# Solving Nonlinear Optimization Problems by Differential Evolution with a Rotation-Invariant Crossover Operation using Gram-Schmidt process

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 nonlinear, non-differentiable, non-convex and multimodal functions. However, the performance of DE degrades in problems with strong linkage among variables, where variables are related strongly each other. One of the desirable properties of optimization algorithms for solving the problems with strong linkage is rotation-invariant property. The rotation-invariant algorithms can solve rotated problems where variables are strongly related as in the same way of solving non-rotated problems. In DE, two operations are applied to each individual: a mutation operation, which is rotation-invariant, and a crossover operation, which is not rotation-invariant. Thus, DE is not rotation-invariant. In this study, we propose a new crossover operation that is rotation-invariant. In order to achieve rotation-invariant property, instead of using the fixed coordinate system, a new coordinate system is build from a current population, or search points in search process. Independent points, or vectors are selected from the population, Gram-Schmidt process is applied to them in order to obtain orthogonal vectors, and the vectors form the new coordinate system. The effect of the rotation-invariant crossover operation is shown by solving some benchmark problems.**

***Keywords*-differential evolution; crossover; rotation-invariant; Gram-Schmidt process**

## 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 with strong linkage among variables, where variables are related strongly each other. One of the desirable properties of optimization algorithms for solving the problems with strong linkage is rotation-invariant property. The rotation-invariant algorithms can solve rotated problems where variables are strongly related as in the same way of solving non-rotated problems. In DE, two operations are applied to each individual: a mutation operation, which is rotation-invariant, and a crossover operation, which is not rotation-invariant. Thus, DE is not rotation-invariant [2].

In this study, we propose a new crossover operation that is rotation-invariant. 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. A child, or a trial vector is generated by crossover which decomposes the difference between a parent and the mutant vector into elements of a coordinate system, some elements are selected probabilistically and the elements are combined. In order to achieve rotation-invariant property, instead of using the fixed coordinate system, a new coordinate system is build from a current population, or search points in search process. Independent points, or vectors are selected from the population, Gram-Schmidt process is applied to them in order to obtain orthogonal vectors, and the vectors form the new coordinate system.

The effect of the rotation-invariant crossover operation is shown by solving some benchmark problems.

In Section II, the rotation-invariant property of some crossover operations are examined. Differential evolution is explained in Section III. A new rotation-invariant crossover operation is proposed in Section IV. In Section V, experimental results on some problems are shown. Finally, conclusions are described in Section VI.

## II. Optimization and Rotation-Invariant Property

In this section, optimization problems are defined and rotation-invariant property is explained.

### A. Optimization Problems

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

$$\begin{array}{lll} \text{(P)} & \text{minimize} & f(\boldsymbol{x}) \\ & \text{subject to} & l_i \le x_i \le u_i,\ i = 1, \ldots, n, \end{array} \quad (1)$$

where $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ is an $n$ 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. *Rotation-Invariant Crossover*

In EAs, crossover operations play important roles in optimization process. Some representative crossover operations used in real-coded EAs are examined from the viewpoint of rotation-invariant property in the following. DE adopts two-parent crossover operations, although there exist some crossover operations using multiple (more than two) parents such as unimodal normal distribution crossover (UNDX) [4] and simplex crossover (SPX) [5]. In two-parent crossover operations, it can be assumed that two individuals $\boldsymbol{x}$ and $\boldsymbol{y}$ are recombined and a child $\boldsymbol{z}$ is generated.

- Arithmetic crossover [6]
  Arithmetic crossover generates a child that is a linear combination of two parents:
  $$z_i = rx_i + (1 - r)y_i \tag{2}$$
  where $r$ is a uniform random number in $[0, 1]$. Fig. 1 shows the arithmetic crossover. Black circles correspond to parents and a white circle corresponds to the child. When a given problem is rotated and search points are rotated, the relation between parents and the child that is denoted by a gray (green) circle is not changed. Therefore, the arithmetic crossover is rotation-invariant.

