Converging and Moving Detection Using an Outward Vector Rate for Adaptive Differential Evolution

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

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

***Abstract*—The performance of Differential Evolution (DE) is affected by algorithm parameters, mutation strategies and so on. One of the most successful studies on controlling the parameters in DE is JADE (adaptive DE with optional external archive). In this study, we propose a method to detect that the population is converging or moving. The vectors from parent to child are classified into inward vectors and outward vectors, and the rate of the outward vectors is used to determine the search state. In order to improve search efficiency of JADE, if the population is converging, an algorithm parameter is adjusted to enhance the convergence. If the population is moving, the parameter is adjusted to enhance the movement. Also, the moving direction of the population is determined by the averaged outward vectors and the search in the direction is strengthened. The advantage of JADE with the proposed method is shown by solving thirteen benchmark problems.**

***Index Terms*—evolutionary state estimation, outward vector rate, moving direction, adaptive differential evolution, differential evolution**

I. Introduction

Optimization problems are very important and frequently appear in the real world. There exist many studies on solving optimization problems using population-based optimization algorithms (POAs) [1] such as evolutionary algorithms (EAs) [2] and particle swarm optimization (PSO) [3]. POAs search for an optimum solution using a population with multiple candidate solutions, and usually use stochastic operations to generate new candidate solutions.

In this study, we focus on estimating the search state of the population such as converging and moving. This kind of estimation is called evolutionary state estimation in [4] and optimization state estimation in [5]. The goal is to improve search efficiency by dynamically updating the search mode based on the estimated state. There are several states in the process of searching for an optimal or near optimal solution by POAs. In this study, we propose the following classification of the states.

- Moving: If the population is far from the optimal solution, the population moves toward the optimal solution.
- Examining: If the population is a little far from the optimal solution or if the optimization problem has a multimodal landscape, it is necessary to examine within and around the population to determine a promising area where the optimal solution exists. It is an intermediate state between moving and converging.
- Converging: When the population reaches the promising area, the population converges to find a highly accurate solution.

In this study, we propose a simple method to detect that the population is converging or moving. In order to detect the state, it is necessary to estimate the direction in which the population is heading. It is assumed that there is a one-to-one correspondence between an original candidate solution and a generated candidate solution. Differential Evolution (DE) [6], [7] and PSO have such correspondence. In other algorithms that do not have the correspondence, this method can be applied by devising a method for creating the correspondence. In the following, the original candidate solution will be referred to as the parent, and the generated candidate solution will be referred to as the child. When the child is better than the parent, the vector from the parent to the child is obtained. If the vector approaches the center of gravity of the population, we call it an inward vector, and if it moves away from the center of gravity, we call it an outward vector. When the population is converging, the population is assumed to converge in the direction of the center of gravity, so the ratio of the inward vectors becomes high. In contrast, when the population moves toward the optimal solution that is far away from the center of gravity, it is assumed that about half candidate solutions, which are far from the optimal solution than the center of gravity, approaches the center of gravity, and the rest are away from the center of gravity. Figure 1 shows this situation.

Therefore, if the rate of outward vectors exceeds half, the state is Moving, and if the ratio of outward vectors is low, the state is Converging. As an index for the detection, we propose the outward vector rate (OVR), which is the rate of outward vectors among all vectors.

In this study, JADE (adaptive DE with optional external archive) [8], one of DE algorithms with a one-to-one correspondence between the parent and the child, is employed

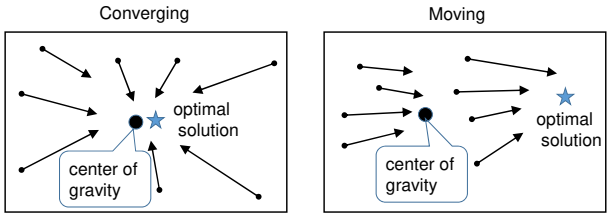

Fig. 1. Inward and outward vectors in Converging and Moving states.

as the optimization algorithm to which the proposed method is applied. The performance of DE is affected by algorithm parameters and mutation strategies. One of the most successful studies of adaptive DE that adjusts parameters adaptively is JADE. The advantage of JADE with the proposed method is shown by solving thirteen benchmark problems.

In Section 2, optimization problems, DE and JADE are briefly explained. Related works are described in Section 3. In Section 4, OVR and JADE with OVR are proposed. Conclusions are described in Section 5.

II. Optimization by Differential Evolution

A. Optimization Problems

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

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & l_j \leq x_j \leq u_j,\ j = 1, \ldots, D, \end{array} \tag{1}$$

where $\boldsymbol{x} = (x_1, x_2, \cdots, x_D)$ is a D dimensional vector and $f(\boldsymbol{x})$ is an objective function. The function f is a nonlinear real-valued function. Values l_j and u_j are the lower bound and the upper bound of x_j, respectively.

