ICIC Express Letters ICIC International ©2024 ISSN 1881-803X
Volume **18**, Number **5**, May **2024** pp. **469–475**

PERTURBATION OF ADAPTIVE DIFFERENTIAL EVOLUTION WITH OPTIONAL EXTERNAL ARCHIVE FOR CONTINUOUS OPTIMIZATION PROBLEM

Ryu Aishima and Michiharu Maeda*

Department of Computer Science and Engineering
Fukuoka Institute of Technology
3-30-1 Wajirohigashi, Higashi-ku, Fukuoka, Fukuoka 811-0295, Japan
*Corresponding author: maeda@fit.ac.jp

Received May 2023; accepted December 2023

Abstract. *This paper describes perturbation of adaptive differential evolution with optional external archive for continuous optimization problem. Perturbation is a phenomenon in which the motion by the contribution of a main force is disturbed by the contribution of other secondary forces. Adaptive differential evolution with optional external archive includes an approach that automatically adjusts the crossover rate and the scale factor and speeds up the convergence of the solution with a novel mutation strategy. In order to show the validity of our algorithm, experimental results are compared to existing algorithms.*

Keywords: Perturbation, Differential evolution, Optional external archive, Optimization problem

1. Introduction.

Metaheuristics have approximation frameworks against the background of increasing scale and complexity for real systems and combine experimental approaches for solving optimization problem [1]. Metaheuristics are preferred for fast and high-quality solutions [2]. Metaheuristics include such as real-coded genetic algorithm (RGA), particle swarm optimization (PSO), and differential evolution (DE). RGA adopts the real number vector as encoding and implements crossover for this coding [3]. PSO explores an optimal solution influenced by two factors. The one is the best position each particle has ever found, and the other is the best position found for all particles [4]. DE is an algorithm to explore the optimal solution that implements to repeat three operations: mutation, crossover, and selection [5]. DE is an effective algorithm for a number of optimization problems because of simple and efficient procedures. DE has been applied in various fields. A new real-coded modified differential evolution based automatic fuzzy clustering algorithm which automatically evolves the number of clusters as well as the proper partitioning from a data set has been proposed [6]. An adaptive differential evolution with multiple trial vectors for training artificial neural networks (ANNs) has been suggested [7]. For optimizing the global properties of segmentation by means of a global optimizer, differential evolution superpixel segmentation has been studied [8]. As the performance depends on parameters such as scale factor and crossover rate, there is an effective DE variant such as adaptive differential evolution with optional external archive (JADE). JADE automatically adapts parameters and implements mutation strategy called DE/current-to-pbest with optional archive and shows relatively superior performances for high dimensional problems [9]. By incorporating a perturbation in which a main force is disturbed by a subsidiary power, differential evolution with perturbation (DEP) has yielded significant results [10]. DEP tends to be able to explore a solution efficiently and to avoid to local optima by introducing perturbation.

DOI: 10.24507/icicel.18.05.469

In this paper, we present perturbation of adaptive differential evolution with optional external archive (PJADE). Perturbation has a physical phenomenon that the motion by the contribution of the main force is disturbed by the influence subsidiary power. Adaptive differential evolution with optional external archive includes an approach that automatically adjusts the crossover rate and the scale factor and speeds up the convergence of the solution with a novel mutation strategy. By introducing perturbation to adaptive differential evolution with optional external archive, our algorithm tends to be able to explore a solution efficiently and to avoid to local optima. In order to show the effectiveness of the proposed algorithm, experiment results are compared to existing algorithms by CEC2017 benchmark functions. PJADE shows equal to or better results than the existing algorithms.

This paper is organized as follows. Section 2 describes existing algorithms for DE and JADE. Section 3 presents PJADE. Section 4 describes numerical experiments and results. Finally, Section 5 summarizes conclusions.

