International Journal of Computational Intelligence Systems, Vol. 6, No. 6 (November, 2013), 1094-1107

An Adaptive Differential Evolution Algorithm Based on New Diversity

Huan Lian [1] [*], Yong Qin [2] , Jing Liu [3]

[1] *College of Mathematics Science, Tianjin Normal University, Tianjin, 300387, P.R.China*

E-mail: lianhuan369@163.com

[2] *State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiao Tong University, Beijing, 100044, P.R. China*

E-mail: qinyong2146@126.com

[3]*School of Mathematics, Beijing Institute of Technology, Beijing, 100081, P.R.China*

Received 4 September 2012

Accepted 7 April 2013

Abstract

A DE approach based on a new measure of population diversity and a novel parameter control mechanism is proposed with the aim of introducing a good behavior of the algorithm. The ratio of the new defined population diversity of different generations is equal to that of the population variance, therefore the adaption of parameter can use some theoretical results in[19]. Combining with the method in[18], we can adjust the mutation factor *F* and the crossover rate *CR* at each generation in the searching process. The performance of the proposed algorithm (DE-F&CR) is compared to the basic DE and other four DE algorithms over 25 standard numerical benchmarks provided by the IEEE Congress on Evolutionary Computation 2005 special session on real parameter optimization. The results and its statistical analysis show that the DE-F&CR generally outperforms the other algorithms in multi-modal optimization.

Keywords: Intelligent algorithm, Differential evolution, Population diversity, Adaptive parameter control

1. Introduction

The Differential Evolution algorithm (DE) introduced by Storn and Price[1] is a stochastic population-based search method. As one of the best evolutionary algorithms (EAs)[2], it has proven to be a promising candidate to solve real value global optimization. The DE algorithm also presents simple structure, convergence speed, versatility and robustness, with few parameters to be specified. There are two main processes in differential evolution during the search for optimum: the variation process which ensures the exploration of the search space and the selection process which ensures the exploitation abilities of the algorithm. If the exploitation process is dominant with respect to the exploration, the population loses its diversity and the algorithm is stuck into a non-optimal state (premature convergence). On the other hand when the exploration is dominant, the algorithm does not approach the global optimum in a reasonable number of generations (slow convergence). Therefore a balance between exploration

[*]Corresponding author (H. Lian)

and exploitation is crucial to realize rapid convergence and still avoid premature convergence.

Population diversity can be a way to monitor an algorithm' state of exploration and exploitation, it is important for helping adjust the ability of exploration and exploitation. Diversity can reveal internal characteristic of a search process. Again, the exploration power of the algorithm depends on the population diversity. It is commonly accepted that maintaining proper population diversity is important to avoid premature convergence and escape the local optima. Diversity measures have been used to control evolutionary algorithms in several studies. The Diversity-Control-Oriented Genetic Algorithm uses a diversity measure based on Hamming distance to calculate a survival probability for the individuals[3]. A low Hamming distance between the individual and the current best individual is translated into a low survival probability to preserve population diversity through the selection procedure. Ursem[4] proposed the Diversity-Guided Evolutionary Algorithm which applies diversity-decreasing operators (selection, recombination) and diversity-increasing operator (mutation) to alternate between two modes based on a distance-to-average-point measure. Riget and Vesterstorm[5] suggested the attractive and repulsive particle swarm optimization (PSO) which used a diversity measure to control the swarm alternating between the phases of attractive and repulsive. A parameter adaption of DE (ADE) based on controlling the population variance and multi-population approach was proposed by Zaharie[6]. Coelho[7] introduced the use of population diversity in order to guide the attractive or repulsive behavior of DE algorithms. Coelho, Ursem and Riget all use the distance-to-average-point as diversity measure to guide the population in the process of searching for the optimum, and the results show the algorithms mentioned above are competitive and powerful.

On the other hand, an essential role in influencing the convergence behavior of the DE algorithm is played by its strategy parameters such as the population size M, the mutation factor F and the crossover rate CR. Many studies have shown that the performance of DE is very sensitive to the choice of strategy parameters [6,8,9,10,11,12]. The strategy parameters in standard DE are being fixed during the optimization process. Later, various adaption mechanisms were proposed to overcome the hand-tuning problems of the DE control parameters. Neri and Tirronen[13]give a survey of recent advances in differential evolution. Teo[14]proposed a DE algorithm with self-adapting populations and showed that DE with self-adaptive populations produced highly competitive results compared to a conventional DE algorithm with static populations. Brest et al.[15] encoded control parameters F and CR into the individual and adapting them by means of evolution to produce a flexible DE algorithm, called jDE. Qin et al.[16] developed a self-adaptive DE algorithm (SaDE) for constrained real-parameter optimization, and the experiments demonstrated that the SaDE algorithm preformed much better than the conventional DE algorithm. In[11] a fuzzy adaptive differential evolution algorithm (FADE) was proposed which used fuzzy logic controllers to adapt the search parameters for the mutation operation and crossover operation. Ghosh et al.[17] described a simple and effective adaption technique for tuning both F and CR which was based on the objective function value of individuals in the DE population. In[7] Coelho considered that the mutation factor F is the parameter which most critically influences the performance and robustness. Therefore only the mutation factor F is adjusted by means of linearly decreasing. In[18] an improved cultural differential evolution approach based on the measure of population's diversity (CDEMD) was proposed, which considered the crossover rate CR as the key factor affecting the DE convergence, so only the CR is tuned by using the information of population's diversity. Parameters of DE were adapted by controlling the population diversity in[6,19], the mutation factor F and the crossover rate CR are adapted asynchronously basing on some theoretical results on the evolution of the population variance. For instance, at odd generations parameters F are adjusted, while at even generations are adjusted the parameters CR. It is important to automatically adapt two parameters the crossover rate CR and the mutation factor F in DE at each generation.

In this paper, a DE approach based on new measure of population diversity and two parameters adapting at each generation is proposed with the aim of introducing a good behavior of the algorithm. The ratio of the new defined population diversity of different generations is equal to that of the population variance, therefore the adaption of parameter can use some theoretical results in[19]. Combining with the method in[18], we can adjust the crossover rate CR and the mutation factor F in DE at each generation.

The remaind of the paper is organized as follows. In Section 2, we introduce the basic differential evolution algorithm. In Section 3, the proposed differential evolution algorithm (DE-F&CR) is analyzed in detail. The experimental setup, parameter settings, the experiment results and statistical comparisons are shown in Section 4. Finally, our conclusions are provided in Section 5.

