An Adaptive Differential Evolution Considering Correlation of Two Algorithm Parameters

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 an evolutionary algorithm. DE has been successfully applied to optimization problems including non-linear, non-differentiable, non-convex and multimodal functions. The performance of DE is affected by algorithm parameters such as a scaling factor F and a crossover rate CR. Many studies have been done to control the parameters adaptively. One of the most successful studies on parameter control is JADE. In JADE, two parameter values are generated according to a probability density function which is learned by the parameter values in success cases, where the child is better than the parent. The values of two parameters are independently generated. In this study, we propose a new method where the values of two parameters are generated dependently using the correlation coefficient. In each generation of DE, the pairs of two parameter values in the success cases are stored and the correlation coefficient is obtained. The parameter F is generated according to Cauchy distribution. The parameter CR is generated according to normal distribution of which mean is modified using the generated value of F and the correlation coefficient. The effect of the proposed method is shown by solving thirteen benchmark problems.**

Keywords—differential evolution; adaptive parameter control; probability density function

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 an EA proposed by Storn and Price [1]. DE has been successfully applied to optimization problems including non-linear, non-differentiable, non-convex and multimodal functions [2]–[4]. It has been shown that DE is a very fast and robust algorithm.

The performance of DE is affected by algorithm parameters such as a scaling factor F, a crossover rate CR and population size, and by mutation strategies such as a rand strategy and a best strategy. Many studies have been done to control the parameters and the strategies. One of the most successful studies on controlling the parameters is JADE(adaptive DE with optional external archive) [5]. In JADE, the parameter values of F and CR are generated according to a probability density function and a child is created from the parent using the generated values. The values in success cases, where the child is better than the parent, are used to learn the probability density function. The values of F are generated according to Cauchy distribution of which location parameter is learned and scale parameter is fixed. The values of CR are generated according to normal distribution of which mean is learned and standard deviation is fixed. The values of F and the values of CR are independently generated.

Therefore, it is thought that JADE is a machine learning system which learns the probability density function of the parameters from training data, which are the parameter values in the success cases. In the system, there are some methods to learn more accurate probability density function:

(a) Considering dependency of the algorithm parameters in the probability density function instead of supposing independence of them.
(b) Learning extra parameters such as standard deviation in normal distribution, scale parameter in Cauchy distribution, and so on.
(c) Learning not only a single distribution function but also a mixture distribution of plural distribution functions.

Note that an accurate probability density function may not lead a good result because the success cases depend not only on the parameter values but also on the parent, selected difference vectors, and so on.

In this study, we propose a new method where parameter values of F and CR are generated not independently but dependently using the correlation coefficient. In each generation of DE, the pairs of parameter values (F, CR) in the success cases are stored and the correlation coefficient is obtained. The parameter values are generated according to a probability function as follows: F is generated according to Cauchy distribution. The mean value of normal distribution function for CR is modified using the generated value of F and the correlation coefficient. A value of CR is generated according to the modified normal distribution. The effect of the proposed method is shown by solving thirteen benchmark problems including unimodal problems and multimodal problems.

In Section II, related works are briefly reviewed. DE and JADE are explained in Section III and IV, respectively. JADE using correlation is proposed in Section V. In Section VI, experimental results on benchmark problems are shown. Finally, conclusions are described in Section VII.

II. Related Works

The methods of controlling algorithm parameters can be classified into some categories as follows:

(1) selection-based control: Strategies and parameter values are selected regardless of current search state. CoDE(composite DE) [6] generates three trial vectors using three strategies with randomly selected parameter values from parameter candidate sets and the best trial vector will head to the survivor selection.

(2) observation-based control: The current search state is observed, proper parameter values are inferred according to the observation, and parameters and/or strategies are dynamically controlled. FADE(Fuzzy Adaptive DE) [7] observes the movement of search points and the change of function values between successive generations, and controls F and CR. DESFC(DE with Speciation and Fuzzy Clustering) [8] adopts fuzzy clustering, observes partition entropy of search points, and controls CR and the mutation strategies between the rand and the species-best strategy. LMDE(DE with detecting Landscape Modality) [9], [10] detects the landscape modality such as unimodal or multimodal using the change of the objective values at sampling points which are equally spaced along a line. If the landscape is unimodal, greedy parameter settings for local search are selected. Otherwise, parameter settings for global search are selected.