B. Differential Evolution

DE is an evolutionary algorithm proposed by Storn and Price [6], [7]. DE has been successfully applied to the optimization problems including non-linear, non-differentiable, non-convex and multimodal functions. It has been shown that DE is fast and robust to these functions [9].

A typical DE algorithm called DE/rand/1/bin is as follows:

1) An initial population $P = \{\boldsymbol{x}_i, i = 1, 2, \cdots, N\}$ is generated randomly in search space, where N is the population size and P is the population.
2) If a predefined condition is satisfied, the algorithm is terminated.
3) Each individual $\boldsymbol{x}_i$ is selected as a parent. A mutant vector $\boldsymbol{m}_i$ is generated as follows:

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

where three numbers $r1$, $r2$ and $r3$ are chosen randomly from $\{1, 2, \cdots, N\}$ without overlapping i and each other. F is a scaling factor and $\boldsymbol{x}_{r1}$ is called as the base vector. A trial vector (child) $\boldsymbol{v}_i$ is generated from $\boldsymbol{x}_i$ and $\boldsymbol{m}_i$ using the binomial crossover as follows:

$$v_{ij} = \begin{cases} m_{ij} & \text{if } j = j_{rand} \text{ or } u(0,1) < CR \\ x_{ij} & otherwise \end{cases} \tag{3}$$

where j_{rand} is a randomly selected integer in $[1, D]$ and $u(0, 1)$ is a uniform random number in $[0, 1]$. CR is a crossover rate. If the child is better than the parent, the child survives. Otherwise the parent survives. Go back to 3) until all individual are selected as parents.

4) The population P is replaced by the survivors. Go back to 2).

C. 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 two probability density functions (PDFs), 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 two PDFs as follows:

$$F_i \sim C(\mu_F, \sigma_F) \tag{4}$$

$$CR_i \sim N(\mu_{CR}, \sigma_{CR}^2) \tag{5}$$

where F_i is a random variable according to a Cauchy distribution $C(\mu_F, \sigma_F)$ with a location parameter μ_F and a scale parameter σ_F=0.1. CR_i is a random variable according to a normal distribution $N(\mu_{CR}, \sigma_{CR}^2)$ 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 $F_i > 1$ or regenerated if $F_i \leq 0$. The location μ_F and the mean μ_{CR} are updated as follows:

$$\mu_F = (1 - c)\mu_F + cS_{F^2}/S_F \tag{6}$$

$$\mu_{CR} = (1 - c)\mu_{CR} + cS_{CR}/S_N \tag{7}$$

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" as follows: A mutation vector is generated as follows:

$$\boldsymbol{m}_i = \boldsymbol{x}_i + F_i(\boldsymbol{x}_{pbest} - \boldsymbol{x}_i) + F_i(\boldsymbol{x}_{r2} - \widetilde{\boldsymbol{x}}_{r3}) \tag{8}$$

where $\boldsymbol{x}_{pbest}$ is a randomly selected individual from the top $100p\%$ individuals. $\widetilde{\boldsymbol{x}}_{r3}$ is selected randomly from the current population in case of "without an archive", and is selected randomly from the union of the current population and an archive in case of "with an archive". In many cases, JADE with an archive which has same archive size as the population size is used. The archive is initialized to be empty. Defeated parents by the children are added to the archive. If the number of archived individuals exceeds the archive size, randomly selected individuals are removed to keep the archive size.

In order to satisfy bound constraints, a child that is outside of the search space is moved into the inside of the search space. In JADE, each outside element of the child is set to be the middle between the corresponding boundary and the element of the parent as follows:

$$v_{ij} = \begin{cases} \frac{1}{2}(l_j + x_{ij}) & \text{if } v_{ij} < l_j \\ \frac{1}{2}(u_j + x_{ij}) & \text{if } v_{ij} > u_j \end{cases} \tag{9}$$

This operation is applied after a child is generated by JADE operations.

III. Related Works

A. Studies on Algorithm Parameters

The performance of DE is affected by algorithm parameters such as F, CR and N, and by mutation strategies. 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 such as CoDE [10].
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 such as FADE [11], DESFC [12], and LMDE [13], [14].
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 such as SaDE [15], JADE [8], SHADE [16], CADE [17] and jSO [18]. The self-adaptation, where parameters are contained in individuals and are evolved by applying evolutionary operators to the parameters, is included in this category such as DESAP [19] and jDE [20].
4) hybrid control: Plural control methods in different categories are combined to adjust algorithm parameters such as ADEGL [21] and JADEadm [22]. Our proposed method is included in this category.

B. Studies on Evolutionary State Estimation

Evolutionary state estimation (ESE) is strongly related to the balance control between exploitation and exploration that has been studied for a long time [23]. In the control, the population diversity is observed. When the diversity is large like in case of the initial population, the search mode is set to exploitation and the vicinity of promising solutions are searched. When the population diversity is small, the mode is set to exploration and a new area is searched. For example, in [24], the normalized distance from the center of gravity of the population is adopted as the diversity measure. If the measure exceeds the threshold d_{high}, the search mode is set to exploitation and selection and a crossover operation are performed. If the measure is less than the threshold d_{low}, the mode is set to exploration and a mutation operation is performed.