2. Differential Evolution Algorithm

As with any other evolutionary algorithm, DE utilizes a population of NP D-dimensional parameter vectors $x_{i,j}(G)$, $i = 1, 2, \cdots, NP$; $j = 1, 2, \cdots, D$; $G = 0, 1, 2, \cdots, G_{max}$, where NP denotes the population size and does not change during the evolution process, G denotes subsequent generations in DE. The general form of DE algorithm:

- Initialize a population of NP individuals: the initial population should cover the entire search space as much as possible. In this step, set $G = 0$ and each individual $x_{i,j}(G)$, $i = 1, \cdots, NP$, $j = 1, \cdots, D$ is a solution vector generated with random values according to a uniform probability distribution in the D-dimensional problem space.
- For each individual $x_i(G) = \{x_{i,1}(G), x_{i,2}(G), \cdots, x_{i,D}(G)\}$, $i = 1, \cdots, NP$, evaluate its fitness based on the objection function value.
- Mutation operation: for each target vector $x_i(G) = \{x_{i,1}(G), x_{i,2}(G), \cdots, x_{i,D}(G)\}$, $i = 1, \cdots, NP$, a mutation vector is generated according to following equation:

$$v_i(G) = x_{r_1}(G) + F(x_{r_2}(G) - x_{r_3}(G)), \quad (1)$$

where $v_i(G) = \{v_{i,1}(G), v_{i,2}(G), \cdots, v_{i,D}(G)\}$ stands for the mutant vector corresponding to the i-target vector, r_1, r_2 and r_3 are mutually different integers and also different from the running index i, randomly selected with uniform distribution from the set $\{1, 2, \cdots, i-1, i+1, \cdots, NP\}$ and $F > 0$ is a real parameter, called mutation factor, which controls the amplification of the different between two individuals and is usually taken in the range $[0.3, 2]$.
- Crossover operation: in order to increase the diversity of the perturbed parameter vectors, crossover is applied in the population. The target vector is mixed with the mutated vector using the following scheme to yield the trial vector $u_i(G) = \{u_{i,1}(G), u_{i,2}(G), \cdots, u_{i,D}(G)\}$, where

$$u_{i,j}(G) = \begin{cases} v_{i,j}(G), & \text{if } (r_j \leqslant CR) \text{ or } (j = r_i); \\ x_{i,j}(G), & \text{otherwise,} \end{cases} \quad (2)$$

for $j = 1, \cdots, D$, $r_j \in [0, 1]$ is the j-th evaluation of a uniform random number generator, $r_i \in \{1, 2, \cdots, D\}$ is randomly chosen index which ensures that $u_i(G)$ gets at least one element from $v_i(G)$, and $CR \in [0, 1]$ is a recombination or crossover rate.
- Selection operation: to decide whether or not it should become a member of generation $G+1$, the trial vector $u_i(G)$ is compared to the target vector $x_i(G)$ using the greedy criterion. The selection operation is described as

$$x_i(G+1) = \begin{cases} u_i(G), & f(u_i(G)) < f(x_i(G)); \\ x_i(G), & \text{otherwise,} \end{cases} \quad (3)$$

if the objective function f is to be minimized. So if the new trial vector yields an equal or lower value of the objective function, it replaces the corresponding target vector in the next generation; otherwise the target is retained in the population. Hence the population either gets better (with respect to the minimization of the objective function) or remains the same in fitness status, but never deteriorates.

Several variants of DE were proposed in[1] which are classified using the following notation: $DE/x/y/z$, where x specifies the vector to be mutated which currently can be rand (a randomly chosen population vector) or best (the best vector of the current generation), y is the number of difference vectors used, z denotes the crossover scheme. Studies in[20] have shown that the version $DE/best/2/z$ appears to perform better in most cases.

3. Proposed Differential Evolution Algorithm

The DE algorithm we considered in this paper is the most general case:

$$z_i = \lambda x_{best} + (1-\lambda)x_{r_1} + F[(x_{r_2} - x_{r_3}) + \delta(x_{r_4} - x_{r_5})], \tag{4}$$

where r_1, r_2, r_3, r_4, r_5 are five mutually different random integers in $[1, NP]$, $\lambda \in [0,1]$ represents the coefficient of the convex combination between the best individual of the population and a randomly selected individual, $\delta = 0$ or 1, $F > 0$ is a real parameter and called mutation factor, which controls the amplification of the different between two individuals and is usually taken in the range $[0.3, 2]$. Obviously, it consists most frequently used variants of DE such as $DE/rand/1/bin$ for $\lambda = 0$ and $\delta = 0$; $DE/best/1/bin$ for $\lambda = 1$ and $\delta = 0$; $DE/best/2/bin$ for $\lambda = 1$ and $\delta = 1$.

3.1. DE using the diversity measure

The"distance-to-midpoint" measure diversity is defined as follows:

$$diversity = \frac{1}{N(x_j^{max} - x_j^{min})} \sum_{i=1}^{N} \sqrt{\sum_{j=1}^{n} (x_{i,j} - \bar{x}_j)^2}, \tag{5}$$

where x_j^{max}, x_j^{min} are maximum and minimum values of the i-th dimension, N is the population size, n is the dimensionality of the problem, $x_{i,j}$ is the j-th value of the i-th individual, $\bar{x}_j$ is the j-th value of the midpoint $\bar{x}$.

A new dimension-wise diversity measure of the population is defined as:

$$div(X(G)) = \frac{1}{NM(M-1)} \sum_{j=1}^{N} \sum_{i,k=1}^{M} \frac{(x_{i,j}(G) - x_{k,j}(G))^2}{(x_{max,j} - x_{min,j})^2}, \tag{6}$$

where $x_{max,j}$, $x_{min,j}$ are maximum and minimum values of the population's search region in j-th dimension, M is the population size, N is the dimensionality of the problem, $x_{i,j}(G)$ is the j-th value of the i-th individual in G-th generation, and

$$div(X(G), j) = \frac{1}{M(M-1)} \sum_{i,k=1}^{M} \frac{(x_{i,j}(G) - x_{k,j}(G))^2}{(x_{max,j} - x_{min,j})^2}, \tag{7}$$

when j is fixed.

This new measure computed at the component level of the population individual is different from the "distance-to-midpoint" measure that is applied in[4,5,7,21], besides in DE algorithm it has some similar function to the population variance appeared in[6,19]. Zaharie defined the population variance as follows: for a population of scalars $x = \{x_1, \cdots, x_m\}$, $x_l \in R$, $l = 1, 2, \cdots, m$, let $Var(x) = \overline{x^2} - \bar{x}^2$ with $\bar{x} = \sum_{l=1}^{m} x_l/m$. It is easy to prove that the ratio of the new defined population diversity of different generations is equal to that of the population variance.

Theorem 1. *Let $X(G) = \{x_1(G), x_2(G), \cdots, x_{NP}(G)\}$, for each individual $i = 1, \cdots, NP$, $x_i(G) = \{x_{i,1}(G), x_{i,2}(G), \cdots, x_{i,D}(G)\}$. Then*

$$\frac{div(X(G), j)}{div(X(G+1), j)} = \frac{Var(X(G)_j)}{Var(X(G+1)_j)}, \tag{8}$$

where $X(G)_j = \{x_{1,j}(G), x_{2,j}(G), \cdots, x_{NP,j}(G)\}$.
Proof. By the definition in[6],

$$Var(X(G)_j) = \overline{X(G)_j^2} - \overline{X(G)_j}^2, \tag{9}$$

with $\overline{X(G)_j} = \sum_{i=1}^{NP} x_{i,j}(G)/NP$. Hence

$$\begin{aligned} & Var(X(G)_j) \\ = & \sum_{i=1}^{NP} \frac{x_{i,j}(G)^2}{NP} - \left(\sum_{i=1}^{NP} \frac{x_{i,j}(G)}{NP} \right)^2 \end{aligned}$$

$$= \frac{NP\sum_{i=1}^{NP} x_{i,j}(G)^2 - \left(\sum_{i=1}^{NP} x_{i,j}(G)\right)^2}{NP^2}$$

$$= \sum_{i=1}^{NP}\frac{x_{i,j}(G)^2}{NP} - \frac{[\sum_{i=1}^{NP} x_{i,j}(G)^2 + 2\sum_{i,k=1,i\neq k}^{NP} x_{i,j}(G)x_{k,j}(G)]}{NP^2}$$