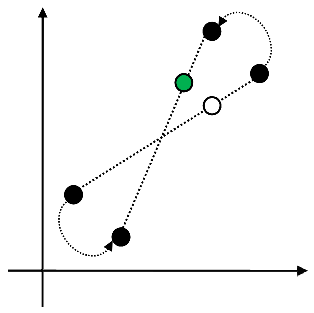

Figure 1. Arithmetic crossover and its rotation

- Uniform crossover
  Uniform crossover generates a child by taking each gene from the first or the second parent with the same probability:
  $$z_i = \begin{cases} x_i & \text{with prob. } 0.5 \\ y_i & \text{with prob. } 0.5 \end{cases} \tag{3}$$
  Fig. 2 shows the uniform crossover. Black circles correspond to parents and one of white circles corresponds to the child. When a given problem is rotated and search points are rotated, the child corresponds to one of circles with (red) diagonal lines and does not correspond to one of gray (green) circles. Therefore, the uniform crossover is not rotation-invariant.

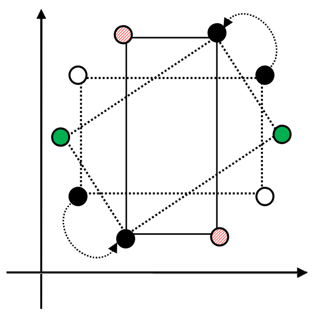

Figure 2. Uniform crossover and its rotation

- Blend crossover (BLX-$\alpha$) [7]
  Blend crossover can be defined as follows:
  $$z_i = r_i x_i + (1 - r_i)y_i \tag{4}$$
  where $r_i$ is a uniform random number in $[-\alpha, 1 + \alpha]$ and generated in each dimension. The parameter $\alpha$ ($\alpha \geq 0$) specifies how much the region, where a child will be generated, is enlarged. If $\alpha$ is zero, the child is generated in a hyper-rectangle that is formed by two parents. The rectangle is shown in Fig. 2 by dotted lines. Therefore, the blend crossover is not rotation-invariant.

In this study, a rotation-invariant crossover operation is proposed for DE.

## III. Differential Evolution

In this section, the outline of DE is described.

### A. *Outline*

In DE, initial individuals are randomly generated within given search space and form an initial population. Each individual contains $n$ 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 $1 + 2\ num$ 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 $num$ 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, such as DE/rand/1/exp. The variants are classified using the notation DE/$base$/$num$/$cross$.

"$base$" 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. 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.

$$\boldsymbol{x}' = \boldsymbol{x}^{p1} + F(\boldsymbol{x}^{p2} - \boldsymbol{x}^{p3}) \tag{5}$$

"$num$" specifies the number of difference vectors used to perturb the base vector.

"$cross$" specifies the type of crossover 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. 3 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,$n$);
  for($k$=1; $k \le n$; $k$++) {
    if($k == j_{rand}$ || $u(0,1) < CR$) $x_k^{\text{child}}$=$x'_k$;
    else $x_k^{\text{child}}$=$x_k^i$;
  }
exponential crossover DE/·/·/exp
  $k$=1; $j$=randint(1,$n$);
  do {
      $x_j^{\text{child}}$=$x'_j$;
      $k$=$k$+1; $j$=$(j+1)\%n$;
  } while($k \le n$ && $u(0,1) < CR$);
  while($k \le n$) {
      $x_j^{\text{child}}$=$x_j^i$;
      $k$=$k$+1; $j$=$(j+1)\%n$;
  }
```

Figure 3. Binomial and exponential crossover operation, where randint(1,$n$) generates an integer randomly from $[1, n]$ and $u(l, r)$ is a uniform random number generator in $[l, r]$.

## B. The Algorithm of DE

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 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. (5). 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. 3. 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. 4 shows a pseudo-code of DE/rand/1/exp.

