Efficient Nonlinear Optimization by Differential Evolution with a Rotation-Invariant Local Sampling Operation

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 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 strongly related to each other. One of the desirable properties of optimization algorithms for solving the problems with the strong dependence is rotation-invariant property. In DE, the mutation operation is rotation-invariant, but the crossover operation is not rotation-invariant usually. In this study, we propose a new operation, called local sampling operation that is rotation-invariant. In the operation, independent points are selected from the population, difference vectors from a parent to the points span a local area centered at the parent, and a new point is generated around the area. Also, the operation is used for judging whether intensive search or extensive search is desirable in each generation. The effect of the proposed method is shown by solving some benchmark problems.**

***Keywords*-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]–[4]. 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 strongly related to each other. One of the desirable properties of optimization algorithms for solving the problems with the strong dependence 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: The mutation operation is rotation-invariant, but the crossover operation is not rotation-invariant usually. In this case, DE is not rotation-invariant [2].

In this study, we propose a new operation, called local sampling operation that is rotation-invariant. In the operation, independent points are selected from the population and difference vectors from a parent to the selected points are obtained. A local area centered at the parent is spanned by the difference vectors. A new point or a child is generated from the vectors around the area using a uniform probability distribution. The operation samples a point around the local area and realizes intensive search. Thus, it is expected that the operation improves the efficiency of the search. Also, the operation is used for judging whether intensive search or extensive search is desirable. In this study, both of the local sampling operation and the ordinary DE operations, or ordinary mutation and crossover, are used probabilistically. If the success rate of the local sampling operation is lower than that of the ordinary operations, it is thought that extensive search will be more desirable than intensive search. On the contrary, if the success rate of the local sampling operation is greater than that of the ordinary operations, the effect of the intensive search might be too strong. In this case, the probability of the intensive search will be decreased to prevent premature convergence. The effect of the proposed method is shown by solving 13 benchmark problems including multimodal problems and problems with strong dependence.

In Section II, rotation-invariant operations are briefly reviewed. DE is explained in Section III, and DE with the local sampling 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 study, the following optimization problem (P) with lower bound and upper bound constraints will be discussed.

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

where $\boldsymbol{x} = (x_1, x_2, \cdots, x_D)$ is an 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 $\mathcal{S}$.

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.

A. None Rotation-Invariant Crossovers

DE adopts two-parent crossover operations. 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.

Uniform crossover generates a child that contains randomly selected elements in either elements of two parents:

$$z_i = \begin{cases} x_i & \text{with prob. } 0.5 \\ y_i & \text{with prob. } 0.5 \end{cases} \tag{2}$$

Fig. 1 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 red (light gray) circles and does not correspond to one of green (dark gray) circles. Therefore, the uniform crossover is not rotation-invariant.

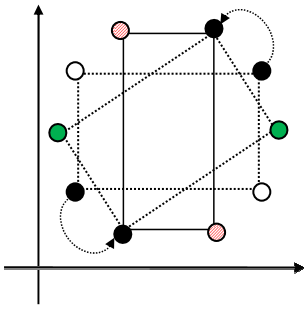

Fig. 1. Uniform crossover and its rotation

Binomial and exponential crossover operations in DE 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.

B. Rotation-Invariant Crossovers

Arithmetic crossover [5] generates a child that is a linear combination of two parents:

$$z_i = rx_i + (1-r)y_i \tag{3}$$

where r is a uniform random number in $[0, 1]$. Fig. 2 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 green (gray) circle is not changed. Therefore, the arithmetic crossover is rotation-invariant.

In DE researches, some rotation-invariant operations are proposed based on the arithmetic crossover such as current-to-rand and current-to-best with crossover rate $CR = 1$ [6]. However, generated points by the operations cover very limited area, and the diversity of the population is rapidly lost. Thus, the operations have problems of premature convergence. Recently, a rotation-invariant crossover operation for DE is proposed [7]. In the operation, an orthogonal coordinate system is built from search points using Gram-Schmidt process. However, the operation is not scale-invariant and the performance of the crossover operation is not high for multimodal problems and problems with strong dependence among variables.