Various measures have been proposed as diversity measures. The distance-based diversity measures are as follows [25], where $||\boldsymbol{x} - \boldsymbol{y}||$ is the distance between the two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, and the Euclidean distance is often used.

- Diameter: Maximum distance in the population $\max_{i \neq j \in [1,N]} ||\boldsymbol{x}_i - \boldsymbol{x}_j||$.
- Radius: Maximum distance from the center of gravity of the population $\max_{i \neq j \in [1,N]} ||\boldsymbol{x}_i - \boldsymbol{g}||$, $\boldsymbol{g} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i$.
- Normalized average distance: Average distance from the center of gravity of the population normalized by the size of the search space $\frac{1}{LN}\sum_{i=1}^{N} ||\boldsymbol{x}_i - \boldsymbol{g}||$. L is the size of the search space $L = ||\boldsymbol{u} - \boldsymbol{l}||$.

However, even if normalized, the threshold is affected by the population size and so on. The issue is that it is difficult to select proper thresholds.

ESE does not aim to observe diversity, but to estimate the search state of the population. In [4], the state is classified into four states: Convergence, Exploitation, Exploration, and Jumping-out. The positional relationship between the best individual and other individuals is used for the classification. Assuming that the best individual is closest to other individuals when the population is converging, and the best individual is farthest from other individuals when the population is moving, and the following measure is proposed.

$$d_i = \frac{1}{N-1}\sum_{j=1, j\neq i}^{N} ||\boldsymbol{x}_i - \boldsymbol{x}_j|| \tag{10}$$

$$DBR = \frac{d_{best} - d_{\min}}{d_{\max} - d_{\min}} \tag{11}$$

where d_i is the average distance between $\boldsymbol{x}_i$ and other individuals, d_{best} is the average distance between the best individual and other individuals, $d_{\min}$ is the minimum value of d_i and $d_{\max}$ is the maximum value of d_i. If DBR is close to 0, the state is classified as Convergence, and if DBR is close to 1, the state is classified as Jumping-out. The time complexity is $O(N^2D)$ because the distances between all individuals are calculated. Since N is usually set to a multiple of D, $O(N^2D) = O(N^3)$. It is thought that an measure with a small computational complexity is necessary.

In [5], the following measure IOS (Indicator of the Optimization State) is proposed to probabilistically classify the search state into two states: Exploitation and Exploration based on the distances to the best individual and the function values.

$$IOS = \sum_{i=1}^{N} ||f_i - d_i|| \tag{12}$$

$$\overline{IOS} = \frac{IOS - IOS_{\min}}{IOS_{\max} - IOS_{\min}} \tag{13}$$

where d_i is the rank of the distance between the individual $\boldsymbol{x}_i$ and the best individual, and the closer the distance, the smaller the rank. f_i is the rank of the function value, and the better the value, the smaller the rank. $IOS_{\min}$ is the minimum value of IOS when the ranks of the distance and the function value are the same (IOS=0). $IOS_{\max}$ is the maximum value of IOS when the ranks of the distance and the function value are opposite ($f_i + d_i = N + 1$). The state is classified as Exploration with the probability of IOS and as Exploitation with the probability of $1 - IOS$. If the best individual is near the optimal solution and the function value becomes worse as the distance increases, IOS becomes small. If the population is far from the optimal solution and the best individual is moving to the optimal solution as the leader, and if the function value becomes worse as the distance increases,

IOS becomes small, too. It is thought that it is difficult to classify Convergence and Jumping-out.

In this study, we focus on the vectors from parent to child and the state is classified using the outward vector rate.

IV. Proposed method

A. Outward Vector Rate

The outward vector rate (OVR) is defined by the rate of outward vectors. If the movement vector from a parent to the child is in the direction closer to the center of gravity of the population, it is classified as an inward vector, and if it is farther away, it is classified as an outward vector. The process of calculating OVR is as follows:

1) Obtain the center of gravity $\boldsymbol{g}$ of the population. Set the counters $n_{\rm in}$ and $n_{\rm out}$ of the inward and outward vectors to 0.
2) For each parent $\boldsymbol{x}_i$ and the child $\boldsymbol{v}_i$, if the child is not better than the parent, go to 5).
3) Obtain the distances between $\boldsymbol{x}_i$ ($\boldsymbol{v}_i$) and the center of gravity as d_{xg} (d_{vg}). The movement vector is given by $\boldsymbol{v}_i - \boldsymbol{x}_i$.
4) If $d_{vg} < d_{xg}$, the movement vector is an inward vector and $n_{\rm in}$ is incremented by 1. If $d_{vg} > d_{xg}$, the movement vector is an outward vector and $n_{\rm out}$ is incremented by 1.
5) Go back to 2) until all individuals have been processed.
6) Obtain OVR using the following equation:

$$OVR = \frac{n_{\rm out}}{n_{\rm in} + n_{\rm out}} \tag{14}$$

To avoid abrupt changes in OVR, the exponentially smoothed moving average with the smoothing constant 0.5 is used. The time complexity is $O(ND) = O(N^2)$ because the distance between each individual and the center of gravity is calculated and N is usually determined in proportion to D.

