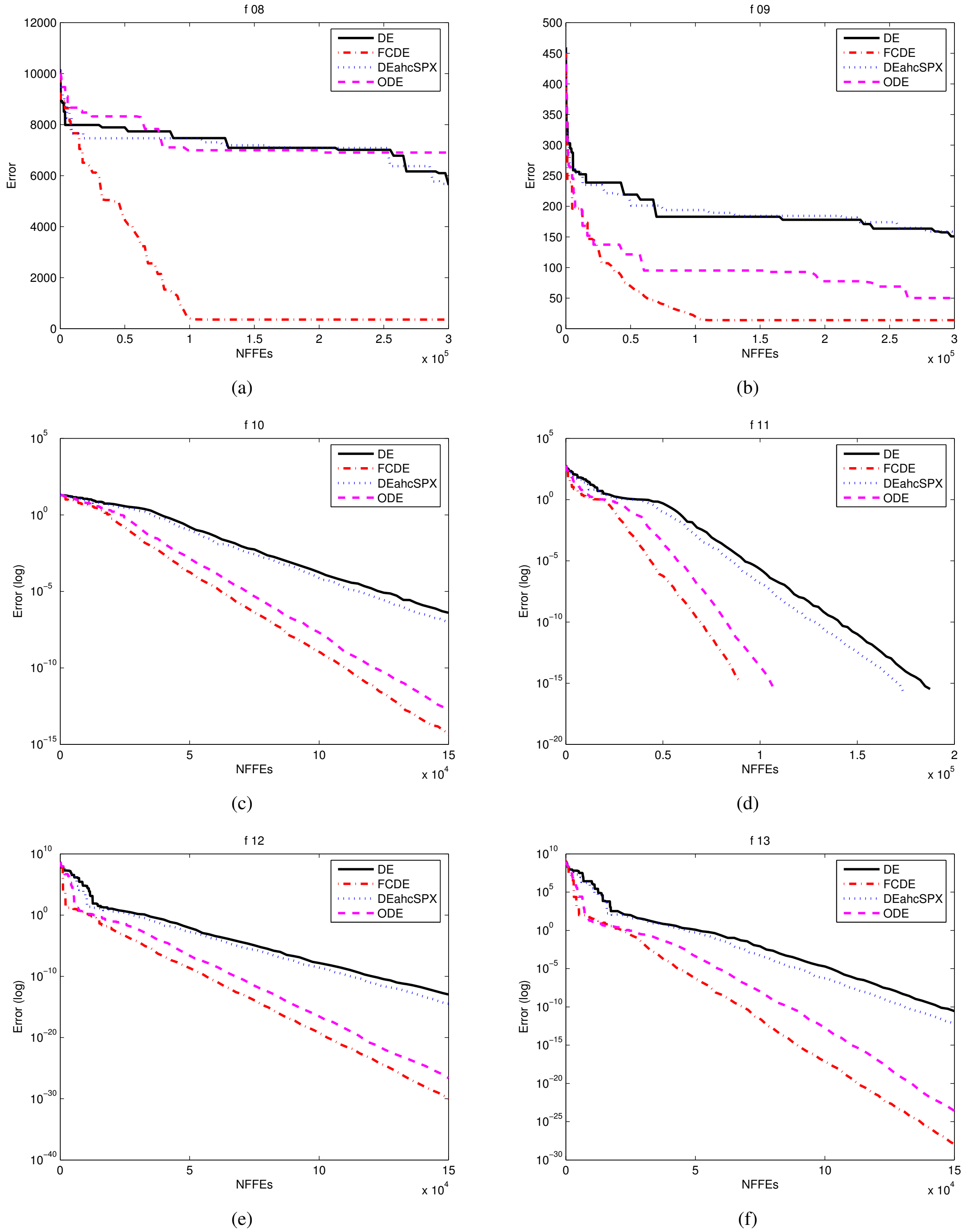

Figure 2: Mean error curves of DE, FCDE, DEahcSPX, and ODE for six multimodal functions. (a) f08. (b) f09. (c) f10. (d) f11. (e) f12. (f) f13. Log-scale of Y axis is to make the comparison more clearly for some functions.

all experimental studies. In ODE, the opposition-based learning is used for the population initialization and generation jumping. In this section, we compare our proposed CDE with the original DE, DEahcSPX and ODE. All the parameter settings are the same as mentioned in Section 5.1. For DEahcSPX, the number of parents in SPX sets to be $n_p = 3$ [9]. For ODE, the jump rate $J_r = 0.3$ [11].