```
DE/rand/1/exp()
{
// Initialize an 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 $P(p2 \ne i \ne p1)$;
     $\boldsymbol{x}^{p3}$=Randomly selected from $P(p3 \ne i \ne p1 \ne p2)$;
     $\boldsymbol{x}'$=$\boldsymbol{x}^{p1}$+$F(\boldsymbol{x}^{p2} - \boldsymbol{x}^{p3})$;
     $\boldsymbol{x}^{\text{child}}$=trial vector is generated from
           $\boldsymbol{x}^i$ and $\boldsymbol{x}'$ by exponential crossover;
// Survivor selection
     if$\big(f(\boldsymbol{x}^{\text{child}}) \le f(\boldsymbol{x}^i)\big)$ $\boldsymbol{z}^i$=$\boldsymbol{x}^{\text{child}}$;
     else                $\boldsymbol{z}^i$=$\boldsymbol{x}^i$;
     $FE$=$FE$+1;
  }
  $P$={$\boldsymbol{z}^i,\ i = 1, 2, \cdots, N$};
 }
}
```

Figure 4. The pseudo-code of DE, $FE$ is the number of function evaluations.

## IV. Rotation-Invariant Crossover

The binomial and exponential crossover operations are not rotation-invariant, because the operations are similar to uniform crossover and select a vertex from vertices of a hyper-rectangle, of which diagonal positions are occupied by a parent and a mutant vector, as a child. One way of realizing rotation-invariant crossover is to use a coordinate system which is defined by search points instead of using the fixed coordinate system.

### A. A Coordinate System and Coordinate Vectors

A coordinate system is defined by coordinate vectors which are mutually orthogonal unit vectors. In this study, the coordinate vectors are generated from a population, or search points $P = \{\boldsymbol{x}^i, i = 1, 2, \cdots, N\}$ using Gram-Schmidt process as follows:

1) Calculating the centroid of the search points

$$c = \frac{1}{N}\sum_i x^i \tag{6}$$

2) Calculating directional vectors from the centroid

$$d_i = x^i - c,\ i = 1, 2, \cdots, N \tag{7}$$

3) Selecting $n$ vectors from the directional vectors In this study, vectors are selected randomly from the directional vectors.

$$V = \{v_k | k = 1, 2, \cdots, n, v_k \in \{d_i\}\} \tag{8}$$

4) Orthonormalizing the selected vectors using Gram-Schmidt process

$$b_1 = \frac{v_1}{||v_1||} \tag{9}$$

$$b_2 = \frac{v_2 - (v_2, b_1)b_1}{||v_2 - (v_2, b_1)b_1||} \tag{10}$$

$$\vdots \qquad \vdots$$

$$b_n = \frac{v_n - \sum_{i=1}^{n-1}(v_n, b_i)b_i}{||v_n - \sum_{i=1}^{n-1}(v_n, b_i)b_i||} \tag{11}$$

where $(v, b)$ is the inner product of $v$ and $b$.

## *B. A Crossover Operation Based on Coordinate Vectors*

In the binomial and exponential crossover operations, either element of the parent $x^i$ or the mutant vector $x'$ is selected. In other words, the vector $y$ from the parent to the mutant vector is decomposed to $n$ elements and some elements are selected probabilistically. Thus, the operations can be defined as follows:

$$y = x' - x^i \tag{12}$$

$$x^{\text{child}} = x^i + \sum_{k \in K}(y, e_k)e_k \tag{13}$$

where $K$ is the set of indexes of selected elements, and $e_k$ is a unit vector of which $k$-th element is 1 and other elements are 0. The new coordinate vectors $b_k$'s can be used instead of $e_k$'s. Fig. 5 shows the definition of the rotation-invariant binomial and exponential crossovers. Fig. 6 shows an example of the rotation-invariant crossover in two-dimensions.

## *C. Algorithm of DE with a rotation-invariant crossover operation*

Some modifications to standard DE are applied to DE with a rotation-invariant crossover operation (RIDE).

1) Continuous generation model [8], [9]. 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.