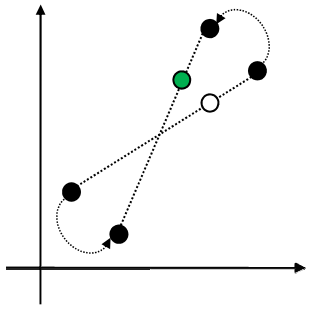

Fig. 2. Arithmetic crossover and its rotation

There exist some crossover operations using multiple (more than two) parents such as unimodal normal distribution crossover (UNDX) [8], simplex crossover (SPX) [9], and real-coded ensemble crossover (REX) [10]. REX is rotation-invariant and scale-invariant crossover. REX generates a child from multiple parents randomly selected from the population without overlapping each other. Let parents be denoted by $\{\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^m\}$ and their centroid by $\boldsymbol{x}^g$. The child $\boldsymbol{x}^c$ is generated according to the following equations:

$$\boldsymbol{x}^c = \boldsymbol{x}^g + \sum_{i=1}^{m} \xi^i (\boldsymbol{x}^i - \boldsymbol{x}^g) \tag{4}$$

$$\xi^i \sim \phi(0, \sigma_\xi^2),\ \sigma_\xi^2 = \frac{1}{m} \tag{5}$$

$$\boldsymbol{x}^g = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{x}^i \tag{6}$$

where m is the number of parents ($m \geq D$), ξ^i is a random number for each parent obeying ϕ, and $\phi(0, \sigma_\xi^2)$ is a symmetric probability distribution with mean 0 and variance σ_ξ^2. Examples of ϕ are as follows:

$$\phi(0, \sigma^2) = N(0, (\sqrt{1/m})^2) \tag{7}$$

$$\phi(0, \sigma^2) = U(-\sqrt{3/m}, \sqrt{3/m}) \tag{8}$$

where $U(l, r)$ is a uniform distribution in $[l, r]$ and N is a normal distribution.

Fig. 3 shows an example of the REX using the uniform distribution in two-dimensions where $D = 2$ and $m = 3$.

In this study, a new operation derived from REX is proposed.

III. Differential Evolution

In this section, the outline of DE is described.

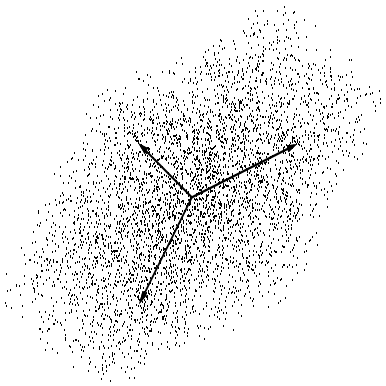

Fig. 3. An Example of Generating Points using REX

A. Outline of Differential Evolution

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 $\boldsymbol{x}^{p1}$, $\boldsymbol{x}^{p2}$ 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.

$$\boldsymbol{x}' = \boldsymbol{x}^{p1} + F(\boldsymbol{x}^{p2} - \boldsymbol{x}^{p3}) \tag{9}$$

"$cross$" 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. 4 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. 4. 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 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. (9). 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. 4. 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. 5 shows a pseudo-code of DE/rand/1.

IV. DE with Local Sampling Operation

In this section, local sampling operation derived from REX and DE with the operation are proposed.

A. Local Sampling Operation

In order to realize the sampling around a local area centered at a point or a parent, it needs to span the area by some vectors starting at the parent. Thus, in local sampling operation, the difference vectors from a parent $\boldsymbol{x}^p$ to randomly selected m individuals are used to generate a child where the uniform distribution is adopted as a probability distribution as follows:

$$\boldsymbol{x}^c = \boldsymbol{x}^p + \sum_{i=1}^{m} \xi^i (\boldsymbol{x}^i - \boldsymbol{x}^p) \tag{10}$$

