2011 IEEE International Conference on Fuzzy Systems
June 27-30, 2011, Taipei, Taiwan

Fuzzy C-Means Clustering and Partition Entropy for Species-Best Strategy and Search Mode Selection in Nonlinear Optimization by Differential Evolution

Tetsuyuki Takahama
Department of Intelligent Systems
Hiroshima City University
Asaminami-ku, Hiroshima, 731-3194 Japan
Email: takahama@info.hiroshima-cu.ac.jp

Setsuko Sakai
Faculty of Commercial Sciences
Hiroshima Shudo University
Asaminami-ku, Hiroshima, 731-3195 Japan
Email: setuko@shudo-u.ac.jp

***Abstract*— Differential Evolution (DE) is a newly proposed evolutionary algorithm. DE is a stochastic direct search method using a population or multiple search points. DE has been successfully applied to optimization problems including non-linear, non-differentiable, non-convex and multimodal functions. However, the performance of DE degrades in problems having strong dependence among variables, where variables are related strongly to each other. In this study, we propose to utilize partition entropy given by fuzzy clustering for solving the degradation. It is thought that a directional search is desirable when search points are distributed with bias. Thus, when the entropy is low, algorithm parameters can be controlled to make the directional search. Also, we propose to use a species-best strategy for improving the efficiency and the robustness of DE. The effect of the proposed method is shown by solving some benchmark problems.**

***Index Terms*—differential evolution; rotation-invariant; intensive search; extensive search**

I. Introduction

Optimization problems, especially nonlinear optimization problems, are very important and frequently appear in the real world. There exist many studies on solving optimization problems using evolutionary algorithms (EAs). Differential evolution (DE) is a newly proposed EA by Storn and Price [1]. DE is a stochastic direct search method using a population or multiple search points. DE has been successfully applied to optimization problems including non-linear, non-differentiable, non-convex and multimodal functions [2], [3]. It has been shown that DE is a very fast and robust algorithm.

However, the performance of DE degrades in problems having strong dependence among variables, where variables are related strongly to each other. It is very important to know the distribution of search points to solve the degradation. For example, when search points are uniformly distributed as in the left part of Fig. 1, a nondirectional search is the better choice than a directional search. On the contrary, when variables are strongly related as in the right part of the figure, a directional search is the better choice.

In this study, in order to know the distribution, we propose to utilize partition entropy given by fuzzy clustering. If the entropy is very high, search points are often uniformly distributed. If the entropy is low, search points are distributed with a bias. Thus, when the entropy is low, algorithm parameters can be controlled to make the directional search. It is thought that this idea can be applicable to other EAs than DE.

Also, we propose to use a species-best strategy [4] for improving the efficiency and the robustness of DE. In DE, a mutant vector is generated for each parent by using a base vector and one or more difference vectors which are the difference between two individuals. The parent and the mutant vector are recombined by a crossover operation to generate a child, or a trial vector. There are some strategies for selecting the base vector: The best individual is used as the base vector in the best strategy and a randomly selected individual is used in the rand strategy. In the species-best strategy, a population is divided into several species by speciation, and the seed of the species to which the parent belongs is selected as the base vector. It is thought that the efficiency of the species-best strategy is better than the rand strategy and the robustness of the species-best strategy is better than the best strategy. Thus, it is expected that the strategy improves the efficiency and the robustness of the search.

The effect of the proposed method is shown by solving 13 benchmark problems including multimodal problems and problems with strong dependence.

In Section II, some studies on DE using fuzzy set theory or speciation are briefly reviewed. Fuzzy clustering and par-

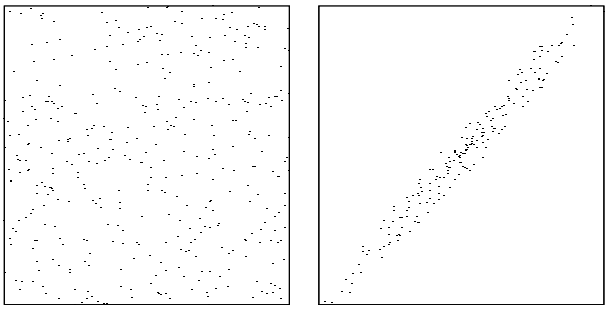

Fig. 1. Uniform Search (left) and Directional Search (right)

978-1-4244-7316-8/11/$26.00 ©2011 IEEE

tition entropy are explained in Section III. DE and DE with speciation using fuzzy clustering are described in Section IV and V, respectively. In Section VI, experimental results on some problems are shown. Finally, conclusions are described in Section VII.

II. Previous Works

In this section, the optimization problem in this study is defined and some studies on DE using fuzzy set theory or speciation are described.

A. Optimization Problems

In this study, the following optimization problem with lower bound and upper bound constraints will be discussed.

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & l_i \leq x_i \leq u_i,\ i = 1, \ldots, D, \end{array} \tag{1}$$

where $\boldsymbol{x} = (x_1, x_2, \cdots, x_D)$ is a D dimensional vector and $f(\boldsymbol{x})$ is an objective function. The function f is a nonlinear real-valued function. Values l_i and u_i are the lower bound and the upper bound of x_i, respectively. Let the search space in which every point satisfies the lower and upper bound constraints be denoted by $\mathcal{S}$.

B. Fuzzy Set Theory

There exist some studies on optimization by DE using fuzzy set theory.