(3) success-based control: It is recognized as a success case when a better search point than the parent is generated. The parameters and/or strategies are adjusted so that the values in the success cases are frequently used. It is thought that the self-adaptation, where parameters are contained in individuals and are evolved by applying evolutionary operators to the parameters, is included in this category. DESAP(DE with Self-Adapting Populations) [11] controls F, CR and N self-adaptively. SaDE(Self-adaptive DE) [12] controls the selection probability of the mutation strategies according to the success rates and controls the mean value of CR for each strategy according to the mean value in success case. jDE(self-adaptive DE algorithm) [13] controls F and CR self-adaptively. JADE [5] and MDE_pBX(modified DE with p-best crossover) [14] control the mean or power mean values of F and CR according to the mean values in success cases.

In the category (1), useful knowledge to improve the search efficiency is ignored. In the category (2), it is difficult to select proper type of observation which is independent of the optimization problem and its scale. In the category (3), when a new good search point is found near the parent, parameters are adjusted to the direction of convergence. In problems with ridge landscape or multimodal landscape, where good search points exist in small region, parameters are tuned for small success and big success will be missed. Thus, search process would be trapped at a local optimal solution. In JADE, as for a mean value of F a weighted mean value by the value of F is used to generate larger F than a usual mean value and it is succeeded to reduce the problem of the convergence.

In this study, we propose to improve JADE in the category (3) by considering the correlation between F and CR in the success cases. The mean values and a correlation coefficient in success cases are used to control the probability density function of F and CR.

III. Optimization by Differential Evolution

A. Optimization Problems

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

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

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. Differential Evolution

DE is a stochastic direct search method using a population or multiple search points. In DE, initial individuals are randomly generated within 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 $\boldsymbol{x}^{r1}$, $\boldsymbol{x}^{r2}$ and $\boldsymbol{x}^{r3}$ are chosen randomly from the population without overlapping $\boldsymbol{x}^i$ and each other. A new vector, or a mutant vector $\boldsymbol{m}$ is generated by the base vector $\boldsymbol{x}^{r1}$ and the difference vector $\boldsymbol{x}^{r2} - \boldsymbol{x}^{r3}$, where F is the scaling factor.

$$\boldsymbol{m} = \boldsymbol{x}^{r1} + F(\boldsymbol{x}^{r2} - \boldsymbol{x}^{r3}) \tag{2}$$

"*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. 1 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 $\boldsymbol{m}$, where CR is a crossover rate.

```
binomial crossover DE/·/·/bin
  $j_{rand}$=randint(1,$D$);
  for($k$=1; $k \le D$; $k$++) {
    if($k == j_{rand}$ || $u(0,1) < CR$) $x^{\rm child}_k$=$m_k$;
    else $x^{\rm child}_k$=$x^i_k$;
  }
exponential crossover DE/·/·/exp
  $k$=1; $j$=randint(1,$D$);
  do {
      $x^{\rm child}_j$=$m_j$;
      $k$=$k$+1; $j$=$(j+1)\%D$;
  } while($k \le D$ && $u(0,1) < CR$);
  while($k \le D$) {
      $x^{\rm child}_j$=$x^i_j$;
      $k$=$k$+1; $j$=$(j+1)\%D$;
  }
```

Fig. 1. 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 $[l, r]$.

C. The Algorithm of Differential Evolution

The algorithm of DE is as follows:

Step1 Initialization of a population. Initial NP individuals $P = \{\boldsymbol{x}^i | i = 1, 2, \cdots, NP\}$ are generated randomly in 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 target vector (parent). If all individuals are selected, go to Step4. A mutant vector $\boldsymbol{m}$ is generated according to Eq. (2). A trial vector (child) is generated from the parent $\boldsymbol{x}^i$ and the mutant vector $\boldsymbol{m}$ using a crossover operation shown in Fig. 1. If the child is better than 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. The population P is formed by the survivors. Go back to Step2.