```
  $y$=$x'$-$x^i$;
  $x^{\text{child}}$=$x^i$;
Rotation-invariant binomial crossover
  $j$=randint(1,$n$);
  for($k$=1; $k \le n$; $k$++) {
    if($k == j$ || $u(0,1) < CR$)
      $x^{\text{child}}$=$x^{\text{child}}$+$(y, b_k)b_k$;
  }
Rotation-invariant exponential crossover
  $k$=1; $j$=randint(1,$n$);
  do {
      $x^{\text{child}}$=$x^{\text{child}}$+$(y, b_j)b_j$;
      $k$=$k$+1; $j$=$(j+1)\%n$;
  } while($k \le n$ && $u(0,1) < CR$);
```

Figure 5. Binomial and exponential rotation-invariant crossover operations, where randint(1,$n$) generates an integer randomly from $[1, n]$ and $u(l, r)$ is a uniform random number generator in $[l, r]$.

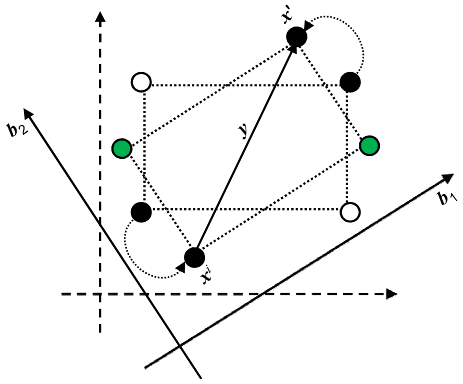

Figure 6. Rotation-invariant crossover in two-dimensions

2) Generating two children at most [10], [11]. When a parent generates a child by using standard crossover and the child is not better than the parent, the parent generates another child by using rotation-invariant crossover. It is thought that generating two children improves robustness of the search process because two types of crossover operations will generate different types of children and the diversity of the search points will increase.

3) Reflecting back out-of-bound solutions [12]. In order to keep bound constraints, an operation to move a point outside of search space into the inside of the space 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 [13]. In this study, reflecting back is used:

$$x_{ij} = \begin{cases} l_i + (l_i - x_{ij}) - \left\lfloor \frac{l_i - x_{ij}}{u_i - l_i} \right\rfloor (u_i - l_i) & (x_{ij} < l_i) \\ u_i - (x_{ij} - u_i) + \left\lfloor \frac{x_{ij} - u_i}{u_i - l_i} \right\rfloor (u_i - l_i) & (x_{ij} > u_i) \\ x_{ij} & (\text{otherwise}) \end{cases} \tag{14}$$

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.

Fig. 7 shows the pseudo-code of RIDE.