- Fuzzy logic: Fuzzy logic is used to control the parameters of DE. The fuzzy adaptive differential evolution (FADE) [5] is proposed to control parameters for a mutation operation and a crossover operation. In each generation, the movement of individuals (vectors) and the change of function values over the whole population between the last two generations were nonlinearly depressed and used as the inputs for fuzzy logic controllers.
- Fuzzy clustering: The hybrid DE based on fuzzy c-means clustering (FCDE) [6] is proposed, which uses the one-step fuzzy c-means clustering. The clustering acts as a multi-parent crossover operation to utilize the information of a population efficiently. In [7], fuzzy clustering is used to divide a population into several clusters or niches, which are a kind of species described below, for multimodal optimization.

C. Speciation

Speciation is mainly used for multimodal optimization where multiple optimal or suboptimal solutions are obtained simultaneously in one run. Each species evolves to find an optimal or suboptimal solution. There exist some types of research using speciation in DE [8].

- Radius-based speciation: In this category, a population is sorted in increasing objective value order, first. Then, the best individual in the sorted population becomes a new species seed. The population members that exist within the specified radius from the seed are assigned to the species, and the members are deleted from the population. This process is repeated until the population becomes empty [4], [9], [10].
- Clustering-based speciation
  A population is divided into several clusters using a clustering algorithm such as k-means clustering [11] or fuzzy c-means clustering [7]. Each cluster corresponds to a species. If the seed of a species is necessary, an individual that has the best objective value in the species is selected as the seed.

In this study, the clustering-based speciation using fuzzy c-means clustering is used not to multimodal optimization but to usual optimization for finding one optimal solution and improving the efficiency of DE using the species-best strategy. Also, as a quite new approach, the partition entropy obtained by the clustering is used to control the parameter for crossover operation of DE.

III. Fuzzy C-Means Clustering and Entropy

In this section, fuzzy c-means clustering and partition entropy are briefly explained [7], [12].

A. Fuzzy Partition and Partition Entropy

Fuzzy partition allows a data point to belong to two or more clusters. Let $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$ be a set of N data points. A fuzzy partition of X into C clusters can be defined with a matrix $U = \{\mu_{ij}\}, 1 \leq i \leq N, 1 \leq j \leq C$, which satisfies the following conditions:

$$\mu_{ij} \in [0, 1] \tag{2}$$

$$\sum_{j=1}^{C} \mu_{ij} = 1,\ 1 \leq i \leq N \tag{3}$$

$$0 < \sum_{i=1}^{N} \mu_{ij} < N,\ 1 \leq j \leq C \tag{4}$$

where μ_{ij} is the degree of membership of the i-th data $\boldsymbol{x}_i$ in the j-th cluster. The higher μ_{ij} indicates that the i-th data belongs to the j-th cluster more strongly.

Some evaluation criteria for the fuzzy partition are proposed such as partition entropy and so on. The normalized partition entropy is defined as follows:

$$PE(C) = -\frac{1}{N \log_2 C} \sum_{i=1}^{N} \sum_{j=1}^{C} \mu_{ij} \log_2 \mu_{ij} \tag{5}$$

where $0 \leq PE(C) \leq 1$. The minimum value 0 corresponds to a hard partition, where each data point $\boldsymbol{x}_i$ belongs to only a cluster c_i:

$$\mu_{ij} = \begin{cases} 1 & j = c_i \\ 0 & j \neq c_i \end{cases}, 1 \leq i \leq N \tag{6}$$

The maximum value 1 corresponds to the fuzziest partition, where each data point belongs to all clusters equivalently:

$$\mu_{ij} = \frac{1}{C},\ 1 \leq i \leq N, 1 \leq j \leq C \tag{7}$$

In this study, the entropy is used to detect the distribution of search points and alter the mode of the search by DE.

B. Fuzzy C-Means Clustering

Fuzzy C-Means (FCM) is a clustering method which realizes the fuzzy partition. This method is frequently used in pattern recognition. The FCM algorithm minimizes the following function:

$$J_m(U,V) = \sum_{i=1}^{N}\sum_{j=1}^{C}(\mu_{ij})^m ||\boldsymbol{x}_i - \boldsymbol{v}_j||^2,\ 1 \le m < \infty \quad (8)$$

where m is a real number parameter specifying degree of fuzziness, $\boldsymbol{v}_j$ is the center of the j-th cluster, $V = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_C\}$ is a set of the cluster centers, and $||\cdot||$ represents any norm expressing the distance between a data point and the cluster center.

The FCM algorithm is as follows:

1) Assign membership values μ_{ij} of all data in all clusters randomly.
2) Update cluster centers.

$$\boldsymbol{v}_j = \frac{\sum_{i=1}^{N}(\mu_{ij})^m x_i}{\sum_{i=1}^{N}(\mu_{ij})^m} \quad (9)$$

3) Update membership degrees.

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C}\left(\frac{||\boldsymbol{x}_i - \boldsymbol{v}_j||^2}{||\boldsymbol{x}_i - \boldsymbol{v}_k||^2}\right)^{\frac{1}{m-1}}} \quad (10)$$

4) The algorithm is terminated, when the following condition is satisfied, or the change of membership degrees between two iterations is no more than ε, the given sensitivity threshold.

$$\sum_{i=1}^{N}\sum_{j=1}^{C}(\mu_{ij}^m(t) - \mu_{ij}^m(t-1))^2 \le \varepsilon \quad (11)$$

where $\mu_{ij}^m(t)$ and $\mu_{ij}^m(t-1)$ are the value at the current iteration and that at the previous iteration, respectively.
5) Go back to 2).

This algorithm minimizes intra-cluster variance, but the minimum is a local minimum and the results depend on the initial choice of membership degrees.

IV. Differential Evolution

In this section, the outline of DE is described.