$$\xi^i \sim U(-\sqrt{3/m}, \sqrt{3/m}) \tag{11}$$

```
DE/rand/1()
{
// Initialize an population
 P=N individuals generated randomly in S;
 for(t=1; FE ≤ FE_max; t++) {
  for(i=1; i ≤ N; i++) {
// DE operation
     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. 5. The pseudo-code of DE, FE is the number of function evaluations.

where m=$D+1$.

B. Algorithm of DE with Local Sampling Operation

Fig. 6 shows the pseudo-code of DE with the local sampling operation. Some modifications to standard DE are applied for proposed method as follows:

1) Two strategies, the local sampling operation and the ordinary operation (rand/1/exp), are adopted and the sampling operation is selected with a local sampling rate (LSR). The rate is controlled by the relative success rate of the local sampling operation (R_1^{succ}) over that of the ordinary operation (R_2^{succ}). When a child survives, the operation that has generated the child is considered as a success operation. A new algorithm parameter $LSR_{\max}$ is introduced to limit the maximum value of the local sampling rate.
   If the success rate of the local sampling operation is greater than that of the ordinary operation ($R_1^{succ} > R_2^{succ}$), the effect of the intensive search might be too strong and may cause too faster convergence. To prevent the premature convergence, the local sampling rate is lowered ($0.5LSR$) in next generation. If the success rate of the local sampling operation is very lower than that of the ordinary operation ($R_1^{succ} < R_2^{succ}/3$), the intensive search by the local operation is not effective and extensive search will be desired. In this study, the crossover rate is decreased in order to search wider area, because the default crossover rate 0.9 in this paper is for searching relatively narrow area.
2) Continuous generation model [11], [12] 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 [13] 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 [14]. In this study, reflecting