```
RIDE/rand/1/exp()
{
// Initialize an population
 $P$=$N$ individuals generated randomly in $\mathcal{S}$;
 for($t$=1; $FE \leq FE_{max}$; $t$++) {
  $B$=obtained coordinate vectors;
  for($i$=1; $i \leq N$; $i$++) {
   for($k$=1; $k$<=2; $k$++) {
// DE operation
     $\boldsymbol{x}^{p1}$=Randomly selected from $P(p1 \neq i)$;
     $\boldsymbol{x}^{p2}$=Randomly selected from $P(p2 \neq i \neq p1)$;
     $\boldsymbol{x}^{p3}$=Randomly selected from $P(p3 \neq i \neq p1 \neq p2)$;
     $\boldsymbol{x}'$=$\boldsymbol{x}^{p1}$+$F(\boldsymbol{x}^{p2} - \boldsymbol{x}^{p3})$;
     if($k$==1)
       $\boldsymbol{x}^{\text{child}}$=trial vector is generated from
              $\boldsymbol{x}^i$ and $\boldsymbol{x}'$ by exponential crossover;
     else
       $\boldsymbol{x}^{\text{child}}$=trial vector is generated from
              $\boldsymbol{x}^i$ and $\boldsymbol{x}'$ by rotation-invariant
              exponential crossover using $B$;
     $FE$=$FE$+1;
// Survivor selection
     if$\left(f(\boldsymbol{x}^{\text{child}}) \leq f(\boldsymbol{x}^i)\right)$ {
       $\boldsymbol{x}^i$=$\boldsymbol{x}^{\text{child}}$;
       break;
     }
   }
  }
 }
}
```

Figure 7. The pseudo-code of RIDE, $FE$ is the number of function evaluations.

## V. 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 [14]. 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].

Independent 30 runs are performed for 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 three algorithms, SDE (standard DE with discrete generation model), CDE (DE with Continuous model) and RIDE are compared. The parameters are: $F = 0.7$, $CR = 0.9$ and exponential crossover is adopted, because these settings showed very good and stable performance [18]. The population size is 60 ($N = 60$). The number of function evaluations (FEs) until finding a near optimal solution is compared.

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

### B. Experimental Results

Table II shows the experimental results. The mean number of FEs until finding a near optimal value and their standard

Table II