A. Outline of Differential Evolution

In DE, initial individuals are randomly generated within the given search space and form an initial population. Each individual contains D genes as decision variables. At each generation or iteration, all individuals are selected as parents. Each parent is processed as follows: The mutation operation begins by choosing several individuals from the population except for the parent in the processing. The first individual is a base vector. All subsequent individuals are paired to create difference vectors. The difference vectors are scaled by a scaling factor F and added to the base vector. The resulting vector, or a mutant vector, is then recombined with the parent. The probability of recombination at an element is controlled by a crossover rate CR. This crossover operation produces a trial vector. Finally, for survivor selection, the trial vector is accepted for the next generation if the trial vector is better than the parent.

There are some variants of DE that have been proposed. The variants are classified using the notation DE/$base$/num/$cross$ such as DE/rand/1/bin and DE/rand/1/exp.

"$base$" specifies a way of selecting an individual that will form the base vector. For example, DE/rand selects an individual for the base vector at random from the population. DE/best selects the best individual in the population.

"num" specifies the number of difference vectors used to perturb the base vector. In case of DE/rand/1, for example, for each parent $\boldsymbol{x}^i$, three individuals $\boldsymbol{x}^{p1}$, $\boldsymbol{x}^{p2}$ and $\boldsymbol{x}^{p3}$ are chosen randomly from the population without overlapping $\boldsymbol{x}^i$ and each other. A new vector, or a mutant vector $\boldsymbol{x}'$ is generated by the base vector $\boldsymbol{x}^{p1}$ and the difference vector $\boldsymbol{x}^{p2} - \boldsymbol{x}^{p3}$, where F is the scaling factor.

$$\boldsymbol{x}' = \boldsymbol{x}^{p1} + F(\boldsymbol{x}^{p2} - \boldsymbol{x}^{p3}) \quad (12)$$

"$cross$" specifies the type of crossover that is used to create a child. For example, 'bin' indicates that the crossover is controlled by the binomial crossover using a constant crossover rate, and 'exp' indicates that the crossover is controlled by a kind of two-point crossover using exponentially decreasing the crossover rate. Fig. 2 shows the binomial and exponential crossover. A new child $\boldsymbol{x}^{\text{child}}$ is generated from the parent $\boldsymbol{x}^i$ and the mutant vector $\boldsymbol{x}'$, where CR is a crossover rate.

```
binomial crossover DE/·/·/bin
  j_rand=randint(1,D);
  for(k=1; k≤D; k++) {
    if(k == j_rand || u(0,1) < CR) x_k^child=x'_k;
    else x_k^child=x_k^i;
  }
exponential crossover DE/·/·/exp
  k=1; j=randint(1,D);
  do {
      x_j^child=x'_j;
      k=k+1; j=(j+1)%D;
  } while(k ≤ D && u(0,1) < CR);
  while(k ≤ D) {
      x_j^child=x_j^i;
      k=k+1; j=(j+1)%D;
  }
```

Fig. 2. Binomial and exponential crossover operation, where randint(1,D) generates an integer randomly from $[1, D]$ and $u(l, r)$ is a uniform random number generator in $[l, r]$.

B. The Algorithm of Differential Evolution

The algorithm of DE is as follows:

Step1 Initialization of a population. Initial N individuals $P = \{\boldsymbol{x}^i, i = 1, 2, \cdots, N\}$ are generated randomly in the search space and form an initial population.

Step2 Termination condition. If the number of function evaluations exceeds the maximum number of evaluation $FE_{\max}$, the algorithm is terminated.

Step3 DE operations. Each individual $\boldsymbol{x}^i$ is selected as a parent. If all individuals are selected, go to Step4. A mutant vector $\boldsymbol{x}'$ is generated according to Eq. (12). A trial vector (child) is generated from the parent $\boldsymbol{x}^i$ and the mutant vector $\boldsymbol{x}'$ using a crossover operation shown in Fig. 2. If the child is better than or equal to the parent, or the DE operation is succeeded, the child survives. Otherwise the parent survives. Go back to Step3 and the next individual is selected as a parent.

Step4 Survivor selection (generation change). The population is organized by the survivors. Go back to Step2.

Fig. 3 shows a pseudo-code of DE/rand/1.

```
DE/rand/1()
{
// Initialize a population
 P=N individuals generated randomly in S;
 for(t=1; FE ≤ FE_max; t++) {
  for(i=1; i ≤ N; i++) {
// DE operations
     x^p1=Randomly selected from P(p1 ≠ i);
     x^p2=Randomly selected from P(p2 ∉ {i,p1});
     x^p3=Randomly selected from P(p3 ∉ {i,p1,p2});
     x'=x^p1+F(x^p2 − x^p3);
     x^child=trial vector is generated from
           x^i and x' by the crossover operation;
// Survivor selection
     if(f(x^child) ≤ f(x^i))  z^i=x^child;
     else                     z^i=x^i;
     FE=FE+1;
  }
  P={z^i,  i = 1,2,···,N};
 }
}
```

Fig. 3. The pseudo-code of DE, FE is the number of function evaluations.

V. DE with Speciation using Fuzzy Clustering

In this section, DE with speciation using fuzzy clustering (DESFC) is proposed.

A. Species-Best Strategy

The following steps are executed in each generation to realize the species-best strategy.