Fig. 2 shows a pseudo-code of DE/rand/1.

```
DE/rand/1()
{
// Initialize a population
 $P$=$N$ individuals generated randomly in $\mathcal{S}$;
 for($t$=1; $FE \le FE_{max}$; $t$++) {
  for($i$=1; $i \le N$; $i$++) {
// DE operation
      $\boldsymbol{x}^{p1}$=Randomly selected from $P$($p1 \ne i$);
      $\boldsymbol{x}^{p2}$=Randomly selected from $P$($p2 \notin \{i, p1\}$);
      $\boldsymbol{x}^{p3}$=Randomly selected from $P$($p3 \notin \{i, p1, p2\}$);
      $\boldsymbol{m}$=$\boldsymbol{x}^{p1}$+$F(\boldsymbol{x}^{p2} - \boldsymbol{x}^{p3})$;
      $\boldsymbol{x}^{\rm child}$=trial vector is generated from
      $\boldsymbol{x}^i$ and $\boldsymbol{m}$ by a crossover operation;
// Survivor selection
      if$\big(f(\boldsymbol{x}^{\rm child}) < f(\boldsymbol{x}^i)\big)$ $\boldsymbol{z}^i$=$\boldsymbol{x}^{\rm child}$;
      else                $\boldsymbol{z}^i$=$\boldsymbol{x}^i$;
      $FE$=$FE$+1;
  }
  $P$=$\{\boldsymbol{z}^i,\ i = 1, 2, \cdots, N\}$;
 }
}
```

Fig. 2. The pseudo-code of DE, FE is the number of function evaluations.

IV. JADE

In JADE, the mean value of the scaling factor μ_F and the mean value of the crossover rate μ_{CR} are learned to define a probability density function, where initial values are μ_F=μ_{CR}=0.5. The scaling factor F_i and the crossover rate CR_i for each individual $\boldsymbol{x}^i$ are independently generated according to the following equations:

$$F_i \sim C(\mu_F, \sigma_F) \quad (3)$$
$$CR_i \sim N(\mu_{CR}, \sigma^2_{CR}) \quad (4)$$

where $C(\mu_F, \sigma_F)$ is a random variable according to Cauchy distribution with a location parameter μ_F and a scale parameter σ_F=0.1, $N(\mu_{CR}, \sigma^2_{CR})$ is a random variable according to normal distribution of a mean μ_{CR} and a standard deviation σ_{CR}=0.1. CR_i is truncated to $[0, 1]$ and F_i is truncated to be 1 if $F_i \ge 1$ or regenerated if $F_i \le 0$. The location μ_F and the mean μ_{CR} are updated as follows:

$$\mu_F = (1-c)\mu_F + cS_{F^2}/S_F \quad (5)$$
$$\mu_{CR} = (1-c)\mu_{CR} + cS_{CR}/S_N \quad (6)$$

where S_N is the number of success cases, S_F, S_{F^2} and S_{CR} are the sum of F, F^2 and CR in success cases, respectively. A constant c is a weight of update in (0,1] and the recommended value is 0.1.

JADE adopts a strategy called "current-to-pbest" where an intermediate point between a target vector and a randomly selected point from top individuals is used as a base vector. A mutation vector is generated as follows:

$$\boldsymbol{m} = \boldsymbol{x}^i + F_i(\boldsymbol{x}^{pbest} - \boldsymbol{x}^i) + F_i(\boldsymbol{x}^{p2} - \boldsymbol{x}^{p3}) \quad (7)$$

where $\boldsymbol{x}^{pbest}$ is a randomly selected individual from the top $100p\%$ individuals.

V. Proposed Algorithm

A. Conditional Normal Distribution

In general, the multivariate normal distribution of a k-dimensional random vector $\boldsymbol{u} = (U_1, U_2, \cdots, U_k)$ can be written as follows:

$$\boldsymbol{u} \sim N(\boldsymbol{\mu}, \Sigma) \quad (8)$$
$$f(\boldsymbol{u}) = \frac{1}{\sqrt{(2\pi)^k|\Sigma|}} \exp\left(-\frac{1}{2}(\boldsymbol{u}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{u}-\boldsymbol{\mu})\right) \quad (9)$$

where $\boldsymbol{\mu}$ is the mean vector, Σ is the covariance matrix, f is the probability density function, $|\Sigma|$ is the determinant of Σ, and Σ^{-1} is the inverse matrix of Σ. In bivariate case where k=2, the covariance matrix is defined as follows:

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix} \quad (10)$$

where ρ is the correlation coefficient between U_1 and U_2. In this case, the conditional distribution of U_2 given U_1 is as follows:

$$U_2|U_1 = u_1 \ \sim \ N(\mu_2 + \rho\frac{\sigma_2}{\sigma_1}(u_1 - \mu_1), (1-\rho^2)\sigma_2^2) \quad (11)$$

B. Conditional Distribution between Algorithm Parameters

In this study, the bivariate distribution between algorithm parameters F and CR is considered. Suppose that the parameter values in success cases are given by $S = \{(F_i, CR_i)\}$. The correlation ρ between F and CR is given as follows:

$$\rho \ = \ \frac{\sum_{(F_i,CR_i)\in S}(F_i - \mu_F)(CR_i - \mu_{CR})}{\sigma_F \sigma_{CR}} \quad (12)$$

where μ_F and μ_{CR} are the means of F and CR, and σ_F and σ_{CR} are the standard deviations of F and CR, respectively. Eq.(11) can be described using F and CR as follows:

$$CR|F = F_i \sim N(\mu_{CR} + \rho\frac{\sigma_{CR}}{\sigma_F}(F_i - \mu_F), (1-\rho^2)\sigma_{CR}^2) (13)$$

However, since F is not generated according to normal distribution but according to Cauchy distribution, Eq.(13) needs to be modified. In this paper, the following modification is adopted:

- The deviation $F_i - \mu_F$ in Cauchy distribution often larger than the deviation in normal distribution. In order to avoid too large change of the mean value μ_{CR}, when $\frac{\sigma_{CR}}{\sigma_F}(F_i - \mu_F)$ is out of the interval $[-\sigma_{CR}, \sigma_{CR}]$, the value is selected from $[-1.5\sigma_{CR}, -\sigma_{CR}]$ or $[\sigma_{CR}, 1.5\sigma_{CR}]$ randomly.
- The fixed standard deviation σ_{CR} is used instead of $\sqrt{(1-\rho^2)}\sigma_{CR}$ to keep proper deviation.
- If the number of success cases are very small, the value of ρ often becomes about -1 or 1. In this paper, ρ is updated only when the number of success cases is greater than or equal to 5.

The algorithm of modified JADE using correlation (Correlation based Adaptive DE, CADE) can be described as follows:

Step0 Parameter setup. The mean value of scaling factor $\mu_F = 0.5$ and the mean value of crossover rate $\mu_{CR} = 0.5$. The standard deviations σ_F=0.1 and σ_{CR}=0.1. The list of success cases S is made empty.

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

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

Step3 DE operation with adaptive parameters. The scaling factor F_i is generated according to Cauchy distribution. The crossover rate CR_i is generated according to normal distribution using the correlation. DE/current-to-pbest/1/bin operation with F_i and CR_i is executed and a new child $\boldsymbol{x}^{\text{child}}$ is generated. If the new one is better than the parent, the operation is treated as a success case and the child becomes a survivor. The successful combination of parameter values (F_i, CR_i) is added to success cases S. Otherwise, the parent $\boldsymbol{x}^i$ becomes a survivor.

Step4 Update of adaptive parameters. The mean of the scaling factor μ_F and the mean of crossover rate μ_{CR} are updated using S. If there are enough success cases, ρ is updated using S.