B. Evolutionary State Estimation and Parameter Control

Based on OVR, the search state is classified into Converging, Examining and Moving. This corresponds to Convergence, Exploitation/Exploration and Jumping-out, respectively. In preliminary experiments, we optimized the benchmark functions and observed changes in OVR. It was found that Converging and Moving were relatively easy to determine, but Exploitation and Exploration were difficult. Therefore, in this study, parameter control is applied in Converging and Moving states, and the original control of JADE is applied in other states. The relationship between OVR values and the states and parameter control are as follows:

- $OVR \in$ [0,0.1): The state is judged as "strongly Converging", and the value of F_i is multiplied by 0.9 to speed up the convergence.
- $OVR \in$ [0.1,0.2): The state is judged as "weakly Converging", and the value of F_i is multiplied by 0.975 to slightly speed up the convergence.
- $OVR \in$ [0.4,0.5): The state is judged as "weakly Moving", The value of F_i is not modified, but the movement is accelerated using the averaged outward vectors, which will be explained later.
- $OVR \in$ [0.5,0.6): The state is judged as "Moving", and the value of F_i is multiplied by 1.025 to keep the population diversity and slightly accelerate the moving speed.
- $OVR \in$ [0.6,1]: The state is judged as "strongly Moving", and the value of F_i is multiplied by 1.1 to strengthen the diversity and accelerate the moving speed.

Since the value of OVR is not determined in the first generation, this control is applied after the first generation.

C. Acceleration of Movement

When the search state is Moving, there is a risk that the population will move very slowly and the efficiency of the search becomes very low, or that the population will lose diversity and fall into a local optimal solution. To avoid this, the speed of movement to promising area is accelerated by adding a movement vector in the direction of the average of the outward vectors to the child. Let $\boldsymbol{M}$ be the average of the outward vectors and the child is updated as follows:

$$\boldsymbol{M} = \frac{\sum_{i=1}^{N}\{\boldsymbol{v}_i - \boldsymbol{x}_i \text{ which is an outward vector}\}}{\rm n_{out}} \tag{15}$$

$$\boldsymbol{v}_i = \boldsymbol{v}_i + \alpha \boldsymbol{M} \tag{16}$$

The acceleration coefficient α is 0.2 for "strongly Moving", 0.1 for "Moving", and 0.05 for "weakly Moving". In order to avoid abrupt changes in $\boldsymbol{M}$, the exponentially smoothed moving average with the smoothing constant 0.5 is used.

D. Algorithm

The algorithm of the proposed method JADEovr (improved JADE using OVR) can be described as follows:

Step0 Parameter setup. The size of the archive N_A is N.

Step1 Initialization of the individuals. $P = \{\boldsymbol{x}_i | i = 1, 2, \cdots, N\}$ are generated randomly in the search space and form an initial population. The archive A is made empty. Initial values OVR=0 and $\boldsymbol{M} = \boldsymbol{0}$.

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

Step3 Initialization for each generation. The list of success cases S is made empty. The counters are initialized as $n_{\rm in} = n_{\rm out} = 0$. The sum of the outward vectors is initialized as $\boldsymbol{M}_{\rm out} = \boldsymbol{0}$.

Step4 JADE operation using OVR. For each individual $\boldsymbol{x}_i$, F_i is generated according to Eq.(4) and CR_i is generated according to Eq.(5). According to OVR F_i is modified after the first generation. $\boldsymbol{x}_{pbest}$ is selected randomly from top p% individuals. $\boldsymbol{x}_{r2}$ is selected randomly excluding $\boldsymbol{x}_i$. $\boldsymbol{x}_{r3}$ are randomly selected from the union of P and A excluding $\boldsymbol{x}_i$ and $\boldsymbol{x}_{r1}$. JADE operation is executed and a child is generated. According to OVR, the acceleration coefficient α is determined and the child is modified using Eq.(16) after the first generation.

Step5 Survivor selection. If the child is better than the parent, or in a success case, the child becomes a survivor. The successful combination of parameter values (F_i, CR_i) is added to S. Defeated parent is added to A. Otherwise, the parent $\boldsymbol{x}_i$ becomes a survivor.