$$= \frac{\sum_{i,k=1}^{NP}[x_{i,j}(G)-x_{k,j}(G)]^2}{NP^2}.$$

In a similar manner we get

$$Var(X(G+1)_j) = \frac{\sum_{i,k=1}^{NP}[x_{i,j}(G+1)-x_{k,j}(G+1)]^2}{NP^2}.$$

Thus

$$\frac{Var(X(G)_j)}{Var(X(G+1)_j)} = \frac{\sum_{i,k=1}^{NP}[x_{i,j}(G)-x_{k,j}(G)]^2}{\sum_{i,k=1}^{NP}[x_{i,j}(G+1)-x_{k,j}(G+1)]^2},$$

$$\frac{div(X(G),j)}{div(X(G+1),j)}$$

$$= \frac{\frac{1}{M(M-1)}\sum_{i,k=1}^{M}\frac{(x_{i,j}(G)-x_{k,j}(G))^2}{(x_{max,j}-x_{min,j})^2}}{\frac{1}{M(M-1)}\sum_{i,k=1}^{M}\frac{(x_{i,j}(G+1)-x_{k,j}(G+1))^2}{(x_{max,j}-x_{min,j})^2}}$$

$$= \frac{\sum_{i,k=1}^{M}(x_{i,j}(G)-x_{k,j}(G))^2}{\sum_{i,k=1}^{M}(x_{i,j}(G+1)-x_{k,j}(G+1))^2}$$

$$= \frac{Var(X(G)_j)}{Var(X(G+1)_j)}. \ \square$$

3.2. DE with adaptive factors

Theorem 1 works as a bridge that connects the population variance with the "distance-to-midpoint" diversity measure. Moreover, it makes us adapting the crossover rate *CR* and the mutation factor *F* in DE at each generation.

Table 1. The proposed DE algorithm with diversity guided adaptive parameters

```
Step 1: Generation $G = 0$,
    Initialize the population
    $X(0) = \{x_1(0), x_2(0), \cdots, x_{NP}(0)\}$,
    Initialize the parameters $F_{min} = 0.3$, $F_{max} = 2$,
    $F_j = \sqrt{1/NP}$, $CR_j = 0.9$, $j = 1,2,\cdots D$.
Step 2: Compute the population $div(X(G))$.
Step 3: Apply the mutation operation:
    for $i = 1,2,\cdots,NP$, $j = 1,2,\cdots,D$,
    $v_{i,j} = \lambda x_{best,j} + (1-\lambda)x_{r_1,j} +$
    $F_j[(x_{r_2,j} - x_{r_3,j}) + \delta(x_{r_4,j} - x_{r_5,j})]$.
Step 4: Apply the crossover operation:
    for $i = 1,2,\cdots,NP$, $j = 1,2,\cdots,D$,
    $u_{i,j}(G) = \begin{cases} v_{i,j}(G), & \text{if } (r_j \leqslant CR_j) \text{ or } (j = r_i); \\ x_{i,j}(G), & \text{otherwise.} \end{cases}$
Step 5: Apply the selection operation:
    for $i = 1,2,\cdots,NP$,
    $x_i(G+1) = \begin{cases} u_i(G), & f(u_i(G)) < f(x_i(G)); \\ x_i(G), & \text{otherwise.} \end{cases}$
Step 6: Compute the $div(X(G),j)$,
    $div(X(G+1),j)$ and $c_j = \frac{div(X(G),j)}{div(X(G+1),j)}$.
Step 7: Compute the parameters $\{CR_j\}$,
    For $j = 1,2,\cdots,D$,
    $CR_j = \frac{div(X(0),j)}{div(X(1),j)}$
    $CR_j = min(0.9, max(0.2, CR_j))$.
Step 8: Compute the parameters $\{F_j\}$,
    Step 8.1: For $j = 1,2,\cdots,D$,
        Calculate the $c_j = \frac{div(X(0),j)}{div(X(1),j)}$,
        Let $t_j = (1-CR_j)^2/NP + (NP-1)/NP$
    Step 8.2: If $c_j < t_j$
        then $F_j = F_{min}$
        else $F_j = \sqrt{(c_j - t_j)/2CR_j}$
    Step 8.3: If $F_j < F_{min}$
        then $F_j = F_{min}$
        If $F_j > F_{max}$
        then $F_j = F_{max}$.
Step 9: G=G+1, Until a termination condition is met.
```

Since the diversity is computed at component level, based on which the parameters *F* and *CR* are adapted at each generation, the basic DE algorithm mentioned above will be modified as follows. The parameters *F* and *CR* are replaced with the sets of parameters $\{F_j\}$ and $\{CR_j\}$, for $j = 1,2,\cdots,D$ at each generation are computed the diversity for all individuals and the parameters are adapted by applying the methods above after the selection step. The proposed DE algorithm with diversity guided behavior and adaptive parameters is present in Table 1.

4. Experiments

4.1. Benchmark functions

To evaluate the performance of the proposed DE-F&CR algorithm, experiments are conducted on a set of standard benchmark functions from the IEEE CEC 2005 competition and special session on real parameter optimization[22]. These functions differ in terms of various features such as multi-modality, ruggedness, noise in fitness, non-separability, and rotation. The complete definition of those functions which are based on classical Rosenbrock's, Rastrigin's, Schwefel's, Griewank's and Ackley's benchmark functions is available in[22], and it will not be described repeatedly here for sake of space. Functions 1-5 are unimodal and 6-25 are multi-modal. Functions 13 and 14 are expanded functions, and functions 15-25 are hybrid composition functions, the local optimum is shifted to a non zero value and the global optimum are non zero.

4.2. Algorithms compared and experimental setup

The performance of the DE-F&CR algorithm is compared with the standard differential evolution algorithm (DE) and four state-of-the-art DE variants, which are listed in Table 2.

Table 2. Parameter specification for the algorithms used

DE/rand/1/bin[1]	$F = 0.5, CR = 0.9$;
jDE[15]	$\tau_1 = \tau_2 = 0.1, F_l = 0.1, F_u = 0.9$;
CDEMD[18]	$CR_{max} = 0.9, CR_{min} = 0.2, F = 0.5$;
DE-F[19]	$F_{max} = 2, F_{min} = 0.3, CR = 0.5$;
DE-F&CR	$F_{max} = 2, F_{min} = 0.3,$ $CR_{max} = 0.9, CR_{min} = 0.2$;
SaDE[16]	F is randomly generated by a normal distribution $N(0.5, 0.3)$.

In the standard DE algorithm, the trial vector generation strategy and the associated control parameters are kept fixed throughout the entire optimization process. Choosing suitable parameter value is a problem-dependent task and requires previous experience of the user. According to the empirical guideline existed in literature, the standard DE algorithm used in the experiment selects DE/rand/1/bin as mutation strategy and control parameters $F = 0.5$, $CR = 0.9$ as the best choice[1]. However, the most suitable evolution strategy and control parameter values can be different for different problems or even at different stages of the search process in the same problem .