EXPERIMENTAL RESULTS. MEAN VALUE $\pm$ STANDARD DEVIATION AND RATIO OF THE MEAN VALUE RELATIVE TO THAT OF SDE IN 30 RUNS ARE SHOWN

| | SDE | CDE | RIDE |
|---|---|---|---|
| $f_1$ | 120714.9 $\pm$ 1228.8 | 119090.0 $\pm$ 1042.4 | **51547.8 $\pm$ 1214.6** |
| | (1.000) | (0.987) | (**0.433**) |
| $f_2$ | 171604.0 $\pm$ 1417.5 | 168699.2 $\pm$ 1740.7 | **86474.9 $\pm$ 1366.2** |
| | (1.000) | (0.983) | (**0.513**) |
| $f_3$ | 1015452.8 $\pm$ 12888.5 | 1014411.1 $\pm$ 13237.6 | **178453.9 $\pm$ 4588.0** |
| | (1.000) | (0.999) | (**0.176**) |
| $f_4$ | 1064604.6 $\pm$ 10676.0 | 1058717.6 $\pm$ 11755.0 | **187892.2 $\pm$ 3654.8** |
| | (1.000) | (0.994) | (**0.177**) |
| $f_5$ | 395632.8 $\pm$ 7141.7 | 384640.8 $\pm$ 5492.1 | **343023.3 $\pm$ 7809.0** |
| | (1.000) | (0.972) | (**0.892**) |
| $f_6$ | 48922.1 $\pm$ 933.9 | 48378.0 $\pm$ 1190.6 | **19771.3 $\pm$ 1019.1** |
| | (1.000) | (0.989) | (**0.409**) |
| $f_7$ | 668549.4 $\pm$ 102128.1 | 637370.6 $\pm$ 129435.1 | **49654.5 $\pm$ 15911.3** |
| | (1.000) | (0.953) | (**0.078**) |
| $f_8$ | 144896.8 $\pm$ 2681.3 | 142918.6 $\pm$ 2039.9 | **128269.3 $\pm$ 4336.1** |
| | (1.000) | (0.986) | (**0.897**) |
| $f_9$ | 260549.0 $\pm$ 6495.7 | **259204.2 $\pm$ 6291.9** | 349479.2 $\pm$ 12522.2 |
| | (1.000) | (**0.995**) | (1.348) |
| $f_{10}$ | 180778.8 $\pm$ 1574.8 | 177994.4 $\pm$ 1690.7 | **78384.2 $\pm$ 1355.4** |
| | (1.000) | (0.985) | (**0.440**) |
| $f_{11}$ | 127930.9 $\pm$ 4080.4 | 127655.2 $\pm$ 4210.2 | **60420.4 $\pm$ 1597.8** |
| | (1.000) | (0.998) | (**0.473**) |
| $f_{12}$ | 107109.1 $\pm$ 1408.0 | 106601.9 $\pm$ 1631.1 | **49109.4 $\pm$ 1703.9** |
| | (1.000) | (0.995) | (**0.461**) |
| $f_{13}$ | 115676.4 $\pm$ 1441.7 | 114009.8 $\pm$ 979.3 | **53068.2 $\pm$ 1167.2** |
| | (1.000) | (0.986) | (**0.465**) |

deviation are shown in the top row for each function. Also, the ratio of the mean number of FEs relative to that of SDE is shown in the bottom row and in parentheses. The best result among three algorithms is highlighted using bold face fonts.