The results are given in Table 3 and 4. Some selected representative convergence graphs are shown in Fig. 1 and 2.

When Compared with DEahcSPX: FCDE is significantly better than DEahcSPX on 10 functions in terms of the best error values. FCDE convergence faster than DEahcSPX on 11 functions shown in Fig. 1 - 2 and Table 4. Only for function f04, DEahcSPX is better than FCDE in terms of all four performance criteria.

When Compared with ODE: Similar conclusions can be obtained from Table 3 - 4 and Fig. 1 - 2, FCDE is better than ODE on the majority test functions (9 out of 13) in terms of the best error values. Moreover, FCDE converges faster than ODE on the majority functions. However, for functions f04 and f07, ODE is significantly better than FCDE for all four criteria.

6. CONCLUSIONS

To make the DE algorithm more effective and more efficient, a hybrid DE algorithm, which combines the fuzzy C-means clustering algorithm, is proposed in this paper. The proposed FCDE method can balance the exploration and the exploitation in the evolutionary process. Furthermore, the population-based algorithm-generator is adopted to efficiently update the population with the clustering offspring. To evaluate the performance of our approach, 13 high-dimensional benchmark functions and four performance criteria are selected from the literature. Experimental results indicate that FCDE is effective and efficient, it can obtain the optimal, or near-optimal, solutions for all test functions. Compared with the original DE, DEahcSPX, and ODE, FCDE performs better, or at least comparably, in terms of the quality of the final solutions and the reduction of the NFFEs.

One additional parameter, clustering period cp, is included in FCDE. In our future work, the effect will be studied in more detail by varying the population size and the problem dimensionality. In addition, another possible direction is applying the one-step fuzzy C-means method to other EC algorithms, such as GAs, PSO, etc.

Acknowledgments

The authors would like to thank the anonymous reviewers for their advice. This work was partly supported by the Fund for Outstanding Doctoral Dissertation of CUG, China Scholarship Council under Grant No. 2008641008, and the National High Technology Research and Development Program of China under Grand No. 2009AA12Z117.

7. REFERENCES

[1] B. Alatas, E. Akin, and A. Karci. Modenar: Multi-objective differential evolution algorithm for mining numeric association rules. *Applied Soft Computing*, 8(1):646–656, January 2008.

[2] U.K. Chakraborty. *Advances in Differential Evolution*. Springer-Verlag, Berlin, 2008.

[3] N. Damavandi and S. Safavi-Naeini. A hybrid evolutionary programming method for circuit optimization. *IEEE Transaction on Circuits ans Systems-I*, 52(5):902–910, May 2005.

[4] S. Das, A. Abraham, and A. Konar. Automatic clustering using an improved differential evolution algorithm. *IEEE Transaction on Systems Man and Cybernetics: Part A*, 38(1):218–237, February 2008.

[5] K. Deb. A population-based algorithm-generator for real-parameter optimization. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 9(4):236–253, 2005.

[6] D. B. Fogel and H.-G. Beyer. A Note on the Empirical Evaluation of Intermediate Recombination. *Evolutionary Computation*, 3(4):491–495, 1996.

[7] R. Gäperle, S.D. Müler, and P. Koumoutsakos. A parameter study for differential evolution. In *Proc. WSEAS Int. Conf. Advances Intell. Syst., Fuzzy Syst., Evol. Comput.*, pages 293–298, 2002.

[8] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[9] N. Noman and H. Iba. Accelerating differential evolution using an adaptive local search. *IEEE Transactions on Evolutionary Computation*, 12(1):107–125, February 2008.

[10] K. Price, R. Storn, and J. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag, Berlin, 2005.

[11] S. Rahnamayan, H.R. Tizhoosh, and M.M.A. Salama. Opposition-based differential evolution. *IEEE Transactions on Evolutionary Computation*, 12(1):64–79, February 2008.

[12] W. Sheng, S. Swift, L. Zhang, and X. Liu. A weighted sum validity function for clustering with a hybrid niching genetic algorithm. *IEEE Transaction on Systems Man and Cybernetics: Part B*, 35(6):1156–1167, December 2005.

[13] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, December 1997.

[14] R. Storn and K. Price. Home page of differential evolution, http://www.ICSI.Berkeley.edu/˜storn/code.html. 2003.

[15] P.N. Suganthan, N. Hansen, and J.J. Liang. Problem definitions and evaluation criteria for the cec2005 special session on real-parameter optimization, http://www.ntu.edu.sg/home/EPNSugan. 2005.

[16] Xin Yao, Yong Liu, and Guangming Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82–102, July 1999.