```
DE/{sampling/D+1 and rand/1/exp}()
{
 CR=CR_0; LSR=LSR_max; m=D+1;
// Initialize an population
 P=N individuals generated randomly in S;
 for(t=1; FE ≤ FE_max; t++) {
 // initialization of counters
  N_op^succ=N_op^fail=0 (op=1,2);
  for(i=1; i ≤ N; i++) {
   if(u(0,1) < LSR) {
     op=1;
// DE/sampling/m operation
     {p_1, p_2, ..., p_m}=Randomly selected from P without
                  overlapping with i and each other;
     x^c=x^i+Σ_{k=1}^{m} ξ^k(x^{p_k} − x^i);
   }
   else {
     op=2;
// DE/rand/1/exp operation
     x^{p1}=Randomly selected from P(p_1 ≠ i);
     x^{p2}=Randomly selected from P(p_2 ≠ i ≠ p_1);
     x^{p3}=Randomly selected from P(p_3 ≠ i ≠ p_1 ≠ p_2);
     x'=x^{p1}+F(x^{p2} − x^{p3});
     x^c=trial vector is generated from
              x^i and x' by exponential crossover;
   }
   FE=FE+1;
// Survivor selection
   if(f(x^c) ≤ f(x^i)) {
     x^i=x^c;
     N_op^succ=N_op^succ + 1;
   }
   else
     N_op^fail=N_op^fail + 1;
// updating LSR and selecting CR
   R_op^succ=N_op^succ / (N_op^succ+N_op^fail) (op=1,2);
   LSR=0.5LSR+0.5 R_1^succ/(R_1^succ+R_2^succ);
   if(LSR > LSR_max) LSR=LSR_max;
   CR=CR_0;
   if(R_1^succ > R_2^succ)
     LSR=0.5LSR;   // prevent premature convergence
   else if(R_1^succ < R_2^succ/3)
     CR=0.5CR;     // search wider area
  }
 }
}
```

Fig. 6. The pseudo-code of DE with local sampling operation where ξ^k=$u(-\sqrt{3/m}, \sqrt{3/m})$.

back is used:

$$x_{ij} = \begin{cases} l_i + (l_i - x_{ij}) - \left\lfloor \frac{l_i - x_{ij}}{u_i - l_i} \right\rfloor (u_i - l_i) & (x_{ij} < l_i) \\ u_i - (x_{ij} - u_i) + \left\lfloor \frac{x_{ij} - u_i}{u_i - l_i} \right\rfloor (u_i - l_i) & (x_{ij} > u_i) \\ x_{ij} & (\text{otherwise}) \end{cases} \tag{12}$$

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.

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 [15]. 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 [16]. 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 [17].

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. If any near optimal solution cannot be found within the maximum number of evaluations $FE_{\max}$, the run is considered as a failure run. 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 maximum number of evaluations $FE_{\max}$ is $D \times 10^5$ (4,000,000).

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 [18]

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 on Standard DEs

Table II shows the experimental results on some standard DEs: DE/rand/1/exp without and with the continuous generation model using F=0.7 and N=1.5D(60), DE/rand/1/bin using F=0.7 and N=1.5D(60) or 3D(120), and DE/rand/1/bin using F=0.5 and N=3D(120) or 5D(200) where CR=0.9.