1) FCM is applied to a population $P = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^N\}$ where N is the population size. The membership grade of the i-th individual $\boldsymbol{x}^i$ in the j-th cluster, μ_{ij}, $1 \leq i \leq N$, $1 \leq j \leq C$ is obtained where C is the number of clusters.
2) Each cluster j corresponds to one species. The i-th individual is assigned to the cluster or the species S_i where μ_{ij} has the maximum value.

$$S_i = \arg \max_{1 \leq j \leq C} \mu_{ij},\ 1 \leq i \leq N \tag{13}$$

3) The seed of j-th species, $seed_j$ is the best individual in the species.

$$seed_j = \arg \min_{\{i | S_i = j\}} f(\boldsymbol{x}^i),\ 1 \leq j \leq C \tag{14}$$

Thus, the seed for the i-th individual can be obtained as $seed_{S_i}$.

The species-best strategy to $\boldsymbol{x}^i$ can be described as follows:

$$\boldsymbol{x}' = \boldsymbol{x}^{seed_{S_i}} + F(\boldsymbol{x}^{p2} - \boldsymbol{x}^{p3}) \tag{15}$$

where $p2$ and $p3$ are random integers in $[1, N]$ and i, $p2$ and $p3$ are different with each other.

B. Algorithm of DESFC

Fig. 4 shows the pseudo-code of DESFC. Some modifications to standard DE are applied for proposed method as follows:

1) The partition entropy $PE(C)$ for $\{\mu_{ij}\}$ is calculated in each generation. If the entropy is low, or $PE(C) < 0.99$, a directional search is desirable. In this study, the crossover rate is set to 0.95 or 0.1 with equal probability using the binomial crossover to search for an area near the parent or the mutant vector. In order to improve the robustness, the species-best strategy and the rand strategy are adopted probabilistically.
2) Continuous generation model [13], [14] is adopted. Usually discrete generation model is adopted in DE and when the child is better than the parent, the child survives in the next generation. In this study, when the child is better than the parent, the parent is immediately replaced by the child. It is thought that the continuous generation model improves efficiency because the model can use newer information than the discrete model.
3) Reflecting back out-of-bound solutions [15] is adopted. In order to keep bound constraints, an operation to move a point outside of the search space $\mathcal{S}$ into the inside of $\mathcal{S}$ is required. There are some ways to realize the movement: generating solutions again, cutting off the solutions on the boundary, and reflecting points back to the inside of the boundary [16]. In this study, reflecting back is used:

$$x_{ij} = \begin{cases} l_i + (l_i - x_{ij}) - \left\lfloor \frac{l_i - x_{ij}}{u_i - l_i} \right\rfloor (u_i - l_i) & (x_{ij} < l_i) \\ u_i - (x_{ij} - u_i) + \left\lfloor \frac{x_{ij} - u_i}{u_i - l_i} \right\rfloor (u_i - l_i) & (x_{ij} > u_i) \\ x_{ij} & (\text{otherwise}) \end{cases} \tag{16}$$

where $\lfloor z \rfloor$ is the maximum integer smaller than or equal to z. This operation is applied when a new point is generated by DE operations.

VI. Solving Optimization Problems

In this paper, well-known thirteen benchmark problems are solved.

A. Test Problems and Experimental Conditions

The 13 scalable benchmark functions are shown in Table I [17]. All functions have an optimal value 0. Some characteristics are briefly summarized as follows: Functions f_1 to f_4 are continuous unimodal functions. The function f_5 is Rosenbrock function which is unimodal for 2- and 3-dimensions but may have multiple minima in high dimension cases [18]. The

```
DESFC()
{
// Initialize a population
 P=N individuals generated randomly in S;
 for(t=1; FE ≤ FE_max; t++) {
// Speciation using FCM
   {μ_ij}=Fuzzy c-means clustering of P;
   for(i=1; i ≤ N; i++)
     S_i=arg max_{1≤j≤C} μ_ij;
   for(j=1; i ≤ C; j++)
     seed_j=arg min_{i|S_i=j} f(x^i);
// Judge distribution of search points
   PE=partition entropy of {μ_ij};
   if(PE ≥ 0.99) isUniform=TRUE;
   else          isUniform=FALSE;

   for(i=1; i ≤ N; i++) {
     CR=CR_0; Select exponential crossover;
     if(i != seed_{s_i} && u(0,1) < 0.6) {
       if(isUniform==FALSE) {
         if(u(0,1) < 0.5) CR=0.95;
         else             CR=0.1;
         Select binomial crossover;
       }
// DE/species-best/1/{bin,exp}
       p_1=seed_{S_i};
     }
     else
// DE/rand/1/exp
     x^{p_1}=Randomly selected from P(p_1 ≠ i);
     x^{p_2}=Randomly selected from P(p_2 ∉ {i,p_1});
     x^{p_3}=Randomly selected from P(p_3 ∉ {i,p_1,p_2});
     x'=x^{p_1}+F(x^{p_2} − x^{p_3});
     x^c=trial vector is generated from
               x^i and x' by the selected crossover;
     FE=FE+1;
// Survivor selection
     if(f(x^c) ≤ f(x^i))
       x^i=x^c;
   }
 }
}
```

Fig. 4. The pseudo-code of DESFC where CR_0 is an initial crossover rate.

function f_6 is a discontinuous step function, and f_7 is a noisy quartic function. Functions f_8 to f_{13} are multimodal functions and the number of their local minima increases exponentially with the problem dimension [19].

Independent 30 runs are performed for the 13 problems. The dimension of problems is 40 (D=40). Each run stops when a near optimal solution, which has equivalent objective value to the optimal solution, is found. In this study, when the difference between the best objective value and the optimal value becomes less than 10^{-7}, the run stops. In f_7, it is difficult to find the good objective value, because a random noise is added. It is assumed that the optimal value of f_7 is 10^{-2} in this experiment.