Step6 Determining inward or outward. If the child is better than the parent, the distances between $\boldsymbol{x}_i$ ($\boldsymbol{v}_i$) and the center of gravity is obtained as d_{xg} (d_{vg}). If the vector $\boldsymbol{v}_i - \boldsymbol{x}_i$ is a inward vector, $n_{\rm in}$ is incremented by 1. If the vector is an outward vector, $n_{\rm out}$ is incremented by 1 and the vector is added to $\boldsymbol{M}_{\rm out}$. Go back to Step 4 until all individuals are processed.

Step7 Resizing the archive. If the size of the archive exceeds N_A, randomly selected elements in A are deleted until the size of A becomes N_A.

Step8 Learning parameters. The means of the scaling factor μ_F and the means of crossover rate μ_{CR} are updated using S according to Eqs. (6) and (7).

Step9 Updating OVR and $\boldsymbol{M}$. In the first generation, OVR and $\boldsymbol{M}$ are determined according to Eq.(14) (using $\boldsymbol{M}_{\rm out}$) and Eq.(15), respectively. The exponentially smoothed moving average with the smoothing constant 0.5 is used to update OVR and $\boldsymbol{M}$ after the first generation. Go back to Step2.

Figure 2 shows the pseudo-code of JADEovr. Lines starting with '+' shows the modified lines from original JADE.

V. Numerical Experiments

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

A. Test Problems

The 13 scalable benchmark functions [8] are solved. Every function has an optimal objective value 0. 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. 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.

Experimental conditions are same as JADE as follows: Population size $N = 100$, initial mean for scaling factor μ_F = 0.5 and initial mean for crossover rate μ_{CR} = 0.5, the pbest parameter p = 0.05, and the learning parameter c = 0.1. The archive size N_A is same as N.

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

B. Experimental Results

Table I show the experimental results on JADE, JADE+F (JADE with control of F_i only), JADE+M (JADE with acceleration of movement only) and JADEovr. The mean value,

```
JADEovr()
{
  μF=μCR=0.5; σF=σCR=0.1; NA=N; // archive size
// Initialize a population
  P=N individuals generated randomly in the search space;
  FE=N;
  A=∅;
+ OVR=0; M=0; // moving average of outward vectors
  for(t=1; FE < FEmax; t++) {
    S=∅; // make the list of success cases empty
+   nin=nout=0; Mout=0;  // sum of outward vectors
+   g=center of gravity of P;
    for(i=1; i ≤ N; i++) {
// JADE operations for each individual
      CRi = μCR + N(0, σ²CR);
      truncate CRi to [0,1];
      do {
        Fi=μF + C(0, σF);
      } while(Fi ≤ 0);
      if(Fi > 1) Fi = 1;
+     if(t>1) modify Fi according to OVR;
      xpbest=randomly selected from top 100p% in P;
      xr2=randomly selected from P (r2 ≠ i);
      xr3=randomly selected from P ∪ A (r3 ∉ {i, r2});
      mi=xi+Fi(xpbest − xi)+Fi(xr2 − xr3);
      vi=generated from xi and mi by binomial crossover;
+     if(t>1) {
+       set α according to OVR;
+       vi+=αM; // enhance movement
+     }
      FE=FE+1;
// Survivor selection
      if(f(vi) < f(xi)) {
        zi=vi;
        S=S ∪ {(Fi, CRi)}; // add a success case
        A=A ∪ {xi};
+       dxg=||xi − g||; dvg=||vi − g||;
+       if(dvg<dxg) nin++;
+       else if(dvg>dxg) {
+         nout++; Mout+=vi−xi;
+       }
      }
      else
        zi = xi;
    }
    P = {zi}
    while(|A|>NA) // resize the archive
      remove a randomly selected element from A;
// Learning parameters
    if(|S| > 0) {
      μF = (1 − c)μF + c∑Fi∈S F²i/∑Fi∈S Fi;
      μCR = (1 − c)μCR + c∑CRi∈S CRi/|S|;
    }
+   if(nin+nout>0) {
+     if(t==1) {
+       OVR=nout/(nin + nout); if(nout>0) M=Mout/nout;
+     } else {
+       OVR=0.5OVR+0.5nout/(nin + nout);
+       if(nout>0) M=0.5M+0.5Mout/nout;
+     }
+   }
  }
}
```

Fig. 2. The pseudo-code of the proposed method JADEovr

the standard deviation, and the median value of best objective values in 50 runs are shown for each function. The median value is shown under the mean value. The maximum number of function evaluations is selected for each function and is shown in column labeled $FE_{\max}$. Since the variability of each trial is not small and the reliability of the mean value is not high, the best median value among algorithms is highlighted. Also, Wilcoxon signed rank test is performed and the result for each function is shown on the right side of the median value in parentheses. Symbols '+', '−' and '=' are shown when each algorithm is significantly better than JADE, is significantly worse than JADE, and is not significantly different from JADE, respectively. Symbols '++' and '−−' show that the significance level is 1% and '+' and '−' show that the significance level is 5%. Also, '==' show that completely same results are obtained because $\boldsymbol{M}$ is not used for optimization by JADE+M in some problems.