As for 6 problems $f_1$, $f_6$, $f_{10}$, $f_{11}$, $f_{12}$ and $f_{13}$, RIDE can find near optimal solutions in less than half FEs, and in about half FEs for the problem $f_2$ compared with SDE and CDE. As for 3 problems $f_3$, $f_4$ and $f_7$, RIDE can solve the problems more than 5 times faster than other algorithms. For 2 problems $f_5$ and $f_8$, RIDE solved the problems about 10% faster than the others. As for the problem $f_9$, CDE solved the problem fastest and RIDE solved it slowest. It is thought that the rotation-invariant property is not effective to the problem.

It is shown that RIDE can solve 10 problems very fast and 2 problems slightly fast. Thus, it is thought that the rotation-invariant crossover is effective to various problems.

Figures 8 to 20 show the change of best objective value found over the number of FEs within 150,000 evaluations. Apparently, RIDE can find better objective values faster than SDE and CDE in all problems except for $f_9$.

## VI. CONCLUSION

Differential evolution is known as a simple, efficient and robust search algorithm that can solve nonlinear optimization problems. In this study, we proposed a rotation-invariant crossover in order to improve the efficiency and also stability

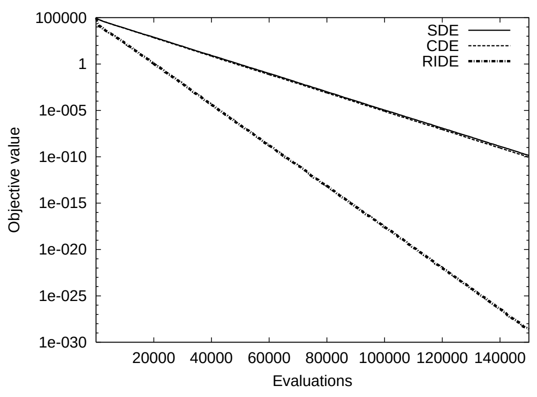

Figure 8. The graph of $f_1$

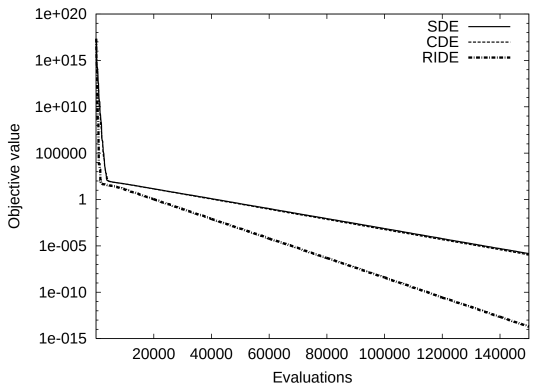

Figure 9. The graph of $f_2$

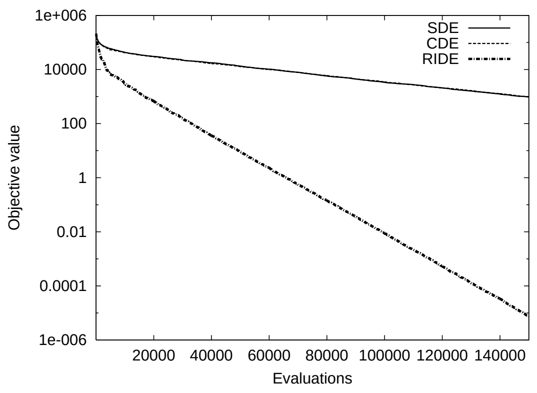

Figure 10. The graph of $f_3$

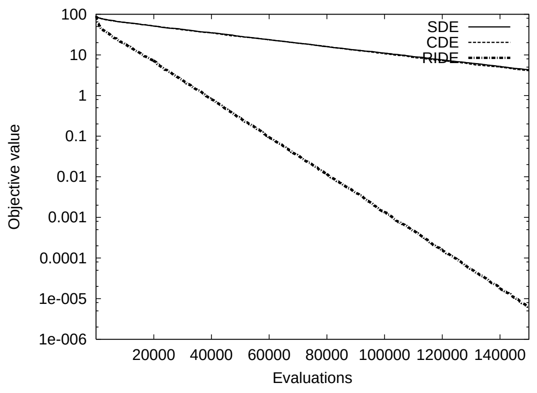

Figure 11. The graph of $f_4$

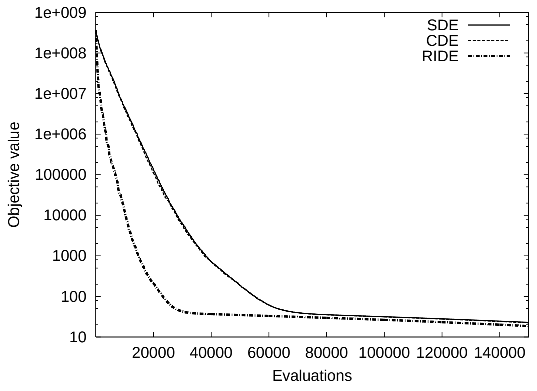

Figure 12. The graph of $f_5$

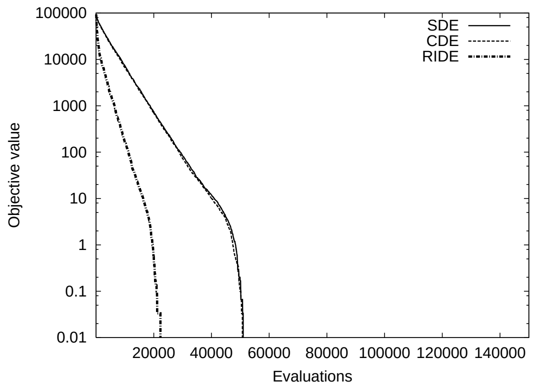

Figure 13. The graph of $f_6$

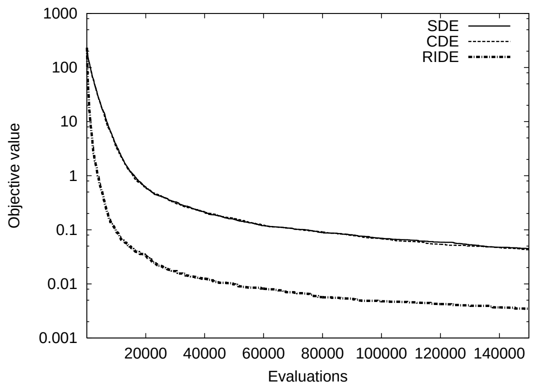

Figure 14. The graph of $f_7$

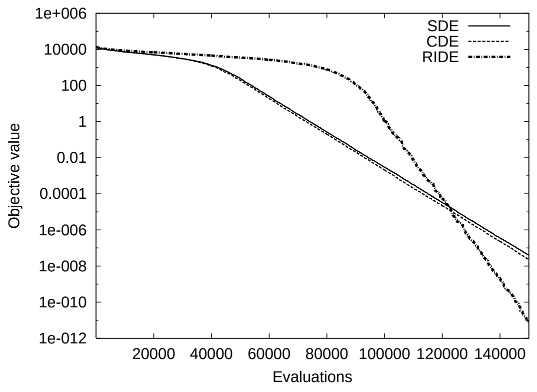

Figure 15. The graph of $f_8$

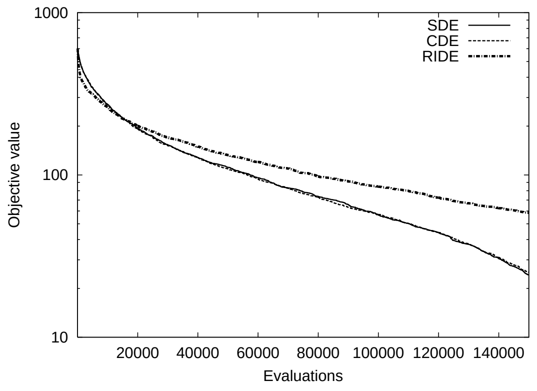

Figure 16. The graph of $f_9$

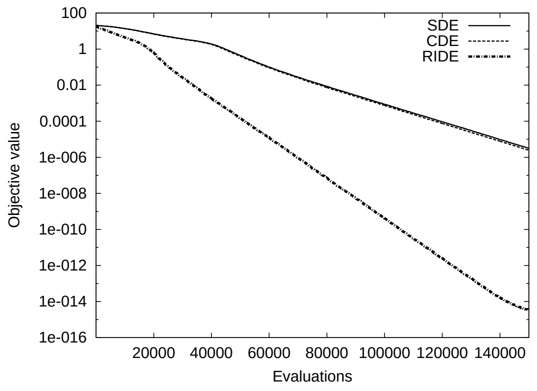

Figure 17. The graph of $f_{10}$

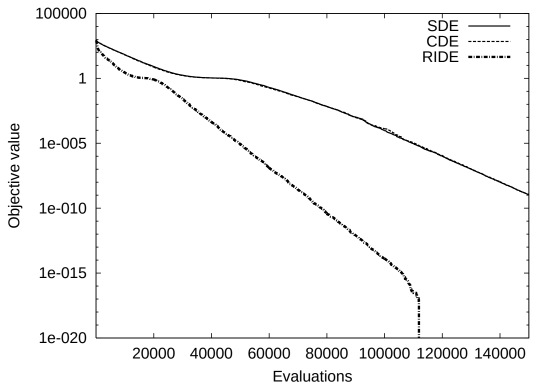

Figure 18. The graph of $f_{11}$

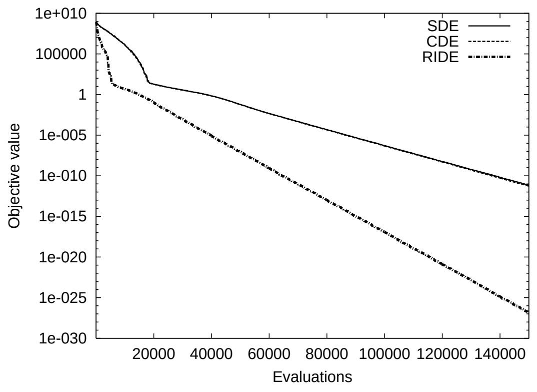

Figure 19. The graph of $f_{12}$

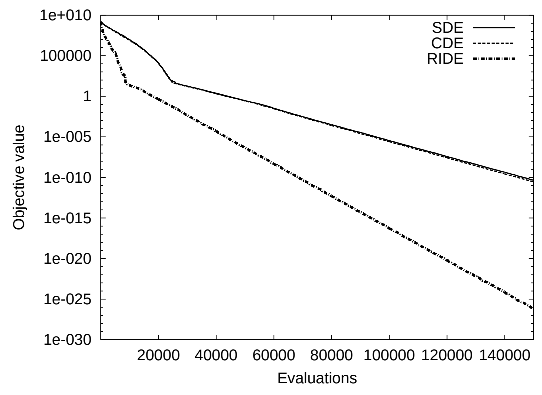