The efficiency of two algorithms DE/rand (continuous generation model) and DESFC are compared where the number of clusters C is changed from 2 to 5 in DESFC. The degree of fuzziness m is fixed to 2. The parameters are: $F = 0.7$, $CR = 0.9$ and the exponential crossover is adopted as usual crossover, because these settings showed very good and stable performance [21]. The population size is twice of the dimension size, $N = 2D = 80$. The number of function evaluations (FEs) until finding a near optimal solution is compared. If the number of FEs exceeds 2,000,000, the run is terminated and regarded as a failure run.

TABLE I
TEST FUNCTIONS OF DIMENSION D. THESE ARE SPHERE, SCHWEFEL 2.22, SCHWEFEL 1.2, SCHWEFEL 2.21, ROSENBROCK, STEP, NOISY QUARTIC, SCHWEFEL 2.26, RASTRIGIN, ACKLEY, GRIEWANK, AND TWO PENALIZED FUNCTIONS, RESPECTIVELY [20]

Test functions	Bound constraints
$f_1(\boldsymbol{x}) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$
$f_2(\boldsymbol{x}) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]^D$
$f_3(\boldsymbol{x}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]^D$
$f_4(\boldsymbol{x}) = \max_i \{ \lvert x_i \rvert \}$	$[-100, 100]^D$
$f_5(\boldsymbol{x}) = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right]$	$[-30, 30]^D$
$f_6(\boldsymbol{x}) = \sum_{i=1}^{D} \lfloor x_i + 0.5 \rfloor^2$	$[-100, 100]^D$
$f_7(\boldsymbol{x}) = \sum_{i=1}^{D} i x_i^4 + rand[0, 1)$	$[-1.28, 1.28]^D$
$f_8(\boldsymbol{x}) = \sum_{i=1}^{D} -x_i \sin \sqrt{\lvert x_i \rvert}$ $+ D \cdot 418.98288727243369$	$[-500, 500]^D$
$f_9(\boldsymbol{x}) = \sum_{i=1}^{D} \left[x_i^2 - 10 \cos(2\pi x_i) + 10 \right]$	$[-5.12, 5.12]^D$
$f_{10}(\boldsymbol{x}) = -20 \exp \left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2} \right)$ $- \exp \left(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i) \right) + 20 + e$	$[-32, 32]^D$
$f_{11}(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos \left(\frac{x_i}{\sqrt{i}} \right) + 1$	$[-600, 600]^D$
$f_{12}(x) = \frac{\pi}{D} [10 \sin^2(\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2$ $\{1 + 10 \sin^2(\pi y_{i+1})\} + (y_D - 1)^2]$ $+ \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ where $y_i = 1 + \frac{1}{4}(x_i + 1)$ and $u(x_i, a, k, m) =$ $\begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-50, 50]^D$
$f_{13}(x) = 0.1[\sin^2(3\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2$ $\{1 + \sin^2(3\pi x_{i+1})\} + (x_D - 1)^2$ $\{1 + \sin^2(2\pi x_D)\}] + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	$[-50, 50]^D$

B. Experimental Results

Table II shows the experimental results. The mean number of FEs until finding a near optimal value and their standard deviation are shown in the top row for each function. Also, the ratio of the mean number of FEs relative to that of the standard DE is shown in the bottom row using parentheses. The best result is highlighted using bold face fonts.

TABLE II
EXPERIMENTAL RESULTS. MEAN VALUE ± STANDARD DEVIATION AND RATIO OF THE MEAN VALUE RELATIVE TO THAT OF THE STANDARD DE IN 30 RUNS ARE SHOWN

	DE	DESFC ($C = 2$)	DESFC ($C = 3$)	DESFC ($C = 4$)	DESFC ($C = 5$)
f_1	160492.4 ± 1435.2 (1.000)	**117206.1 ± 1564.7** (0.730)	120674.8 ± 1831.4 (0.752)	122346.5 ± 1755.9 (0.762)	124235.9 ± 1194.0 (0.774)
f_2	226925.5 ± 1780.4 (1.000)	**170303.5 ± 2038.4** (0.750)	174018.3 ± 2069.4 (0.767)	177648.0 ± 1673.2 (0.783)	179780.8 ± 1981.5 (0.792)
f_3	1383166.8 ± 17090.3 (1.000)	**810832.5 ± 13592.4** (0.586)	838104.6 ± 16195.6 (0.606)	847113.2 ± 13832.2 (0.612)	861445.0 ± 17661.7 (0.623)
f_4	1435549.4 ± 12435.7 (1.000)	**845085.4 ± 12057.4** (0.589)	884353.9 ± 14714.8 (0.616)	907182.5 ± 14305.0 (0.632)	921243.2 ± 13765.9 (0.642)
f_5	505521.2 ± 6922.3 (1.000)	**375997.1 ± 11190.8** (0.744)	417049.6 ± 10243.3 (0.825)	446234.7 ± 9937.2 (0.883)	464278.1 ± 7319.0 (0.918)
f_6	64055.6 ± 1613.6 (1.000)	**47752.3 ± 1296.9** (0.745)	49372.5 ± 1214.7 (0.771)	49977.7 ± 1297.2 (0.780)	50268.7 ± 1612.4 (0.785)
f_7	798441.2 ± 139293.5 (1.000)	**563133.9 ± 107809.7** (0.705)	572408.3 ± 115549.5 (0.717)	553073.6 ± 95603.7 (0.693)	598517.6 ± 92705.4 (0.750)
f_8	192272.8 ± 2857.9 (1.000)	**144776.4 ± 3354.0** (0.753)	148036.9 ± 3334.3 (0.770)	150478.6 ± 2474.9 (0.783)	151358.5 ± 3162.7 (0.787)
f_9	349435.7 ± 8654.0 (1.000)	**282471.5 ± 10327.1** (0.808)	286350.3 ± 6992.8 (0.819)	293521.2 ± 10027.9 (0.840)	292191.9 ± 7366.3 (0.836)
f_{10}	239050.7 ± 1933.8 (1.000)	**174369.5 ± 2165.9** (0.729)	179843.3 ± 1916.2 (0.752)	183511.6 ± 2097.5 (0.768)	185127.6 ± 1934.2 (0.774)
f_{11}	171085.8 ± 5818.7 (1.000)	**121418.5 ± 3495.4** (0.710)	127356.7 ± 5213.6 (0.744)	128429.4 ± 5079.4 (0.751)	131122.6 ± 5681.8 (0.766)
f_{12}	142630.8 ± 1479.8 (1.000)	**106608.6 ± 2220.4** (0.747)	109387.1 ± 2468.9 (0.767)	110834.6 ± 1590.7 (0.777)	112822.2 ± 2121.4 (0.791)
f_{13}	153476.0 ± 1643.0 (1.000)	**113441.0 ± 1508.6** (0.739)	116932.2 ± 2255.1 (0.762)	118936.9 ± 1966.8 (0.775)	119801.5 ± 1684.0 (0.781)