Brest et al.[15] proposed an adaptive DE algorithm, called jDE, which used DE/rand/1/bin as mutation strategy and fixed the population size *NP* during the run. In their algorithm, control parameters *F* and *CR* were encoded into the individual and adapted by introducing two new parameters τ_1 and τ_2. τ_1 and τ_2 represent probabilities to adjusted factors *F* and *CR*, respectively. A set of F_i values were assigned to individual *i* in the population. Then, a random number rand was generated according to uniform distribution on the interval [0,1]. If rand $< \tau_1$, F_i was regenerated by taking a value from [0.1,1] in a random manner, otherwise it was kept unchanged. The control parameter *CR* was adapted in the same way but take a value from [0,1]. It was believed that the better values of these encoded control parameters lead to better individuals which are more likely to survive and propagate these better parameters. Experimental results suggested that jDE performed remarkably better than standard DE algorithm DE/rand/1/bin, FEP and CEP[23], the adaptive LEP and Best L*é*vy[24] , and the FADE algorithm[11].

Based on their preliminary work[25], Qin et al.[16] developed a self-adaptive DE (SaDE) algorithm for Global Numerical Optimization. In the SaDE algorithm, the trial vector generation strategies and their associated control parameter values were gradually self-adapted by learning from their previous experiences in generating promising solutions. They chose DE/ rand/1/bin, DE/rand-to-best/2/bin, DE/rand/2/bin and DE/current-to-rand/1 to construct a strategy candidate pool, for each target vector in the current population, one trial vector generation strategy was selected from the candidate pool according to the probability learned from its success rate in generating improved solutions within a certain number of previous generations. The control parameter F was approximated by a normal distribution with mean value 0.5 and standard deviation 0.3, denoted by $N(0.5, 0.3)$, and parameter CR obeys a normal distribution with mean value CR_m and standard deviation $Std = 0.1$, denoted by $N(CR_m, Std)$, where CR_m was initialized as 0.5. They have compared the performance of SaDE with the conventional DE and jDE[15], SDE[26], ADE[6] algorithms over bound constrained numerical optimization problems, and concluded that SaDE was more effective in obtaining better quality solutions, which are more stable with the relatively smaller standard deviation, and had higher success rates.

The application of diversity measure can be an alternative strategy to improve the convergence and local search in DE algorithm. Coelho et al.[18] proposed a new cultural DE approach (CDEMD), which used information of population's diversity for tuning of the control parameter *CR*. It is believed that the utilization of improvement in CDE based on diversity measure can be useful to escape from local minima. CDEMD was used to solving the economic load dispatch problems of thermal generators, the results with the CDEMD were superior to that the result presented in recent literature. Zaharie[19] proposed an adaptive parameter control in DE algorithms (DE-F), the adaptation strategy is based on theoretical and empirical results concerning the population diversity evolution. Since the parameters influence in an interacting manner the convergence properties of DE, it is difficult to automatically adapt all the parameters. Their first approach is to adapt only one parameter F while the other one CR is fixed.

The adaptation of control parameters in jDE is based the evolution of the individual, and the adaptation of trial vector generation strategies is according to their previous experiences in generating promising solutions. That is, the self-adaption mechanism in two representative self-adaptive DE algorithms, namely SaDE and jDE, do not use any population diversity information. So their evolution strategy and control parameter setting could not be most suitable for different problems or different stages of the search process in same problem. CDEMD only adapted control parameter CR by using the "distance-to-midpoint" measure diversity information, while DE-F adapt only one parameter F and the other one CR are fixed by using the population variance as the population diversity. Based on the relationship between the control parameters, the population diversity evolution and the convergence behavior of DE, two control parameters F and CR ought be adapted according to the population diversity information to match each other at each generation, such that they are in the good convergence region, near the border which separated this region from the premature convergence region. Therefore, by using the relationship between the population diversity defined in this paper and the population variance, two control parameters F and CR in the DE-F&CR algorithm are automatically adapted at each generation of the evolution process, to make the algorithm avoid premature convergence. The adaptation mechanism of control parameters F and CR in the DE-F&CR algorithm is presented in Table 1.

As a rule of thumb, if nothing is about the problem in hand then the population size, NP should be selected from the range $2 \cdot D$ to $40 \cdot D$, where D is dimensionality of the problem to be solved. For an easy problem small population size is sufficient while parameter-dependent, multi-modal functions requires the large population in order to avoid stagnation to a local optimum. The population size for all the algorithms is set as NP=100 which used in literature[12,15,27,28].

4.3. Simulation strategies

Functions 1-25 are tested with dimensions D=10 and 30. The maximum number of function evaluations is set to 10000D. In the experiments, each function was run for 25 independent trial runs. All the simulations were coded in Matlab language and conducted on a machine with a 2.2GHz Intel Core(TM) 2 Duo CPU and 2GB RAM. The best *error values* reached by an algorithm at the end of optimization was recorded. Note that the best *error values* corresponds to the difference between the best of the

Table 3. Mean and ranks of the best *error values* for *D*=10

Function	DE/rand/1/bin ($F=0.5, CR=0.9$)	jDE	SaDE	CDEMD ($F=0.5$)	DE-F ($CR=0.5$)	DE-F& CR
f_1	**0(3.5)**	**0(3.5)**	**0(3.5)**	**0(3.5)**	**0(3.5)**	**0(3.5)**
f_2	2.5575E-21(2)	4.1051E-15(6)	2.8687E-19(4)	7.8287E-19(5)	5.2418E-21(3)	**5.7396E-22(1)**
f_3	5.7921E-10(3)	5.1327E-08(6)	2.5618E-10(2)	8.1715E-09(5)	4.4299E-09(4)	**1.6976E-10(1)**
f_4	4.1667E-09(6)	1.7895E-16(5)	1.9216E-18(3)	1.0868E-17(4)	1.3426E-18(2)	**1.0627E-18(1)**
f_5	1.1983E-03(5)	**1.1208E-04(1)**	7.8028E-04(3)	2.6291E-03(6)	1.0866E-03(4)	1.1302E-04(2)
f_6	4.0337E-08(5)	7.8526E-09(4)	5.4922E-10(3)	2.6325E-07(6)	1.2986E-11(2)	**3.4757E-13(1)**
f_7	3.4727E-01(4)	5.7123E-02(2)	**2.2532E-02(1)**	3.5753E-01(5)	3.7803E-01(6)	2.8974E-01(3)
f_8	2.0329E+01(3)	2.7284E+01(6)	2.5203E+01(5)	2.0303E+01(2)	2.0356E+01(4)	**2.0298E+01(1)**
f_9	1.8098E+01(6)	1.9161E-11(2)	**5.9084E-18(1)**	1.6732E+01(3)	1.7046E+01(4)	1.7956E+01(5)
f_{10}	2.6125E+01(2)	2.7931E+01(4)	2.6504E+01(3)	5.7821E+01(6)	3.3803E+01(5)	**2.3623E+01(1)**
f_{11}	8.6646E+00(5)	8.4196E+00(2)	8.5243E+00(4)	8.4997E+00(3)	8.9906E+00(6)	**8.0611E+00(1)**
f_{12}	8.3654E+01(5)	3.1307E+01(4)	2.6526E+01(3)	2.1068E+04(6)	1.6640E+01(2)	**6.9592E+00(1)**
f_{13}	2.1211E+00(6)	1.9568E+00(4)	1.2060E+00(2)	1.9146E+00(3)	2.1932E+00(5)	**1.2056E+00(1)**
f_{14}	3.6473E+00(4)	3.5893E+00(2)	**3.5882E+00(1)**	3.6542E+00(5)	3.6772E+00(6)	3.6259E+00(3)
f_{15}	4.6788E+02(5)	4.6305E+02(4)	4.5278E+02(3)	4.7904E+02(6)	**4.0E+02(1.5)**	**4.0E+02(1.5)**
f_{16}	1.5267E+02(5)	1.4987E+02(3)	1.4873E+02(2)	1.5723E+02(6)	1.5164E+02(4)	**1.2010E+02(1)**
f_{17}	1.8171E+02(4)	1.7503E+02(2)	1.8017E+02(3)	1.8307E+02(6)	1.8192E+02(5)	**1.7172E+02(1)**
f_{18}	8.2119E+02(6)	8.2101E+02(3)	8.2105E+02(4)	8.2113E+02(5)	8.2091E+02(2)	**8.2011E+02(1)**
f_{19}	8.2141E+02(6)	8.2107E+02(2)	8.2114E+02(4)	8.2110E+02(3)	8.2118E+02(5)	**8.2101E+02(1)**
f_{20}	8.2124E+02(6)	8.2121E+02(5)	8.2112E+02(3)	8.2111E+02(2)	8.2117E+02(4)	**8.2106E+02(1)**
f_{21}	1.0728E+03(6)	**1.0623E+03(1)**	1.0685E+03(5)	1.0657E+03(3)	1.0657E+03(3)	1.0657E+03(3)
f_{22}	5.2998E+02(6)	**5.2681E+02(1)**	5.2686E+02(3)	5.2903E+02(5)	5.2713E+02(4)	5.2684E+02(2)
f_{23}	1.0922E+03(5.5)	1.0802E+03(2)	1.0805E+03(3)	1.0920E+03(4)	1.0922E+03(5.5)	**1.0709E+03(1)**
f_{24}	4.0504E+02(6)	3.8153E+02(2)	3.8756E+02(3)	3.9153E+02(5)	3.8791E+02(4)	**3.8074E+02(1)**
f_{25}	4.0360E+02(6)	3.9224E+02(2)	3.9541E+02(3)	3.9814E+02(4)	3.9932E+02(5)	**3.9177E+02(1)**
Average rank	4.84	3.14	2.98	4.46	3.98	1.6