Figure 20. The graph of $f_{13}$

of DE. It was shown that RIDE can reduce the number of function evaluations for finding near optimal solutions more than 50% in 9 problems out of 13 problems. Thus, it is thought that RIDE is a very efficient optimization algorithm compared with standard DEs.

In the future, we will apply RIDE to various real world problems that have large numbers of decision variables and constraints.

## 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] I. Ono and S. Kobayashi, "A real coded genetic algorithm for function optimization using unimodal normal distributed crossover," in *Proceedings of the 7th International Conference on Genetic Algorithms*, 1997, pp. 246–253.

[5] S. Tsutsui, M. Yamamura, and T. Higuchi, "Multi-parent recombination with simplex crossover in real coded genetic algorithms," in *Proc. of Genetic and Evolutionary Computation Conference(GECCO'99)*, 1999, pp. 657–664.

[6] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs (3rd ed.)*. London, UK: Springer-Verlag, 1996.

[7] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval schemata," in *Foundations of Genetic Algorithms 2*, L. D. Whitley, Ed. San Mateo, CA: Morgan Kaufmann Publishers, 1993, pp. 187–202.

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

[9] 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.

[10] T. Takahama and S. Sakai, "Constrained optimization by the epsilon constrained differential evolution with an archive and gradient-based mutation," in *Proc. of the 2010 IEEE Congress on Evolutionary Computation*, Jul. 2010, pp. 1680–1688.

[11] T. Takahama and S. Sakai, "Efficient constrained optimization by the $\varepsilon$ constrained adaptive differential evolution," in *Proc. of the 2010 IEEE Congress on Evolutionary Computation*, Jul. 2010, pp. 2052–2059.

[12] 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.

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

[14] 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.

[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.

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