Step5 Go back to Step2.

```
CADE/current-to-pbest/1/bin()
{
  $\mu_F$=$\mu_{CR}$=0.5; $\sigma_F = \sigma_{CR}$=0.1; $S$=$\phi$;
+ $\rho = 0$;
// Initialize a population
  $P$=$NP$ individuals generated randomly in $\mathcal{S}$;
  $FE$=$FE$+$N$;
  for($t$=1; $FE \leq FE_{max}$; $t$++) {
    for($i$=1; $i \leq N$; $i$++) {
      do {
        $F_i$=$\mu_F + C(0, \sigma_F)$;
      } while($F_i \leq 0$);
      if($F_i > 1$) $F_i = 1$;
+     $\delta$=$\frac{\sigma_{CR}}{\sigma_F}(F_i - \mu_F)$;
+     if($\delta < -\sigma_{CR}$) $\delta$=$-\sigma_{CR}u(1, 1.5)$;
+     else if($\delta > \sigma_{CR}$) $\delta$=$\sigma_{CR}u(1, 1.5)$;
+     $CR_i$=$\mu_{CR} + \rho\delta + N(0, \sigma_{CR})$;
      if($CR_i < 0$) $CR_i$=0;
      else if($CR_i > 1$) $CR_i$=1;
      $\boldsymbol{x}^{pbest}$=Randomly selected from top 100$p$% in $P$;
      $\boldsymbol{x}^{r1}$=Randomly selected from $P$($r1 \notin \{i\}$);
      $\boldsymbol{x}^{r2}$=Randomly selected from $P$($r2 \notin \{i, r1\}$);
      $\boldsymbol{x}'$=$\boldsymbol{x}^i$+$F_i(\boldsymbol{x}^{pbest} - \boldsymbol{x}^i)$+$F_i(\boldsymbol{x}^{r1} - \boldsymbol{x}^{r2})$;
      $\boldsymbol{x}^{child}$=trial vector is generated from
          $\boldsymbol{x}^i$ and $\boldsymbol{x}'$ by binomial crossover;
      $FE$=$FE$+1;
// Survivor selection
      if($f(\boldsymbol{x}^{child}) < f(\boldsymbol{z})$) {
        $\boldsymbol{z}^i$=$\boldsymbol{x}^{child}$;
        $S$=$S \cup \{(F_i, CR_i)\}$; // success cases are stored
      }
      else $\boldsymbol{z}^i$=$\boldsymbol{x}^i$;
    }
    $P$=$\{\boldsymbol{z}^i\}$;
    if($|S| > 0$) {
      $\mu_F$=$(1-c)\mu_F + c\sum_{F_i \in S} F_i^2 / \sum_{F_i \in S} F$;
      $\mu_{CR}$=$(1-c)\mu_{CR} + c\sum_{CR_i \in S} CR_i / |S|$;
+     if($|S| \geq 5$) {
+       $\rho_0$=correlation between $F$ and $CR$ in $S$;
+       $\rho$=$(1-c)\rho + c\rho_0$;
+     }
    }
  }
}
```

Fig. 3. The pseudo-code of CADE

Fig. 3 shows the pseudo-code of CADE. The lines which start with '+' show the modification to JADE.

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 [5]. 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 [15]. The 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 [16].

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 [17]

Test functions	Bound constraints
$f_1(\boldsymbol{x}) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$
$f_2(\boldsymbol{x}) = \sum_{i=1}^{D} \lvert x_i\rvert + \prod_{i=1}^{D} \lvert x_i\rvert$	$[-10, 10]^D$
$f_3(\boldsymbol{x}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j\right)^2$	$[-100, 100]^D$
$f_4(\boldsymbol{x}) = \max_i\{\lvert x_i\rvert\}$	$[-100, 100]^D$
$f_5(\boldsymbol{x}) = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	$[-30, 30]^D$
$f_6(\boldsymbol{x}) = \sum_{i=1}^{D} \lfloor x_i + 0.5\rfloor^2$	$[-100, 100]^D$
$f_7(\boldsymbol{x}) = \sum_{i=1}^{D} ix_i^4 + rand[0, 1)$	$[-1.28, 1.28]^D$
$f_8(\boldsymbol{x}) = \sum_{i=1}^{D} -x_i \sin\sqrt{\lvert x_i\rvert} + D \cdot 418.9828872724 3369$	$[-500, 500]^D$
$f_9(\boldsymbol{x}) = \sum_{i=1}^{D} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	$[-5.12, 5.12]^D$
$f_{10}(\boldsymbol{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]^D$
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^D$
$f_{12}(x) = \frac{\pi}{D}[10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2 \{1 + 10\sin^2(\pi y_{i+1})\} + (y_D - 1)^2] + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ where $y_i = 1 + \frac{1}{4}(x_i + 1)$ and $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-50, 50]^D$
$f_{13}(x) = 0.1[\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2 \{1 + \sin^2(3\pi x_{i+1})\} + (x_D - 1)^2 \{1 + \sin^2(2\pi x_D)\}] + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	$[-50, 50]^D$

Independent 50 runs are performed for 13 problems. The dimension of problems is 30 (D=30). Each run stops when the number of function evaluations (FEs) exceeds the maximum number of evaluations $FE_{\max}$.