run value $f(X_{best})$ and the actual optimum f^* of a particular function, that is $f(X_{best}) - f^*$.

A nonparametric statistical test called Friedman test[29,30] and Iman-Davenport extension[31] for independent samples were conducted at the 0.05 significance level in order to judge whether all the six algorithms obtain similar results with nonsignificant difference[32,33]. The post hoc Holm procedure with adjusted p-values[34] was used to obtain what algorithms are better or worse than our proposal after the null hypothesis was rejected through Friedman and Iman-Davenport extension tests.

A p-value provides information about whether a statistical hypothesis test is significant or not, and also indicates "how significant" the result is: the smaller the p-value, the stronger the evidence against the null hypothesis, indicating the better final objective function achieved by the best algorithm in each case statistically significant and have not occurred by chance[33].

4.4. Results on numerical benchmarks

Tables 3 and 4 show the mean and ranks of the best *error values* for 25 independent runs of the six algorithms on 25 numerical benchmarks for *D*=10 and *D*=30, respectively. We rank the six algorithms for each function separately, giving the algorithm with the lowest function value a rank of 1, the next lowest a rank of 2, etc, as shown in Tables 3 and 4. In case of ties, average ranks are assigned. The result of applying Friedman and Iman-Davenport extension tests is shown in Table 5, the null hypothesis for dimension *D*=10 and 30 are rejected for 15.8393>2.29, 20.4767>2.29. Then we use the proposed algorithm DE-F&CR as the control method, and the comparative results obtained by the Holm

Table 4. Mean and ranks of the best *error values* for D=30

Function	DE/rand/1/bin ($F = 0.5, CR = 0.9$)	jDE	SaDE	CDEMD ($F = 0.5$)	DE-F ($CR = 0.5$)	DE-F& CR
f_1	1.1106E-07(4)	4.5236E-25(2)	**6.5286E-26(1)**	1.3212E-07(5)	2.1316E-07(6)	3.2132E-24(3)
f_2	9.0341E-04(5)	2.5364E-04(2)	**6.4652E-05(1)**	1.8247E-03(6)	8.3531E-04(4)	3.2328E-04(3)
f_3	2.0843E+06(5)	3.2663E+04(3)	2.0521E+04(2)	2.2735E+06(6)	1.0140E+06(4)	**1.0682E+03(1)**
f_4	5.9369E-03(5)	2.5073E-03(2)	**4.8601E-05(1)**	1.0639E+01(6)	5.5105E-03(4)	5.1443E-03(3)
f_5	1.6062E+03(5)	2.1208E+03(6)	7.7268E+02(2)	1.5757E+03(4)	1.4689E+03(3)	**1.7500E+02(1)**
f_6	4.2355E+01(4)	1.5432E+01(2)	3.3211E+01(3)	5.4534E+01(5)	7.3721E+01(6)	**3.7766E+00(1)**
f_7	1.7380E-02(6)	3.4562E-03(2)	5.7254E-03(4)	6.6601E-03(5)	4.9737E-03(3)	**1.1102E-16(1)**
f_8	2.1458E+01(6)	2.0932E+01(3)	2.0988E+01(4)	2.0921E+01(2)	2.1023E+01(5)	**2.0906E+01(1)**
f_9	1.8496E+02(5)	1.1869E-15(1)	2.3727E-14(2)	1.8222E+02(4)	1.9088E+02(6)	**1.7724E-08(3)**
f_{10}	5.5758E+02(6)	2.7736E+02(5)	2.0061E+02(3)	2.0878E+02(4)	1.9644E+02(2)	**1.9246E+02(1)**
f_{11}	5.2736E+01(6)	3.2670E+01(2)	4.0815E+01(4)	4.0413E+01(3)	4.1099E+01(5)	**2.0672E+01(1)**
f_{12}	9.5148E+05(4)	3.3465E+04(2)	**7.1132E+03(1)**	1.1357E+06(6)	1.0243E+06(5)	8.8059E+05(3)
f_{13}	1.7042E+01(6)	1.6586E+01(3)	1.5128E+01(2)	1.6719E+01(4)	1.6849E+01(5)	**1.6898E+00(1)**
f_{14}	1.3970E+01(6)	1.3968E+01(5)	1.3955E+01(3)	1.3949E+01(2)	1.3960E+01(4)	**1.3816E+01(1)**
f_{15}	8.7016E+02(6)	8.6503E+02(5)	8.6331E+02(4)	8.6069E+02(3)	8.4226E+02(2)	**8.2675E+02(1)**
f_{16}	2.2434E+02(4)	1.5428E+02(3)	1.1385E+02(2)	2.2706E+02(5)	3.1008E+02(6)	**2.4329E+01(1)**
f_{17}	2.8044E+02(5)	2.6189E+02(4)	1.5809E+02(2)	2.5775E+02(3)	5.3420E+02(6)	**8.7529E+01(1)**
f_{18}	8.6741E+02(5)	8.6611E+02(4)	9.5342E+02(6)	8.2386E+02(3)	8.2094E+02(2)	**7.2435E+02(1)**
f_{19}	8.5362E+02(6)	8.4801E+02(5)	8.4580E+02(4)	8.2300E+02(2)	8.2659E+02(3)	**8.2140E+02(1)**
f_{20}	8.2395E+02(4)	8.5366E+02(5)	2.1413E+03(6)	8.2347E+02(3)	8.2194E+02(2)	**8.1950E+02(1)**
f_{21}	8.5993E+02(5)	8.5017E+02(2)	1.5864E+03(6)	8.5986E+02(4)	8.5825E+02(3)	**6.6023E+02(1)**
f_{22}	5.0275E+02(5)	5.0242E+02(4)	1.1653E+03(6)	5.0209E+02(3)	5.0135E+02(2)	**5.0073E+02(1)**
f_{23}	8.6834E+02(6)	8.6625E+02(3)	5.7560E+02(2)	8.6730E+02(5)	8.6701E+02(4)	**5.6358E+02(1)**
f_{24}	2.1142E+02(6)	2.1081E+02(4)	2.0985E+02(2)	2.1069E+02(3)	2.1114E+02(5)	**2.0588E+02(1)**
f_{25}	9.1002E+02(6)	8.6712E+02(5)	5.0212E+02(4)	2.1143E+02(3)	2.1061E+02(2)	**2.1057E+02(1)**
Average rank	5.24	3.36	3.08	3.96	3.96	1.4

procedure are shown in Table 6 for D=10 and Table 7 for D=30.