2. **Existing Algorithm.**

2.1. **Differential evolution.**

DE is an algorithm that is simple and easy to implement. DE updates a new solution by repeating mutation, crossover, and selection. Mutation operates to generate mutant vector to use three solutions randomly from a set of candidate solution. Mutation is defined as follows:

$$v_{i,j} = x_{r1,j} + F \cdot (x_{r2,j} - x_{r3,j}) \tag{1}$$

where v is the mutant vector, i is number of solutions, and j is number of dimensions. x_{r1}, x_{r2}, and x_{r3} are selected solutions randomly from current population, $r1$, $r2$, and $r3$ are each another solution. F is the scale factor. Crossover operates to generate trial vector with the use of mutant vector and target vector. Crossover is defined as follows:

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } (rand_j \le C_r \text{ or } j = j_{rand}) \\ x_{i,j} & \text{otherwise} \end{cases} \tag{2}$$

where u is the trial vector, and x is the target vector. $rand_j$ is random number to generate from uniform distribution in $[0, 1]$, C_r is the crossover rate, and j_{rand} is the index to select randomly in $1, 2, \ldots, D$. D is the number of dimensions. Selection compares x_i with u_i, and the one with the better objective function value as the next generation solution. Selection is defined as follows:

$$x_i = \begin{cases} u_i & \text{if } (f(u_i) < f(x_i)) \\ x_i & \text{otherwise} \end{cases} \tag{3}$$

The pseudo code of DE is shown in Algorithm 1.

Algorithm 1. Differential evolution

```
Generate randomly initial solutions $x_i$ $(i = 1, 2, \ldots, N)$.
Calculate the fitness of initial solutions.
$FEs = N$.
while $FEs < MaxFEs$ do
    for $i = 1$ to $N$ do
        Randomly select three solutions $x_{r1}$, $x_{r2}$ and $x_{r3}$ from current population.
        Generate the mutant vector $v_i$ according to (1).
        Generate the trial vector $u_i$ according to (2).
        Calculate the fitness of the trial vector $u_i$ and $FEs = FEs + 1$.
        if $f(u_i) < f(x_i)$ then
            $x_i = u_i$
        end if
    end for
end while
```

2.2. **Adaptive differential evolution with optional external archive.** JADE is the algorithm to be effective DE variant. JADE automatically adjusts the parameters F and C_r with optional archive. Mutation operates to generate mutant vector to use the strategy called DE/current-to-$p$best. Mutation is defined as follows:

$$v_{i,j} = x_{i,j} + F_i \cdot \left(x^p_{best,j} - x_{i,j}\right) + F_i \cdot \left(x_{r1,j} - \tilde{x}_{r2,j}\right) \tag{4}$$

where x^p_{best} is the solution to be selected randomly from the top $N \times p$ ($p \in [0, 1]$) in the current population. x_{r1} is the solution to be selected randomly from $\mathbf{P}$, and $\mathbf{P}$ is the current population. $\tilde{x}_{r2}$ is the solution to be selected randomly from $\mathbf{P} \cup \mathbf{A}$, and $\mathbf{A}$ is a set of archived inferior solutions. F_i is the scale factor of each solution. Crossover operates to generate trial vector with the use of mutant vector and target vector. Crossover is defined as follows:

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } (rand_j \le CR_i \text{ or } j = j_{rand}) \\ x_{i,j} & \text{otherwise} \end{cases} \tag{5}$$

where CR_i is the crossover rate of each solution. CR_i of each individual x_i is independently generated according to a normal distribution of mean μCR and standard deviation 0.1. The mean μCR is initialized to be 0.5 and then updated at the end of each generation as follows:

$$CR_i = randn_i(\mu CR, 0.1) \tag{6}$$

$$\mu CR = (1 - c) \cdot \mu CR + c \cdot mean_A(S_{CR}) \tag{7}$$

where c is a positive constant between 0 and 1. S_{CR} is a set of all successful crossover rates CR_i's at generation t. $mean_A(\cdot)$ is the usual arithmetic mean. On the other hand, F_i of each individual x_i is independently generated according to a Cauchy distribution of location parameter μF and scale parameter 0.1. The location parameter μF is initialized to be 0.5 and then at the end of each generation as follows:

$$F_i = randc_i(\mu F, 0.1) \tag{8}$$

$$\mu F = (1 - c) \cdot \mu F + c \cdot mean_L(S_F) \tag{9}$$

where S_F is a set of all successful mutation factor F_i's at generation t, and $mean_L(\cdot)$ is calculated as follows:

$$mean_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \tag{10}$$

The pseudo code of JADE is shown in Algorithm 2.