B. Experimental Results on the Proposed Method

The control parameters for CADE are same as JADE: The population size NP=100, the initial means μ_F=μ_{CR}=0.5, the standard deviations are fixed as σ_F=σ_{CR}=0.1 and c=0.1. The initial correlation coefficient ρ=0.

Table II compares CADE with other methods including JADE, jDE, SaDE, DE/rand/1/bin and PSO. Results except for CADE are taken from [5]. The best result among algorithms is highlighted using bold face fonts.

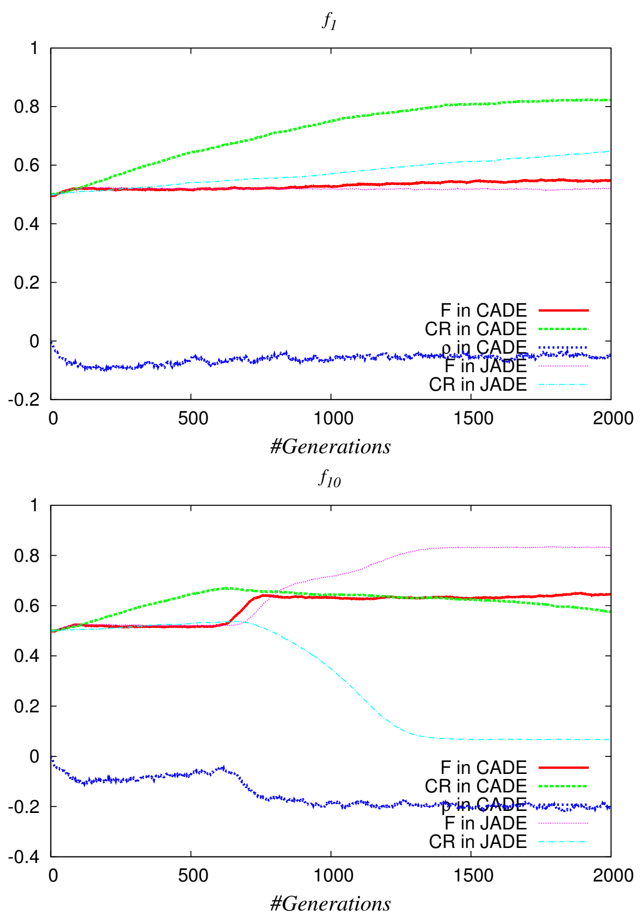

Fig. 4. The graph of parameter values in CADE and JADE

CADE attained the best results among all methods in 10 problems f_1, f_2, f_5, f_6, f_7, f_9, f_{10}, f_{11}, f_{12} and f_{13}. Also, CADE attained the best final results in 3 problems f_5, f_{10} and f_{13}, where the 8 final results are shown in the bottom row for functions f_5, f_6 and from f_8 to f_{13}. CADE outperformed JADE without archive in 12 problems and in 3 problems for final results. CADE outperformed JADE with archive in 11 problems and in 4 problems for final results. CADE outperformed jDE in 11 problems, SaDE and PSO in 12 problems, and DE/rand/1/bin in all problems. Thus, it is thought that CADE is effective to various problems.

Figure 4 show the change of parameter values F and CR in CADE and JADE for a unimodal function f_1 and a multimodal function f_{10}. Also, the change of ρ in CADE is shown. In functions f_1, f_2 and f_{13}, although the values of F are similar in CADE and JADE, the values of CR in CADE are larger than those of JADE. It is thought that large CR amplifies the convergence speed of CADE. In functions f_3 to f_9 and f_{11}, the difference between CADE and JADE is not large. In multimodal function f_{10} and f_{12}, CADE adopted smaller F and larger CR than JADE and attained better results than JADE.