In Table 3, considering the mean of the best *error values* for 10- dimensional problems, on average, DE-F&CR ranked the first with rank 1.6; SaDE ranked the second with rank 2.98; jDE ranked the third with rank 3.14; DE-F ranked the fourth with rank 3.98; CDEMD ranked the fifth with rank 4.46, and the last is DE/rand/1/bin with rank 4.84. Table 6 indicate that DE-F&CR outperformed all the other algorithms in a statistically fashion, as we rejected all the five null hypothesis according that their corresponding p-value is less than adjusted α-value, i.e, 0.0034<0.0250, 0.0089<0.0500. For the unimodal function f_5, it was outperformed by jDE and managed to remain second best. In the multimodal functions f_7, f_9, f_{14} and hybrid composition functions f_{15}, f_{21}, f_{22}, it was beaten by jDE, SaDE and DE-F, and remained comparable to those of the above algorithms, only in function f_9 DE-F&CR ranked the fifth with the worst performance.

For the 30-dimensional problems, it become more difficult than their 10-dimensional counterparts. Hence, the results are not as well as those for function with D=10 even through the maximum function evaluation is increased. Tables 4 and 7 show the mean of the best *error values* and comparative results for D=30. On average, DE-F&CR ranked the first with rank 1.4; SaDE ranked the second with rank 3.08; jDE ranked the third with rank 3.36; DE-F and CDEMD ranked the fourth and fifth with equal ranks 3.96; and the last is DE/rand/1/bin with rank 5.24. From Table 7 it can be easily seen that overall the DE-F&CR algorithm outperforms all the other compared algorithms in a statistically meaningful way, for the reason that their corresponding p-value is less than adjusted α-value, i.e, 2.0370E-04<0.0250, 0.0018<0.0500. In particular,

Table 5. Results of the Friedman and Iman-Davenport tests(α=0.05)

Dimension	Friedman value	Iman-Davenport χ_F^2	Critical value valueF_F of $F_{0.95}(5,120)$
D=10	49.6974	15.8393	2.2900
D=30	57.5489	20.4767	2.2900

Table 6. Results of the Holm procedure(D=10, α=0.05)

Index(i)	Algorithm	$z=(R_i-1.6)/0.5292$	p-value	$\alpha/(6-i)$
1	DE/rand/1/bin	$(4.84-1.6)/0.5292$	1.8643E-09	0.0100
2	CDEMD	$(4.46-1.6)/0.5292$	3.3320E-08	0.0125
3	DE-F	$(3.98-1.6)/0.5292$	1.0229E-05	0.0167
4	jDE	$(3.14-1.6)/0.5292$	0.0034	0.0250
5	SaDE	$(2.98-1.6)/0.5292$	0.0089	0.0500

Table 7. Results of the Holm procedure(D=30, α=0.05)

Index(i)	Algorithm	$z=(R_i-1.4)/0.5292$	p-value	$\alpha/(6-i)$
1	DE/rand/1/bin	$(5.24-1.4)/0.5292$	0	0.0100
2	CDEMD	$(3.96-1.4)/0.5292$	1.4991E-06	0.0125
3	DE-F	$(3.96-1.4)/0.5292$	1.4991E-06	0.0167
4	jDE	$(3.36-1.4)/0.5292$	2.0370E-04	0.0250
5	SaDE	$(3.08-1.4)/0.5292$	0.0018	0.0500

DE-F&CR was outperformed by SaDE in unimodal function f_1, f_2 and f_4. For the multimodal functions, DE-F&CR was only beaten by SaDE in f_{12}. In 21 of 25 cases, DE-F&CR could outperformed all the other variants in the mean of the best *error values*. Note that DE-F&CR has shown its superiority in rotated and shifted functions as well as function with noise.

For further illustration, we plot the convergence graph of DE-F&CR algorithm on functions 1-5, functions 6-10, functions 11-15, functions 16-20 and functions 21-25 for D=30 in Figures 1-5, respectively, in order to show the evolutionary processes and convergence of the proposed algorithm. From Figures 1-2, we could observe that the DE-F&CR algorithm almost approach the global optimum for functions f_1, f_7 and f_9. However, from f_{16} to f_{25}, it can hardly find the global optimal solution due to the high mulitmodal of those hybrid composition functions and the local search process make the algorithm prematurely converge.

Tables 3, 4, 6 and 7 indicate that, considering the mean of the best *error values* for both 10-dimensional and 30-dimensional problems, SaDE and jDE performed better than DE-F and CDEMD, because DE-F and CDEMD adjusts only one control parameter in the evolution process. Our proposed DE-F&CR algorithm with a new measure of population diversity to adjust parameter performed better than DE-F and CDEMD which use the “distance-to-midpoint” measure diversity information to adjust control parameter, as well as SaDE and jDE which do not use any the population diversity information. The proposed method performed steadily with the dimension increased, which is very good at solving benchmark functions.

4.5. *Application to data classification*

Data classification is an important tool for a variety of applications in data mining, statistical data analysis and data compression. The goal of classification is to group data into clusters such that the similarities among data members within the same cluster are maximal while similarities among data members from different clusters are minimal[35,36]. Classification can be encoded as a multi-variable optimization. When in a multi-dimension space a class of prototype is represented by a centroid, the classification can be seen as the problem of finding the optimal positions of all the class centroids. PSO, DE and ABC algorithms have already been applied to classification in literature[35,37,38]. Therefore, we compare the performance of the DE-F&CR algorithm with some DE variants over classification of real-life data sets.

For a database with C classes and N parameters, the classification problem can be seen as that of finding the optimal positions of C centroids in a N-dimensional space. Each database that we tracked will be partitioned into two sets: a training set and a testing set. The i-individual of the population in our work is encoded as follows:

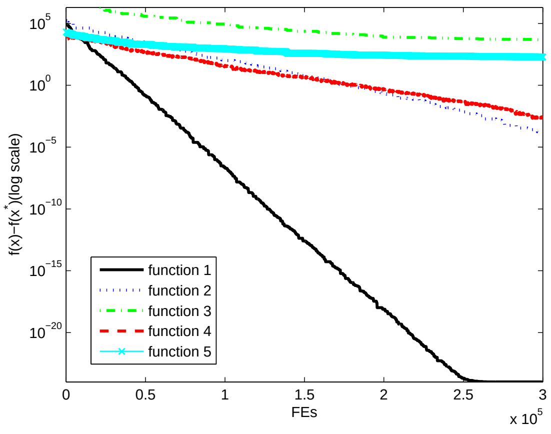

Fig. 1. Convergence Graph for $f_1 - f_5$.