DESFC can solve all problems in all runs successfully. DESFC outperformed the standard DE in all problems and in all settings of C. The value $C = 2$ showed the best results. DESFC with $C = 2$ can solve the problems f_3 and f_4 within 60% FEs compared with the standard DE, solve the problems f_7 and f_{11} in about 70% FEs, and solve the other problems in about 75% FEs except for f_9 in about 80% FEs.

It is shown that the proposed method can solve 4 problems very fast and 9 problems fast. Thus, it is thought that the method is effective to various problems.

To determine the significance of the proposed method, statistical analysis was performed using one-sided Welch's t-test for the mean FEs of the standard DE and that of DESFC with C being 1, 2, 3, 4 and 5. It is thought that the proposed method is significantly better than the standard DE because p-values in all parameter settings and all functions are less than 0.001.

Figures 5 to 17 show the change of best objective value found over the number of FEs within 200,000 evaluations. Apparently, the proposed method can find better objective values faster than the standard DE in all problems.

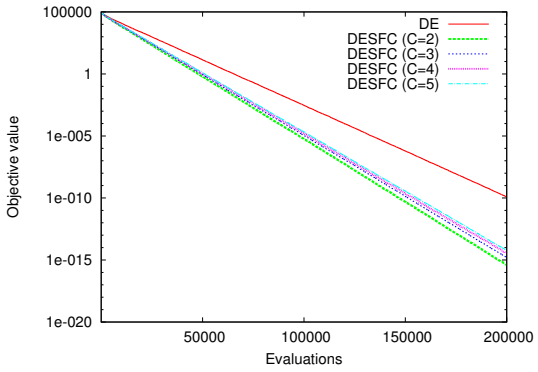

Fig. 5. The graph of f_1

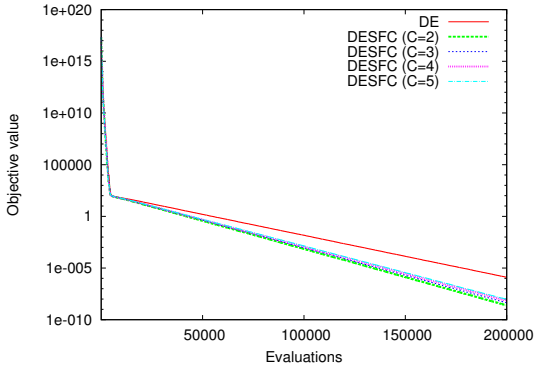

Fig. 6. The graph of f_2

VII. CONCLUSION

Differential evolution is known as a simple, efficient and robust search algorithm that can solve nonlinear optimization problems. In this study, we proposed the DE with speciation using fuzzy clustering and partition entropy to improve the efficiency and also robustness of DE. Especially, it is a quite new approach to detect the distribution of search points using FCM and to control an algorithm parameter. It was shown that DESFC can reduce the number of function evaluations for