TABLE II. EXPERIMENTAL RESULTS ON CADE AND OTHER DEs. MEAN VALUES AND STANDARD DEVIATIONS IN 50 RUNS ARE SHOWN

	FE_{max}	CADE	JADE w/o archive	JADE with archive	jDE	SaDE	DE/rand/1/bin	PSO
f_1	150,000	**1.29e-70 (8.24e-70)**	1.8e-60 (8.4e-60)	1.3e-54 (9.2e-54)	2.5e-28 (3.5e-28)	4.5e-20 (6.9e-20)	9.8e-14 (8.4e-14)	9.6e-42 (2.7e-41)
f_2	200,000	**5.05e-50 (1.37e-49)**	1.8e-25 (8.8e-25)	3.9e-22 (2.7e-21)	1.5e-23 (1.0e-23)	1.9e-14 (1.05e-14)	1.6e-09 (1.1e-09)	9.3e-21 (6.3e-20)
f_3	500,000	2.26e-62 (1.20e-61)	5.7e-61 (2.7e-60)	**6.0e-87 (1.9e-86)**	5.2e-14 (1.1e-13)	9.0e-37 (5.43e-36)	6.6e-11 (8.8e-11)	2.5e-19 (3.9e-19)
f_4	500,000	1.25e-07 (9.88e-08)	8.2e-24 (4.0e-23)	**4.3e-66 (1.2e-65)**	1.4e-15 (1.0e-15)	7.4e-11 (1.82e-10)	4.2e-01 (1.1e+00)	4.4e-14 (9.3e-14)
f_5	300,000	**1.62e-30 (5.21e-30)**	8.0e-02 (5.6e-01)	3.2e-01 (1.1e+00)	1.3e+01 (1.4e+01)	2.1e+01 (7.8e+00)	2.1e+00 (1.5e+00)	2.5e+01 (3.2e+01)
	2,000,000	**1.62e-30 (5.21e-30)**	8.0e-02 (5.6e-01)	3.2e-01 (1.1e+00)	8.0e-02 (5.6e-01)	1.8e+01 (6.7e+00)	8.0e-02 (5.6e-01)	1.7e+01 (2.3e+01)
f_6	10,000	**2.4e+00 (1.58e+00)**	2.9e+00 (1.2e+00)	5.6e+00 (1.6e+00)	1.0e+03 (2.2e+02)	9.3e+02 (1.8e+02)	4.7e+03 (1.1e+03)	4.5e+01 (2.4e+01)
	150,000	0.0e+00 (0.0e+00)	0.0e+00 (0.0e+00)	0.0e+00 (0.0e+00)	0.0e+00 (0.0e+00)	0.0e+00 (0.0e+00)	0.0e+00 (0.0e+00)	8.0e-02 (2.7e-01)
f_7	300,000	**6.33e-04 (2.30e-04)**	6.4e-04 (2.5e-04)	6.8e-04 (2.5e-04)	3.3e-03 (8.5e-04)	4.8e-03 (1.2e-03)	4.7e-03 (1.2e-03)	2.5e-03 (1.4e-03)
f_8	100,000	3.52e-06 (3.35e-06)	3.3e-05 (2.3e-05)	7.1e+00 (2.8e+01)	**7.9e-11 (1.3e-10)**	4.7e+00 (3.3e+01)	5.9e+03 (1.1e+03)	2.4e+03 (6.7e+02)
	900,000	0.0e+00 (0.0e+00)	0.0e+00 (0.0e+00)	7.1e+00 (2.8e+01)	0.0e+00 (0.0e+00)	4.7e+00 (3.3e+01)	5.7e+01 (7.6e+01)	2.4e+03 (6.7e+02)
f_9	100,000	**9.94e-05 (6.20e-05)**	1.0e-04 (6.0e-05)	1.4e-04 (6.5e-05)	1.5e-04 (2.0e-04)	1.2e-03 (6.5e-04)	1.8e+02 (1.3e+01)	5.2e+01 (1.6e+01)
	500,000	0.0e+00 (0.0e+00)	0.0e+00 (0.0e+00)	0.0e+00 (0.0e+00)	0.0e+00 (0.0e+00)	0.0e+00 (0.0e+00)	7.1e+01 (2.1e+01)	5.2e+01 (1.6e+01)
f_{10}	50,000	**1.18e-10 (8.39e-11)**	8.2e-10 (6.9e-10)	3.0e-09 (2.2e-09)	3.5e-04 (1.0e-04)	2.7e-03 (5.1e-04)	1.1e-01 (3.9e-02)	4.6e-01 (6.6e-01)
	200,000	**3.80e-15 (1.66e-15)**	4.4e-15 (0.0e+00)	4.4e-15 (0.0e+00)	4.7e-15 (9.6e-16)	4.3e-14 (2.6e-14)	9.7e-11 (5.0e-11)	4.6e-01 (6.6e-01)
f_{11}	50,000	**1.73e-10 (1.21e-09)**	9.9e-08 (6.0e-07)	2.0e-04 (1.4e-03)	1.9e-05 (5.8e-05)	7.8e-04 (1.2e-03)	2.0e-01 (1.1e-01)	1.3e-02 (1.7e-02)
	300,000	0.0e+00 (0.0e+00)	0.0e+00 (0.0e+00)	2.0e-04 (1.4e-03)	0.0e+00 (0.0e+00)	0.0e+00 (0.0e+00)	0.0e+00 (0.0e+00)	1.1e-02 (1.6e-02)
f_{12}	50,000	**1.14e-19 (3.84e-19)**	4.6e-17 (1.9e-16)	3.8e-16 (8.3e-16)	1.6e-07 (1.5e-07)	1.9e-05 (9.2e-06)	1.2e-02 (1.0e-02)	1.9e-01 (3.9e-01)
	150,000	1.57e-32 (0.0e+00)	1.6e-32 (5.5e-48)	1.6e-32 (5.5e-48)	2.6e-29 (7.5e-29)	1.2e-19 (2.0e-19)	1.1e-14 (1.0e-14)	1.9e-01 (3.9e-01)
f_{13}	50,000	**5.68e-19 (1.22e-18)**	2.0e-16 (6.5e-16)	1.2e-15 (2.8e-15)	1.5e-06 (9.8e-07)	6.1e-05 (2.0e-05)	7.5e-02 (3.8e-02)	2.9e-03 (4.8e-03)
	150,000	**1.35e-32 (0.0e+00)**	1.4e-32 (1.1e-47)	1.4e-32 (1.1e-47)	1.9e-28 (2.2e-28)	1.7e-19 (2.4e-19)	7.5e-14 (4.8e-14)	2.9e-03 (4.8e-03)