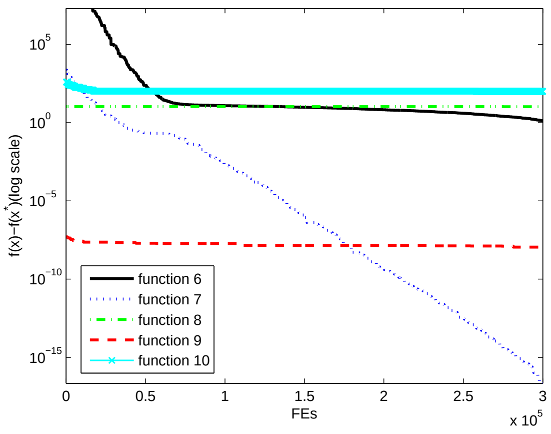

Fig. 2. Convergence Graph for $f_6 - f_{10}$.

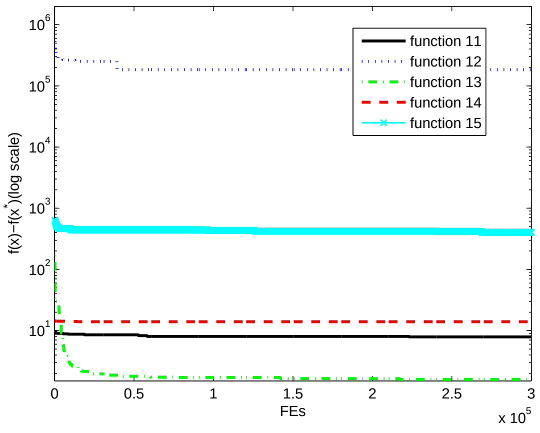

Fig. 3. Convergence Graph for $f_{11} - f_{15}$.

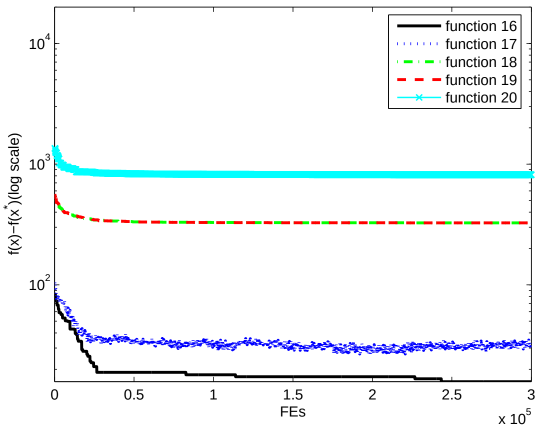

Fig. 4. Convergence Graph for $f_{16} - f_{20}$.

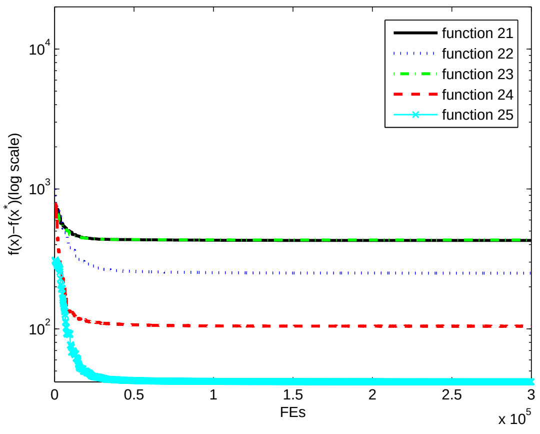

Fig. 5. Convergence Graph for $f_{21} - f_{25}$.

$$P_i = \{P_i^1, P_i^2, \cdots, P_i^C\}, \tag{10}$$

where the position of the *j*-th centroid is constituted by *N* real numbers representing its *N* coordinates:

$$P_i^j = \{p_i^1, p_i^2, \cdots, p_i^N\}, j = 1, 2, \cdots, C. \tag{11}$$

So, each individual P_i is composed by $C * N$ real value components. To evaluate the quality of individual P_i, the fitness function is designed to minimize the sum on all training set instances of Euclidean distance in *N*-dimension space between generic instance x_k and the centroid of its class according to database. The fitness of individual P_i is computed as in[35]:

$$f(P_i) = \frac{1}{D_{Train}} \sum_{k=1}^{D_{Train}} d(x_k, P_i^{CL_{konwn}(x_k)}), \tag{12}$$

where D_{Train} is the number of instance in the training set, which is used to normalize the sum. $CL_{konwn}(x_k)$ defines the class that instance x_k belongs to according to database.

Table 8. Properties of the date sets

Data	D	C	N	D_{Train}	D_{Test}
Iris	150	3	4	112	38
Wine	178	3	13	133	45
Glass	214	6	9	161	53
Thyroid	215	3	5	162	53
Heart	303	2	35	227	76

In this experiment, five well-known real-world classification problem taken from the machine learning repository[39] have been considered for investigation. The data sets and their features are presented in Table 8. For each data set, we list the total instance number *D*, the number of class *C* which it is divided, the number *N* of parameters composing each instance, the number of instance D_{Train} in the training set and the number of instance D_{Test} in the test set. The training set is assigned the former 75% of the database instances D, and the testing set the remaining 25%.

For the DE-F&CR algorithm and the compared algorithm, the parameter setting are the same as in Table 2, the population size is set as *NP*=50 and maximum generation number is G_{max}=2000. Table 9 shows the results on each of the 5 classification problem with respect to the incorrect classification percentages, which is the percentage of incorrectly classified patterns of the test data sets. We classified each pattern by assigning it to the class whose center is closest by means of the Euclidean distance. This assigned class is compared with the desired class, if they are not the same class the pattern is considered as incorrectly classified. It is calculated for all test data and the number of incorrectly classified pattern is percentaged to the D_{Test}. Results are averaged over 20 independent trial runs. The DE-F&CR algorithm achieves the best results for the Iris, Glass and Heart problem, however is beaten by SaDE for the Wine and Thyroid problem. More over, the average classification error percentages for the five problem are 14.87% for DE, 10.31% for jDE, 9.82% for SaDE, 13.34% for CDEMD, 13.10% for DE-F and 8.58% for DE-F&CR. That is, DE-F&CR ranked the first; SaDE ranked the second; jDE ranked the third; DE-F ranked the fourth; CDEMD ranked the fifth; and the last is DE. Therefore, the DE-F&CR algorithm can be successfully applied to clustering for the purpose of classification.

5. Conclusions

In the DE algorithm, control parameters play a key role in the algorithm's performance. It is difficult to choose suitable parameter values, since the best setting is depending on the nature of problem and available computation resources. Therefore, this paper proposed a new parameter control mechanism based on a novel measure of population diversity. The notion of the presented control mechanism differs from the existed one in that the factors *CR* and *F* are guided by the population's current diversity at each generation in order to maintain the population diversity at a proper level. We compared the proposed algorithm with two adaptive DE algorithms which also use diversity measure to adjust control parameter, as well as two representative self-adaptive differential evolution algorithms jDE and SaDE, the results evaluated on a set of bench-

Table 9. Average classification error percentage and ranks

Data	DE	jDE	SaDE	CDEMD	DE-F	DE-F&CR
Iris	5.33	2.63	2.63	5.26	4.87	1.32
Wine	6.67	2.78	1.64	8.87	3.79	2.22
Glass	30.19	22.64	24.53	22.71	26.42	20.75
Thyroid	5.65	3.77	1.86	7.50	9.42	2.83
Heart	26.52	19.74	18.45	22.37	21.05	15.79
Average	14.87	10.31	9.82	13.34	13.10	8.58
Rank	6	3	2	5	4	1

mark problems and the statistical analysis through the Friedman test demonstrated that the proposed algorithm was overall more effective in obtaining better quality solutions. Further study of the characteristic of the adaption scheme and improvement of the adjustment mechanism for control parameters will be done in future.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (10971242, 61074151) and Chinese State Key Laboratory of Rail Traffic Control and Safety Independent Project (Grant: RCS2009ZT002).