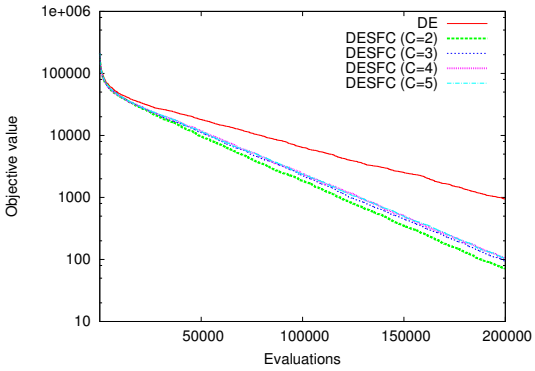

Fig. 7. The graph of f_3

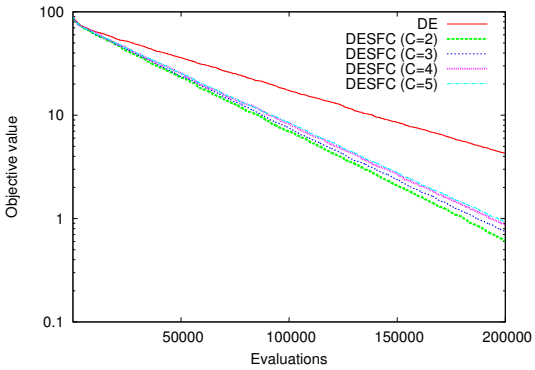

Fig. 8. The graph of f_4

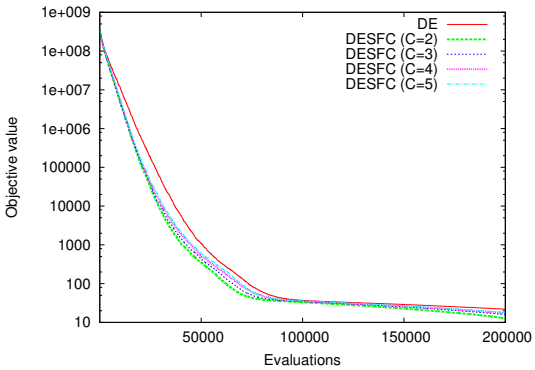

Fig. 9. The graph of f_5

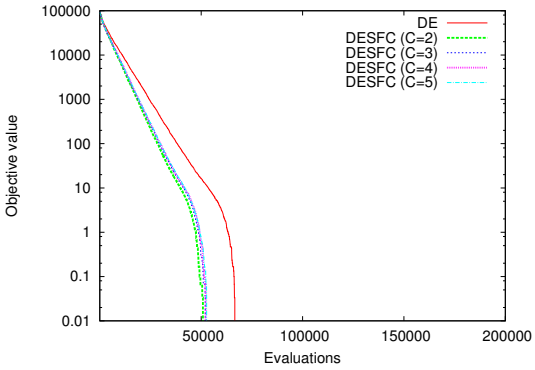

Fig. 10. The graph of f_6

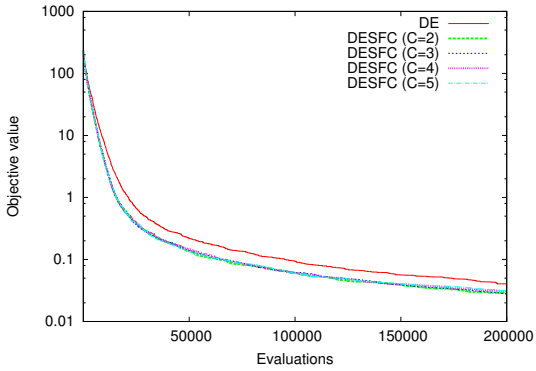

Fig. 11. The graph of f_7

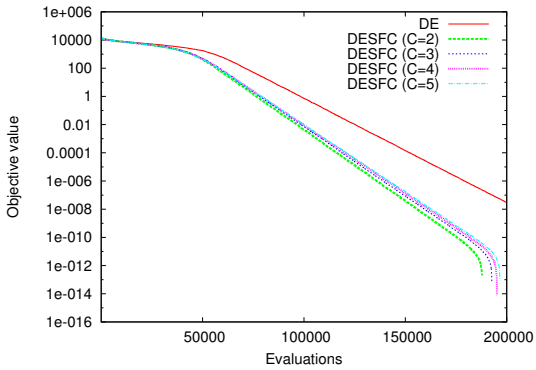

Fig. 12. The graph of f_8

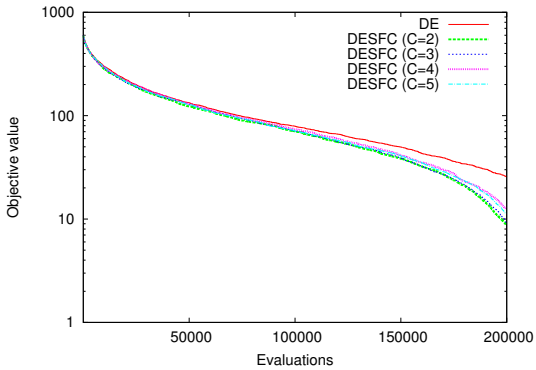

Fig. 13. The graph of f_9

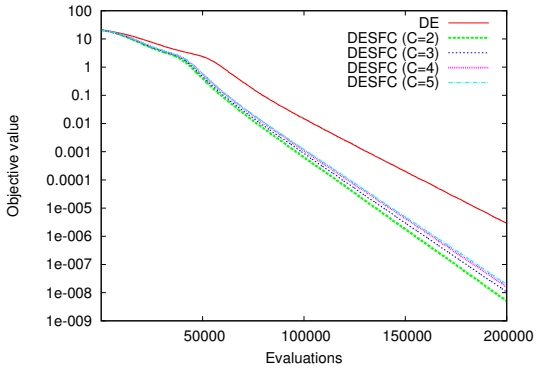

Fig. 14. The graph of f_{10}

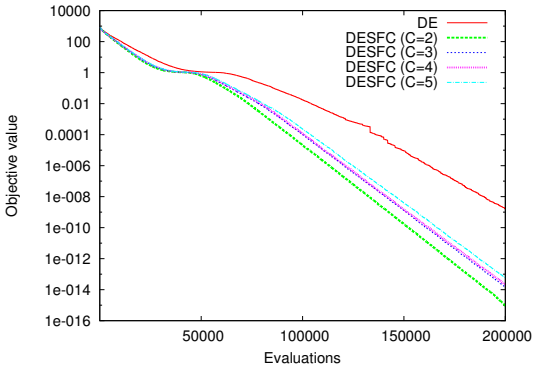

Fig. 15. The graph of f_{11}

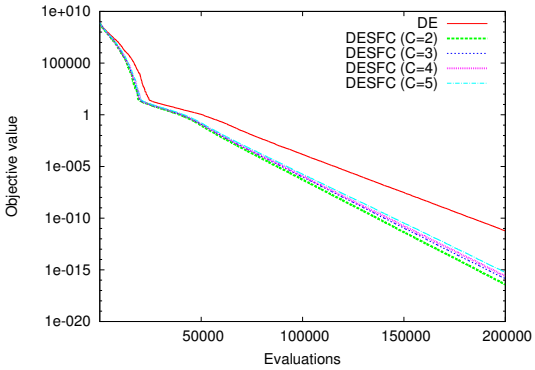

Fig. 16. The graph of f_{12}

finding near optimal solutions more than 25% in 11 problems out of the 13 problems. Thus, it is thought that DESFC is a very efficient optimization algorithm compared with standard DEs.