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 DE/rand/1/exp without the continuous generation model is shown in the bottom row and in parentheses and the number of failure runs in which any near optimal solution cannot be found is shown in brackets. The best result among algorithms is highlighted using bold face fonts.

DE/rand1/1/exp can find near optimal solutions in all problems. On the contrary, DE/rand/1/bin cannot find any near optimal solutions in f_9, and sometimes cannot find near optimal solutions in f_3, f_4, and f_8. DE/rand/1/exp with the continuous generation model can find near optimal solutions faster than DE/rand/1/exp without the model in all problems. Therefore, from the viewpoint of stability and efficiency, it is thought that DE/rand/1/exp with the model using F=0.7 and CR=0.9 is the best standard DE to solve these problems. These settings showed very good and stable performance in constrained optimization [19].

C. Experimental Results on the Proposed Method

The efficiency of two algorithms, DE (standard DE with continuous generation model) and DE with local sampling operation with varying the maximum local sampling rate ($LSR_{\max}$) in 0.1, 0.2, 0.3, 0.4 and 0.5 are compared. The parameters are: exponential crossover, $F = 0.7$, $CR = 0.9$, population size N=1.5D(60).

Table III shows the experimental results. The ratio of the mean number of FEs relative to that of the standard DE is shown in the bottom row and in parentheses. The best result

TABLE II

EXPERIMENTAL RESULTS ON STANDARD DEs. MEAN VALUE ± STANDARD DEVIATION AND RATIO OF THE MEAN VALUE RELATIVE TO THAT OF THE DE/RAND/1/EXP IN 30 RUNS ARE SHOWN

	exp,N=60,F=0.7	exp,N=60,F=0.7,cont.	bin,N=60,F=0.7	bin,N=120,F=0.7	bin,N=120,F=0.5	bin,N=200,F=0.5
f_1	120687.6 ± 1221.2 (1.000)	**118810.9 ± 1124.8** (**0.984**)	273600.9 ± 7420.5 (2.267)	1234419.5 ± 37838.9 (10.228)	164949.2 ± 4609.7 (1.367)	368089.7 ± 7373.5 (3.050)
f_2	171661.1 ± 1220.2 (1.000)	**168780.6 ± 1431.4** (**0.983**)	445419.2 ± 12487.9 (2.595)	2095869.6 ± 66035.0 (12.209)	274991.9 ± 4572.9 (1.602)	624598.6 ± 10869.0 (3.639)
f_3	1018658.6 ± 15166.7 (1.000)	1013391.8 ± 15147.8 (0.995)	1513985.2 ± 69486.2 (1.486)	—— (——)[30]	**989561.6 ± 32844.7** (**0.971**)	2326449.9 ± 60224.8 (2.284)
f_4	1067726.3 ± 9962.8 (1.000)	**1062459.0 ± 10551.5** (**0.995**)	3719822.4 ± 272990.9 (3.484)[21]	—— (——)[30]	—— (——)[30]	1518417.2 ± 572038.5 (1.422)[2]