References

1. R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optiization over continuous spaces," *Journal of the Global Optimizaton*, **11**, 341–359 (1997).
2. R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm algorithm with ensenble of parameters and mutation strategies," *Applied Soft Computing*, **11**, 1679–1696 (2011).
3. H. Shimodaira, "A diversity control oriented genetic algorithm (DGGA): development and experimental results," *In: Proceedings of the Genetic and Evolutionary Computation Conference*, **1**, 603–611 (1999).
4. R. K. Ursem, "Diversity-guided evolutionary algorithms," *In: Proceeding of 7th International Conference Parallel Problem Solving From Nature*, **2439**, 462–471 (2002).
5. J. Riget and J. S. Vesterstrom, "A diversity-guided particle swarm optimization-the ARPSO," Technical report, EVAlife, Dept of Computer Science, University of Aarhus, Denmark, (2002).
6. D. Zaharie, "Control of population diversity and adaptation in differential evolution algorithms," *In: Proceeding of Mendel, 9th International Conference on Soft Computing*, 41–46 (2003).
7. L. S. Coelho and P. Alotto, "Electromagnetic optimization based on an inproved diversity-guided differential evolution approach and adaptive mutation factor," *The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, **28**, 1112–1120 (2009).
8. S. Das, A. Konar and U. K. Chakraborty, "Two improved differential evolution schemes for faster global search," *In: ACM-SIGEVO proceedings of genetic and evolutionary computation comference, Washington DC, USA*, 991–998 (2005).
9. R. Gämperle, S. D. Müller and P. Koumoutsakos, "A parameter study for differential evolution," *Advances in Intelligence Systems, Fuzzy Systems, Evolutionary Computation*, 293–298 (2002).
10. J. Liu and J. Lampinen, "Adaptive parameter control of differential evolution," *In: Proceeding of Mendel, 8th International Conference on Soft Computing*, 19–26 (2002).
11. J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, **9**, 448–462 (2005).
12. Q. K. Pan, P. N. Suganthan and L. Wamg, et al., "A differential evolution algorithm with self-adapting strategy and control parameters," *Computers & Operations Research*, **38**, 394–408 (2011).
13. F. Neri and V. Tirronen, "Recent advances in differential evolution: a survey and experimental analysis," *Artificial Intelligence Review*, **33(1-2)**, 61–106 (2010).
14. J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *soft computing- a fusion of foundations, Methodologies and Applications*, **10(8)**, 673–686 (2006).
15. J. Brest, S. Greiner, B. Bošković, M. Mernik and V.

Žumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, **10**, 646–657 (2006).

16. A. K. Qin, V. L. Huang and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computations*, **13**, 398–417 (2009).
17. A. Ghosh, S. Das, A. Chowdhury and R. Giri, "An improved differential evolution algorithm with fitness-based adaption of the control parameters," *Information Scinece*, **181**, 3794–3765 (2011).
18. L. S. Coelho, R. C. T. Souza and V. C. Mariani, "Improved differential evolution approach based on cultrural algorithm and diversity measure applied to solve economic load dispatch problems," *Mathematics and Computers in Simulstion*, **79**, 3136–3147 (2009).
19. D. Zaharie, "Parameter adaptation in differential evolution by controlling the population diversity," *In: Proceeding of 4th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing*, 385–397 (2003).
20. K. Price, "Differential evolution: a fast and simple numerical optimizer," *Biennial Conference of the North American, Fuzzy Information Processing Society*, 524–527 (1996).
21. M. Pant, T. Radha and V. P. Singh, "A simple diversity guided particle swarm optimization," *IEEE Congress on Evolutionary Computation*, 3294–3299 (2007).
22. P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger and S. Tiwari, "Problem definitions and evoluation criteria for the CEC 2005 special session on real-parameter optimization," Technical Report, Nanyang Technological University, Singapore, (2005).
23. X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, **3(2)**, 82–102 (1999).
24. C. Y. Lee and X. Yao, "Evolutionary programming using mutations based on the L*é*vy probability distribution," *IEEE Transactions on Evolutionary Computation*, **8(1)**, 1–13 (2004).
25. A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," *In Proceedings of IEEE Congress on Evolutionary Computation, Scotland*, 1785–1791 (2005).
26. M. G. H. Omran, A. Salman, and A. P. Engelbrecht, "Self-adaptive differential evolution," *in Lecture Notes in Artificial Intelligence. Berlin, Germany: Springer-Verlag*, 192–199 (2005).
27. N. Noman, D. Bollegala and H. Iba, "An adaptive differential evolution algorithm," *IEEE Congress on Evolutionary Computation*, 2229–2235 (2011).
28. S. M. Islam, S. Das, S. Ghosh, S. Roy and P. N. Sughenthan, "An Adaptive Differential Evolution Algorithm With Novel Mutation and Crossover Strategies for Global Numerical Optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **42(2)**, 482–500 (2012).
29. M. Friedman, "The use of ranks to aviod the assumption of normality implicit in the analysis of variance," *Journal of the Amerrican Statistical Association*, **37**, 674–701 (1937).
30. M. Friedman, "A comparison of alternative tests significance for the problem of m rankings," *Annals of Mathematical Statistics*, **11**, 86–92 (1940).
31. R. L. Iman and J. M. Davenport, "Approximations of the critical region of the friedman statistic," *Communications in Statistics*, **9**, 571–595 (1980).
32. S. García, D. Molina, M. Lozano and F. Herrera, "A study on the use of non-parameteric tests for analyzing the evolutionary algorithms' behavior: a case study on the CEC'2005 Special Session on Real Parameter Optimization," *Journal of Heuristica*, **15(6)**, 617–644 (2009).
33. S. García, A. Fernández, J. Luengo and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining," *Experimental Analysis of Power, Information Sciences*, **180**, 2044–2064 (2010).
34. S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, **6**, 65–70 (1979).
35. D. Karaboga and C. Ozturk, "A novel clustering approach: Artificial Bee Colony (ABC) algorithm," *Applied Soft Computing*, **11**, 652–657 (2011).
36. D.J. Hand, H. Mannila and P. Smyth, "Principles of Data Mining," *The MIT Press*, (2001).
37. I. De Falco, A. Della Cioppa and E. Tarantino, "Facing classification problems with Particle Swarm Optimization, " *Applied Soft Computing*, **7(3)**, 652–658 (2007).
38. S. Paterlinia and T. Krink, "Differential evolution and particle swarm optimisation in partitional clustering," *Computational Statistics & Data Analysis*, **50(5)**, 1220–1247 (2006).
39. C.L. Blake and C.J. Merz, "University of California at Irvine Repository of Machine Learning Databases," 1998, http://www.ics.uci.edu/ mlearn/MLRepository.html.