From Table I, JADEovr attained the best median results in 8 functions f_1, f_4, f_6, f_8 and f_{10}–f_{13} out of 13 functions. JADE+F attained the best median results in 8 functions f_1, f_2, f_6 and f_9–f_{13}. JADE+M attained the best median results in 3 functions f_3, f_5 and f_7. JADE attained no best median result.

JADEovr attained significantly better results than JADE in 12 functions except for f_7 and attained no significantly worse result than JADE. The JADE+F attained significantly better results than JADE in 9 functions except for f_3–f_5 and f_7, and attained no significantly worse result. The JADE+M attained significantly better results than JADE in 3 functions and significantly worse result in f_9. It is thought that the parameter control of F_i is very effective to many functions and the movement acceleration is effective to the functions f_3–f_5. Therefore, the combination of the parameter control of F_i and the movement acceleration according to OVR is very effective and can improve the performance of JADE.

Figure 3 shows the change of OVR over the number of function evaluations for JAVEovr and the change of μ_F for JADEovr and JADE in the unimodal function f_1, the function with ridge structure f_5, the unimodal function with noise f_7, the multimodal function f_8, and the strongly multimodal function f_9.

As for the unimodal function f_1, the proper state would be always Converging because uniformly distributed individuals in the search space approach to the center of the search space. In the figure, OVR is very small and less than 0.1 (strongly Converging) in very early generations, and then is in the rage of [0.1,0.2] (weakly Converging). The values of μ_F in JADEovr are always smaller than those in JADE. It is thought that OVR can detect the proper state and μ_F is modified to smaller values to speed up the convergence.

As for the function with ridge structure f_5, usually the population approaches the origin first and passes through a narrow ridge to reach the optimal solution and the proper state would be Converging and then Moving. OVR is small and less than 0.2 (strongly Converging to weakly Converging) in early generations, is increasing and beyond 0.2 (Examining), and is

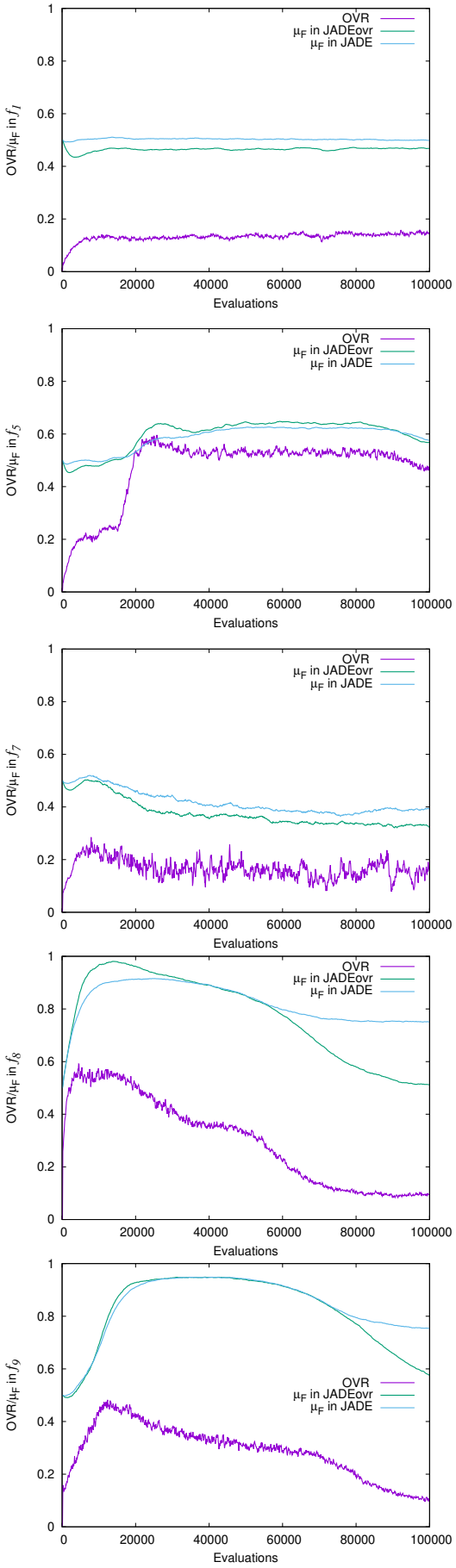

Fig. 3. The graphs of OVR and μ_F in JADEovr and JADE.