f_5	394404.4 ± 6095.7 (1.000)	**385424.9 ± 5781.6** (**0.977**)	1015989.5 ± 32078.1 (2.576)	3509049.3 ± 67394.5 (8.897)	778516.4 ± 20022.7 (1.974)	1325423.5 ± 19040.6 (3.361)
f_6	48922.1 ± 933.9 (1.000)	**48378.0 ± 1190.6** (**0.989**)	117252.9 ± 5938.6 (2.397)	515339.9 ± 20755.0 (10.534)	68150.6 ± 2835.2 (1.393)	151484.7 ± 4346.2 (3.096)
f_7	668549.4 ± 102128.1 (1.000)	637370.6 ± 129435.1 (0.953)	**618519.4 ± 213603.8** (**0.925**)	2878214.3 ± 648200.9 (4.305)[1]	285187.7 ± 70312.2 (0.427)	652842.2 ± 163166.7 (0.977)
f_8	145271.6 ± 1931.0 (1.000)	**143776.5 ± 2483.4** (**0.990**)	587550.0 ± 23764.9 (4.044)[25]	2588009.5 ± 186355.9 (17.815)[8]	437998.8 ± 64120.8 (3.015)[11]	1138988.6 ± 264940.6 (7.840)
f_9	260477.0 ± 6551.8 (1.000)	**259316.9 ± 6198.4** (**0.996**)	—— (——)[30]	—— (——)[30]	—— (——)[30]	—— (——)[30]
f_{10}	179986.9 ± 1541.5 (1.000)	**177519.0 ± 1551.8** (**0.986**)	412877.4 ± 11872.2 (2.294)	1859080.6 ± 39810.4 (10.329)	249526.8 ± 4568.6 (1.386)	552932.2 ± 8150.8 (3.072)
f_{11}	127775.0 ± 4265.3 (1.000)	**127422.2 ± 4366.1** (**0.997**)	280974.1 ± 7950.9 (2.199)	1257975.2 ± 35875.5 (9.845)	169502.7 ± 3213.2 (1.327)	375195.9 ± 6231.6 (2.936)
f_{12}	107053.5 ± 1373.2 (1.000)	**106594.1 ± 1615.0** (**0.996**)	258240.5 ± 9767.1 (2.412)	1194020.1 ± 47682.5 (11.153)	151385.9 ± 4734.2 (1.414)	341041.5 ± 8906.5 (3.186)
f_{13}	115407.5 ± 1481.4 (1.000)	**113853.3 ± 1156.7** (**0.987**)	278689.3 ± 11640.6 (2.415)	1263133.1 ± 37724.3 (10.945)	163955.0 ± 3572.6 (1.421)	364009.4 ± 7076.8 (3.154)

TABLE III

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

	DE,exp,N=60,F=0.7,cont.	$LSR_{\max}$ =0.1	$LSR_{\max}$ =0.2	$LSR_{\max}$ =0.3	$LSR_{\max}$ =0.4	$LSR_{\max}$ =0.5
f_1	118810.9 ± 1124.8 (1.000)	100972.8 ± 1559.2 (0.850)	86536.4 ± 1436.0 (0.728)	75447.5 ± 1207.6 (0.635)	66982.1 ± 1100.1 (0.564)	**66663.0 ± 948.8** (**0.561**)
f_2	168780.6 ± 1431.4 (1.000)	128531.5 ± 919.1 (0.762)	126651.3 ± 1710.4 (0.750)	**124648.1 ± 1153.8** (**0.739**)	124916.3 ± 1217.1 (0.740)	124700.6 ± 982.5 (0.739)
f_3	1013391.8 ± 15147.8 (1.000)	275779.2 ± 4446.3 (0.272)	173049.3 ± 5047.5 (0.171)	**153441.8 ± 4777.3** (**0.151**)	153483.1 ± 3278.1 (0.151)	154720.0 ± 4523.8 (0.153)
f_4	1062459.0 ± 10551.5 (1.000)	783882.6 ± 17057.7 (0.738)	562021.4 ± 13479.1 (0.529)	565417.7 ± 12034.2 (0.532)	**558065.8 ± 12404.9** (**0.525**)	559516.4 ± 13811.5 (0.527)
f_5	385424.9 ± 5781.6 (1.000)	329711.7 ± 6625.1 (0.855)	301176.7 ± 6522.3 (0.781)	285254.9 ± 7030.1 (0.740)	281899.0 ± 9921.7 (0.731)	**280037.9 ± 9764.2** (**0.727**)