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 to improve JADE by introducing the correlation of algorithm parameters in parameter tuning. It was shown that CADE outperformed DE/rand/1/bin in all problems. Also, CADE outperformed JADE, jDE, SaDE and PSO in more than 10 problems out of 13 problems. Thus, it is thought that CADE is a very efficient optimization algorithm compared with other methods.

In the future, we will design more dynamic control of parameter values for CADE.

ACKNOWLEDGMENT

This research is supported in part by Grant-in-Aid for Scientific Research (C) (No. 24500177, 26350443) of Japan society for the promotion of science and the 2014 Research Fund (ChosaKenkyu-Hi) of the Center for the Co-creation of Hiroshima's Future, Hiroshima Shudo University.

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. Price, R. 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] S. Das and P. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[5] J. Zhang 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.

[6] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, Feb. 2011.

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

[8] T. Takahama and S. Sakai, "Fuzzy c-means clustering and partition entropy for species-best strategy and search mode selection in nonlinear optimization by differential evolution," in *Proc. of the 2011 IEEE International Conference on Fuzzy Systems*, Jun. 2011, pp. 290–297.

[9] T. Takahama and S. Sakai, "Differential evolution with dynamic strategy and parameter selection by detecting landscape modality," in *Proc. of the 2012 IEEE Congress on Evolutionary Computation*, Jun. 2012, pp. 2114–2121.

[10] T. Takahama and S. Sakai, "Large scale optimization by differential evolution with landscape modality detection and a diversity archive," in *Proc. of the 2012 IEEE Congress on Evolutionary Computation*, Jun. 2012, pp. 2842–2849.

[11] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Comput.*, vol. 10, no. 8, pp. 673–686, 2006.

[12] A. Qin, V. Huang, and P. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, april 2009.

[13] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transaction on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[14] S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, "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*, vol. 42, no. 2, pp. 482–500, april 2012.

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

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

[17] 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.