TABLE I
EXPERIMENTAL RESULTS ON 13 FUNCTIONS

Func	FEmax	JADE	JADE+F	JADE+M	JADEovr
f_1	150,000	6.50e-58 ± 4.5e-57 1.13e-63	3.67e-73 ± 1.5e-72 **2.29e-75** (++)	6.50e-58 ± 4.5e-57 1.13e-63 (==)	3.67e-73 ± 1.5e-72 **2.29e-75** (++)
f_2	200,000	2.21e-24 ± 1.2e-23 6.73e-33	1.72e-43 ± 8.2e-43 **5.47e-48** (++)	2.26e-24 ± 1.2e-23 6.73e-33 (=)	1.72e-43 ± 8.2e-43 7.02e-48 (++)
f_3	500,000	2.29e-83 ± 1.1e-82 7.25e-86	2.51e-82 ± 9.8e-82 2.47e-85 (=)	2.59e-85 ± 1.5e-84 **1.08e-88** (++)	9.16e-85 ± 3.9e-84 8.83e-88 (++)
f_4	500,000	1.58e-62 ± 4.3e-62 1.15e-63	1.32e-62 ± 5.8e-62 4.24e-64 (=)	1.16e-63 ± 5.8e-63 3.99e-65 (++)	3.14e-64 ± 9.7e-64 **1.95e-65** (++)
f_5	150,000	2.39e-01 ± 9.5e-01 1.82e-19	2.39e-01 ± 9.5e-01 5.15e-20 (=)	1.59e-01 ± 7.8e-01 **1.37e-21** (++)	2.39e-01 ± 9.5e-01 3.26e-21 (++)
f_6	10,000	4.92e+00 ± 1.4e+00 5.00e+00	2.60e-01 ± 5.2e-01 **0.00e+00** (++)	4.92e+00 ± 1.4e+00 5.00e+00 (==)	2.60e-01 ± 5.2e-01 **0.00e+00** (++)
f_7	300,000	6.24e-04 ± 2.5e-04 5.65e-04	6.13e-04 ± 2.2e-04 5.98e-04 (=)	5.59e-04 ± 2.2e-04 **5.17e-04** (=)	5.78e-04 ± 2.1e-04 5.79e-04 (=)
f_8	100,000	7.11e+00 ± 2.8e+01 3.70e-05	2.37e+00 ± 1.7e+01 4.10e-08 (++)	4.74e+00 ± 2.3e+01 3.78e-05 (=)	2.37e+00 ± 1.7e+01 **3.19e-08** (++)
f_9	100,000	1.34e-04 ± 7.2e-05 1.18e-04	2.35e-05 ± 2.2e-05 **1.81e-05** (++)	1.54e-04 ± 7.7e-05 1.44e-04 (−)	3.53e-05 ± 3.1e-05 2.83e-05 (++)
f_{10}	50,000	2.87e-09 ± 4.8e-09 1.88e-09	3.02e-11 ± 1.8e-11 **2.96e-11** (++)	2.87e-09 ± 4.8e-09 1.88e-09 (==)	3.02e-11 ± 1.8e-11 **2.96e-11** (++)
f_{11}	50,000	1.71e-07 ± 1.2e-06 4.68e-12	5.45e-04 ± 2.2e-03 **3.52e-14** (+)	1.71e-07 ± 1.2e-06 4.68e-12 (=)	5.45e-04 ± 2.2e-03 **3.52e-14** (+)
f_{12}	50,000	3.20e-16 ± 1.1e-15 1.43e-17	5.63e-20 ± 2.4e-19 **9.41e-21** (++)	3.20e-16 ± 1.1e-15 1.43e-17 (==)	5.63e-20 ± 2.4e-19 **9.41e-21** (++)
f_{13}	50,000	7.98e-16 ± 1.4e-15 3.03e-16	1.91e-19 ± 4.4e-19 **4.45e-20** (++)	7.98e-16 ± 1.4e-15 3.03e-16 (==)	1.91e-19 ± 4.4e-19 **4.45e-20** (++)
+		—	9	3	12
=		—	4	9	1
−		—	0	1	0

almost in the rage of [0.4,0.6] (between weakly Moving and strongly Moving). The values of μ_F in JADEovr are always greater than those in JADE except for early generations. It is thought that OVR can detect the state change and μ_F is modified to greater values to keep diversity and strengthen the movement.

As for the unimodal but noisy function f_7, the proper state is always Converging because uniformly distributed individuals approach to the center of the search space as well as in f_1. OVR is almost in the range of [0.1,0.2] (weakly Converging) but sometimes beyond 0.2 (Examining). The values of μ_F in JADEovr are always smaller than those in JADE. It is thought that OVR can often detect the proper state and μ_F is modified to smaller values to speed up the convergence. But the control of μ_F is not enough to outperform the control of JADE because Examining state is sometimes detected due to the noise.

As for the multimodal function having the optimal solution near the boundary of search space f_8, the proper state is Moving then Converging, because the population moves to near the boundary and then converges to the optimal solution. OVR is very large and greater than 0.4 (strongly Moving to weakly Moving) except for very early generations, and is decreasing to less than 0.4 (Examining) and to less then 0.2 (Converging). The values of μ_F in JADEovr are greater than those in JADE in the first half generations and are less than those in JADE in the second half of generations. It is thought that OVR can detect the state change and μ_F is modified to greater values to keep diversity and strengthen the movement first and then is modified to smaller values to speed up the convergence.