f_6	48378.0 ± 1190.6 (1.000)	42077.3 ± 1244.4 (0.870)	36727.7 ± 1292.8 (0.759)	31599.5 ± 873.5 (0.653)	**27392.4 ± 1155.8** (**0.566**)	27425.8 ± 864.5 (0.567)
f_7	637370.6 ± 129435.1 (1.000)	300859.8 ± 71258.1 (0.472)	202957.3 ± 42308.6 (0.318)	137099.1 ± 30026.6 (0.215)	113842.0 ± 23450.0 (0.179)	**111413.2 ± 34472.5** (**0.175**)
f_8	143776.5 ± 2483.4 (1.000)	98116.1 ± 1166.3 (0.682)	**98015.0 ± 1165.1** (**0.682**)	98190.0 ± 1470.5 (0.683)	98392.6 ± 1694.1 (0.684)	98017.0 ± 1578.7 (0.682)
f_9	259316.9 ± 6198.4 (1.000)	**119022.0 ± 1531.2** (**0.459**)	121793.8 ± 2032.2 (0.470)	121856.7 ± 2197.9 (0.470)	122356.6 ± 1831.0 (0.472)	121519.9 ± 1968.4 (0.469)
f_{10}	177519.0 ± 1551.8 (1.000)	151936.2 ± 1707.9 (0.856)	130632.8 ± 1357.7 (0.736)	113076.1 ± 1372.8 (0.637)	**101621.9 ± 1473.8** (**0.572**)	102068.0 ± 1046.0 (0.575)
f_{11}	127422.2 ± 4366.1 (1.000)	109819.1 ± 6264.6 (0.862)	92179.6 ± 4865.7 (0.723)	80993.0 ± 5365.8 (0.636)	71416.1 ± 5190.2 (0.560)	**70353.4 ± 2509.1** (**0.552**)
f_{12}	106594.1 ± 1615.0 (1.000)	93476.3 ± 2706.8 (0.877)	83297.2 ± 1833.6 (0.781)	72247.7 ± 1733.9 (0.678)	69112.7 ± 2431.6 (0.648)	**68805.3 ± 1496.6** (**0.645**)
f_{13}	113853.3 ± 1156.7 (1.000)	98550.0 ± 1389.3 (0.866)	85755.3 ± 1402.5 (0.753)	74333.3 ± 1307.6 (0.653)	68831.4 ± 1775.5 (0.605)	**68361.5 ± 1281.7** (**0.600**)

among algorithms is highlighted using bold face fonts.

DE with the local sampling operation outperformed the standard DE in all problems and in all settings of $LSR_{\max}$. The best value of $LSR_{\max}$ seems to be 0.5, because the settings attained best results in 6 problems and second-best results in 6 problems. DE with the operation of $LSR_{\max} = 0.5$ can find near optimal solutions more than 5 times faster than the standard DE in problems f_3 and f_7. It can solve the problems in 50% to 60% FEs compared with the standard DE for problems f_1, f_4, f_6, f_9, f_{10}, f_{11} and f_{13}. In other problems, it can find near optimal solutions in 60% to 75% FEs. It is shown that proposed method can solve 9 problems very fast and 4 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 the proposed method with $LSR_{\max}$ being 0.1, 0.2, 0.3, 0.4 and 0.5. 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 7 to 15 show the change of best objective value found over the number of FEs within 200,000 evaluations without the graphs of f_3 and f_4, which are similar to that of f_1, and without the graphs of f_{12} and f_{13} which are similar to that of f_{11}. Apparently, proposed method can find better objective values faster than the standard DE in all problems.