The effect of the degree of fuzziness m is not studied in this paper. It is thought that a proper value of the degree enhances the ability to discriminate a biased distribution from the uniform distribution. We plan to analyze the effect of the degree.

ACKNOWLEDGMENT

This research is supported in part by Grant-in-Aid for Scientific Research (C) (No. 20500138,22510166) of Japan society for the promotion of science.

REFERENCES

[1] R. Storn and K. Price, "Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[2] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005.

[3] U. K. Chakraborty, Ed., *Advances in Differential Evolution*. Springer, 2008.

[4] M. Shibasaka, A. Hara, T. Ichimura, and T. Takahama, "Species-based differential evolution with switching search strategies for multimodal function optimization," in *Proc. of the 2007 IEEE Congress on Evolutionary Computation*, Sep. 2007, pp. 1183 –1190.

[5] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Comput.*, vol. 9, no. 6, pp. 448–462, 2005.

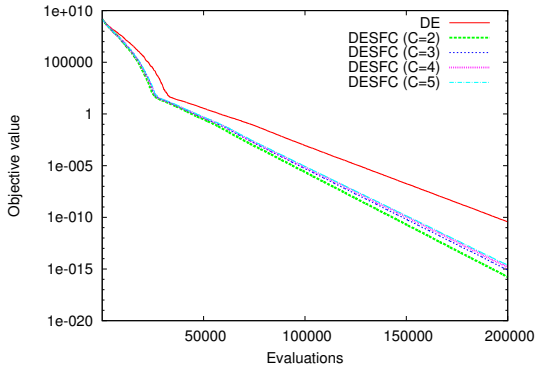

Fig. 17. The graph of f_{13}

[6] W. Gong, Z. Cai, C. X. Ling, and J. Du, "Hybrid differential evolution based on fuzzy c-means clustering," in *Proc. of Genetic and Evolutionary Computation Conference 2009*, 2009, pp. 523–530.

[7] A. E. Imrani, A. Bouroumi, H. Z. E. Abidine, M. Limouri, and A. Essaïd, "A fuzzy clustering-based niching approach to multimodal function optimization," *Journal of Cognitive Systems Research*, vol. 1, no. 2, pp. 119–133, 2000.

[8] J.-P. Li, X.-D. Li, and A. Wood, "Species based evolutionary algorithms for multimodal optimization: A brief review," in *Proc. of the 2010 IEEE Congress on Evolutionary Computation*, Jul. 2010, pp. 1–8.

[9] X. Li, "Efficient differential evolution using speciation for multimodal function optimization," in *Proc. of the 2005 conference on Genetic and evolutionary computation*, 2005, pp. 873–880.

[10] B. Qu and P. Suganthan, "Modified species-based differential evolution with self-adaptive radius for multi-modal optimization," in *Proc. of the 2010 International Conference on Computational Problem-Solving*, Dec. 2010, pp. 326 –331.

[11] A. Passaro and A. Starita, "Particle swarm optimization for multimodal functions: a clustering approach," *Journal of Artificial Evolution and Applications*, vol. 2008, pp. 1–15, January 2008.

[12] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1981.

[13] T. Takahama, S. Sakai, and N. Iwane, "Solving nonlinear constrained optimization problems by the ε constrained differential evolution," in *Proc. of the 2006 IEEE Conference on Systems, Man, and Cybernetics*, Oct. 2006, pp. 2322–2327.

[14] T. Takahama and S. Sakai, "Reducing function evaluations in differential evolution using rough approximation-based comparison," in *Proc. of the 2008 IEEE Congress on Evolutionary Computation*, Jun. 2008, pp. 2307–2314.

[15] S. Kukkonen and J. Lampinen, "Constrained real-parameter optimization with generalized differential evolution," in *Proc. of the 2006 IEEE Congress on Evolutionary Computation*. Vancouver, BC, Canada: IEEE Press, 16-21 July 2006, pp. 207–214.

[16] T. Takahama and S. Sakai, "Solving difficult constrained optimization problems by the ε constrained differential evolution with gradient-based mutation," in *Constraint-Handling in Evolutionary Optimization*, E. Mezura-Montes, Ed. Springer-Verlag, 2009, pp. 51–72.

[17] Z. Jingqiao and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, Oct. 2009.

[18] Y.-W. Shang and Y.-H. Qiu, "A note on the extended rosenbrock function," *Evolutionary Computation*, vol. 14, no. 1, pp. 119–126, 2006.

[19] X. Yao, Y. Liu, , and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 82–102, 1999.

[20] X. Yao, Y. Liu, K.-H. Liang, and G. Lin, "Fast evolutionary algorithms," in *Advances in evolutionary computing: theory and applications*, A. Ghosh and S. Tsutsui, Eds. New York, NY, USA: Springer-Verlag New York, Inc., 2003, pp. 45–94.

[21] T. Takahama and S. Sakai, "Fast and stable constrained optimization by the ε constrained differential evolution," *Pacific Journal of Optimization*, vol. 5, no. 2, pp. 261–282, May 2009.