As for the strongly multimodal function f_9, the proper state would be Examining then Converging, because the population examines various near optimal solutions and converges to the optimal solution after the population reaches a valley including the optimal solution. OVR is increasing from 0.15 (Converging to Examining), is beyond 0.4 (Moving), and is decreasing to less than 0.2 (Examining to Converging). The values of μ_F in JADEovr are almost same as those in JADE and are smaller than those in JADE in later generations. It is thought that OVR can detect the state change from Examining to Converging but sometimes detects Moving incorrectly. The reason why JADEovr outperformed JADE is that after the population reaches the valley including the optimal solution, it converges to the optimal solution more rapidly than JADE due to detection of Converging state.

It is thought that OVR can detect Converging and Moving states and the scaling factor is controlled almost correctly.

VI. CONCLUSIONS

In this study, the method of detecting Converging and Moving states using the outward vector rate is proposed. Also, a dynamic modification of the scaling factor and the acceleration of movement are also proposed to improve the performance of JADE. The proposed method was applied to the optimization of various 13 functions including unimodal functions, ridge functions and multimodal functions. It was shown that the outward vector rate can classify the search state

correctly, the modification of the scaling factor can speed up the convergence or enhance the movement, and the movement acceleration can improve the performance in Moving state. Also, it was shown that the proposed method JADEovr was very efficient compared with JADE.

In the future, we will investigate the threshold values for OVR to classify the search state and the algorithm parameter α to enhance the movement in detail. Also, we will introduce the ε constrained method into JADEovr to solve constrained optimization problems. Although optimization was successful with the multimodal functions used in this study, Fig. 1 assumes nearly unimodal functions. In strongly multimodal functions, each individual may move toward a different local optimum, so it is necessary to improve OVR to be robust to such functions. The change of the function values was confirmed, but the behavior of the population has not been sufficiently analyzed, so it is necessary to perform a detailed analysis.

Acknowledgment

This study is supported by JSPS KAKENHI Grant Numbers 19K04916 and 20K11977.

REFERENCES

[1] A. E. Babalola, B. A. Ojokoh, and J. B. Odili, "A review of population-based optimization algorithms," in *2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)*, 2020, pp. 1–7.

[2] A. E. Eiben and J. E. Smith, *What Is an Evolutionary Algorithm?* Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 25–48.

[3] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*. San Francisco: Morgan Kaufmann, 2001.

[4] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 6, pp. 1362–1381, 2009.

[5] W.-J. Yu, M. Shen, W. neng Chen, Z. hui Zhan, Y.-J. Gong, Y. Lin, O. Liu, and J. Zhang, "Differential evolution with two-level parameter adaptation," *IEEE Transactions on Cybernetics*, vol. 44, no. 7, pp. 1080–1099, July 2014.

[6] R. Storn and K. Price, "Minimizing the real functions of the ICEC'96 contest by differential evolution," in *Proc. of the International Conference on Evolutionary Computation*, 1996, pp. 842–844.

[7] 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.

[8] 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.

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

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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, Apr. 2009.

[16] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proc. of the 2013 IEEE Congress on Evolutionary Computation*, 2013, pp. 71–78.

[17] T. Takahama and S. Sakai, "An adaptive differential evolution considering correlation of two algorithm parameters," in *Proc. of the Joint 7th International Conference on Soft Computing and Intelligent Systems and 15th International Symposium on Advanced Intelligent Systems (SCIS&ISIS2014)*, Dec. 2014, pp. 618–623.

[18] J. Brest, M. S. Maučec, and B. Bošković, "Single objective real-parameter optimization: Algorithm jso," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 1311–1318.

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

[20] 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, Dec. 2006.

[21] T. Takahama and S. Sakai, "An adaptive differential evolution with learning parameters according to groups defined by the rank of objective values," in *Proc. of the Eighth International Conference on Swarm Intelligence (ICSI2017)*, Jul. 2017, pp. 411–419.

[22] T. Takahama and S. Sakai, "Adaptive directional mutation for an adaptive differential evolution algorithm," in *Proc. of Joint 11th International Conference on Soft Computing and Intelligent Systems and 21th International Symposium on Advanced Intelligent Systems*, Dec. 2020, pp. 262–268.

[23] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM computing surveys (CSUR)*, vol. 45, no. 3, pp. 1–33, 2013.

[24] R. K. Ursem, "Diversity-guided evolutionary algorithms," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2002, pp. 462–471.

[25] O. Olorunda and A. P. Engelbrecht, "Measuring exploration/exploitation in particle swarms using swarm diversity," in *2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence)*. IEEE, 2008, pp. 1128–1134.