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 the local sampling operation to improve the efficiency and also stability of DE. It was shown that the operation can reduce the number of function evaluations for finding near optimal solutions more than 40% in 9 problems out of 13 problems. Thus, it is thought that DE with the local sampling operation is a very efficient optimization algorithm compared with standard DEs.

The proposed method is scale-invariant but not completely rotation-invariant because the ordinary crossover operation is adopted probabilistically. In the future, we will design completely rotation-invariant and scale-invariant operations.

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] 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.

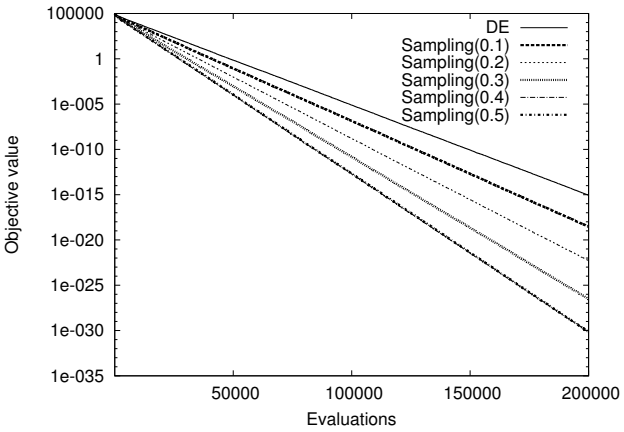

Fig. 7. The graph of f_1

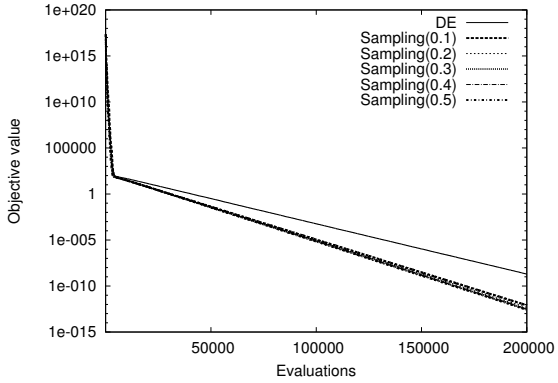

Fig. 8. The graph of f_2

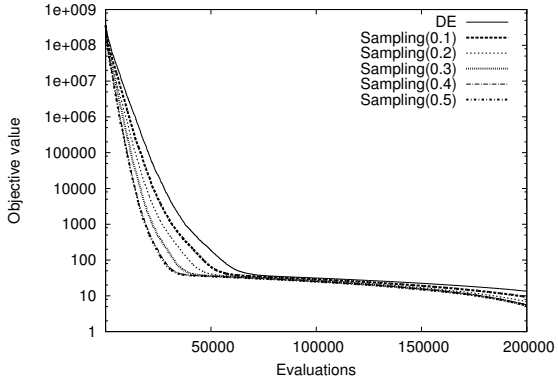

Fig. 9. The graph of f_5

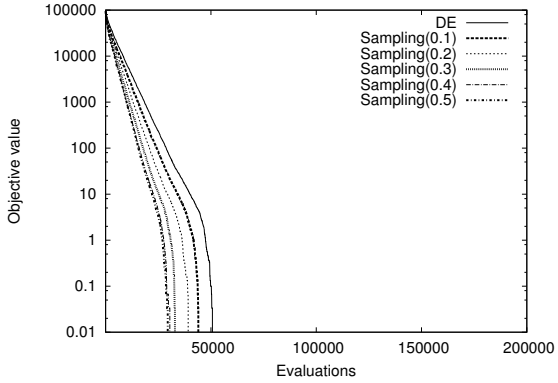

Fig. 10. The graph of f_6

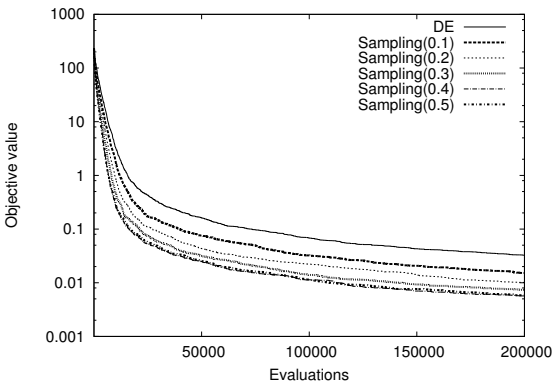

Fig. 11. The graph of f_7

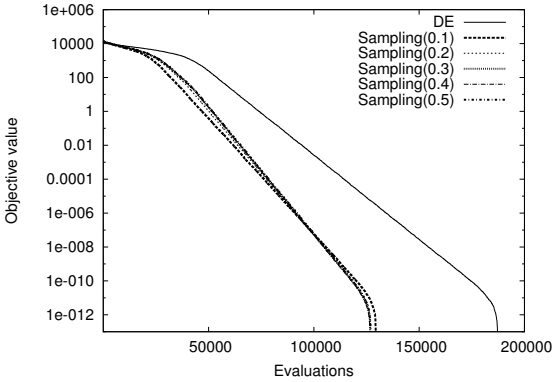

Fig. 12. The graph of f_8

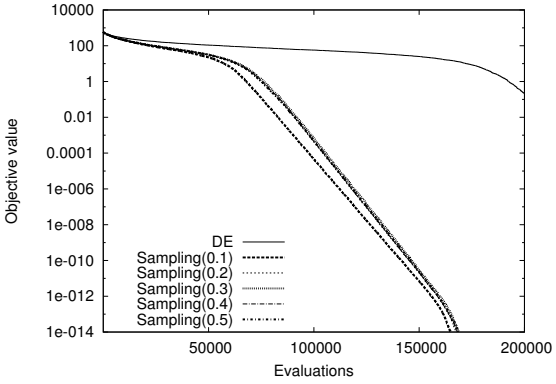

Fig. 13. The graph of f_9

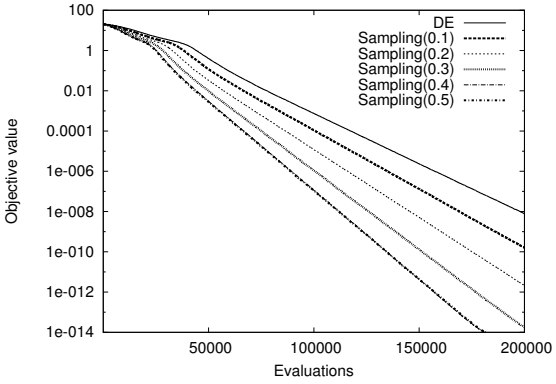

Fig. 14. The graph of f_{10}

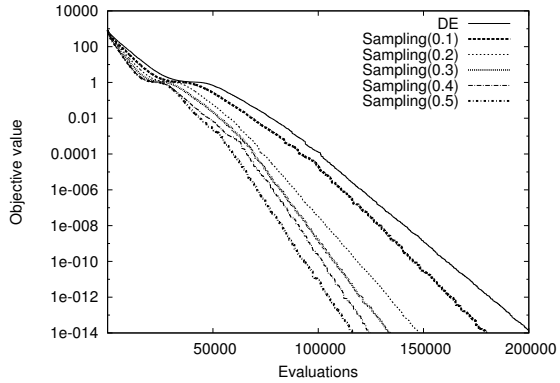

Fig. 15. The graph of f_{11}

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

[6] E. n. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 2006, pp. 485–492.

[7] T. Takahama and S. Sakai, "Solving nonlinear optimization problems by differential evolution with a rotation-invariant crossover operation using Gram-Schmidt process," in *Proc. of Second World Congress on Nature and Biologically Inspired Computing (NaBIC2010)*, Dec. 2010, pp. 533–540.

[8] 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.

[9] 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.

[10] S. Kobayashi, "The frontiers of real-coded genetic algorithms," *Journal of Japanese Society for Artificial Intelligence*, vol. 24, no. 1, pp. 147–162, Jan. 2009, in Japanese.

[11] T. Takahama and S. Sakai, "Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites," in *Proc. of the 2006 IEEE Congress on Evolutionary Computation*, Jul. 2006, pp. 308–315.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

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

[18] 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.

[19] 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.