3. Perturbation of Adaptive Differential Evolution with Optional External Archive.

In this section, we describe perturbation of adaptive differential evolution with optional external archive (PJADE). The solution with perturbation tends to be better than the current solution. The proposed algorithm introduces perturbation in the crossover operation. The solution with perturbation is defined as follows:

$$u'_{i,j} = \begin{cases} v_{i,j} + sIr_j & \text{if } (rand_j \le CR_i \text{ or } j = j_{rand}) \\ x_{i,j} + sIr_j & \text{otherwise} \end{cases} \tag{11}$$

where u' is the solution with perturbation. r is a vector to be generated from uniform distribution in $[-1, 1]$. s and I are calculated in the following formula.

$$s = a - 4aFEs/MaxFEs \tag{12}$$

$$I = (x_{\max} - x_{\min})/R \tag{13}$$

where a and R are constant parameters. FEs and $MaxFEs$ are the current and maximum number of function evaluation, respectively. $x_{\max}$ and $x_{\min}$ are maximum and minimum in the search range, respectively.

The pseudo code of PJADE is shown in Algorithm 3.

Algorithm 2. Adaptive differential evolution with optional external archive

```
Set $\mu CR = 0.5$, $\mu F = 0.5$ and $\mathbf{A} = \emptyset$.
Generate randomly initial solutions $x_i$ $(i = 1, 2, \ldots, N)$.
Calculate the fitness of initial solutions.
$FEs = N$.
while $FEs < MaxFEs$ do
    $S_F = \emptyset$, $S_{CR} = \emptyset$.
    for $i = 1$ to $N$ do
        Generate $CR_i$ and $F_i$ according to (6) and (8), respectively.
        Randomly select $x_{best}^p$ from the top $N \times p$ $(p \in [0, 1])$ in the current population.
        Randomly select $x_{r1}$ from current population $\mathbf{P}$.
        Randomly select $\tilde{x}_{r2}$ from $\mathbf{P} \cup \mathbf{A}$.
        Generate the mutant vector $v_i$ according to (4).
        Generate the trial vector $u_i$ according to (5).
        Calculate the fitness of the trial vector $u_i$ and $FEs = FEs + 1$.
        if $f(u_i) < f(x_i)$ then
            $x_i = u_i$; $x_i \to \mathbf{A}$; $CR_i \to S_{CR}$, $F_i \to S_F$.
        end if
        Randomly remove solutions from $\mathbf{A}$ so that $|\mathbf{A}| \leq N$.
        Update $\mu CR$ and $\mu F$ according to (7) and (9), respectively.
    end for
end while
```

Algorithm 3. Perturbation of adaptive differential evolution with optional external archive

```
Set $\mu CR = 0.5$, $\mu F = 0.5$ and $\mathbf{A} = \emptyset$.
Generate randomly initial solutions $x_i$ $(i = 1, 2, \ldots, N)$.
Calculate the fitness of initial solutions.
$FEs = N$.
while $FEs < MaxFEs$ do
    $S_F = \emptyset$, $S_{CR} = \emptyset$.
    for $i = 1$ to $N$ do
        Generate $CR_i$ and $F_i$ according to (6) and (8), respectively.
        Randomly select $x_{best}^p$ from the top $N \times p$ $(p \in [0, 1])$ in the current population.
        Randomly select $x_{r1}$ from current population $\mathbf{P}$.
        Randomly select $\tilde{x}_{r2}$ from $\mathbf{P} \cup \mathbf{A}$.
        Generate the mutant vector $v_i$ according to (4).
        if $FEs < MaxFEs/4$ then
            Generate the trial vector $u_i$ according to (5).
            Calculate the fitness of the trial vector $u_i$ and $FEs = FEs + 1$.
            Generate the trial vector with perturbation $u_i'$ according to (11).
            Calculate the fitness of the trial vector with perturbation $u_i'$ and $FEs = FEs + 1$.
            if $f(u_i') < f(u_i)$ then
                $u_i = u_i'$
            else
                $u_i = u_i$
            end if
        else
            Generate the trial vector $u_i$ according to (5).
            Calculate the fitness of the trial vector $u_i$ and $FEs = FEs + 1$.
        end if
        if $f(u_i) < f(x_i)$ then
            $x_i = u_i$; $x_i \to \mathbf{A}$; $CR_i \to S_{CR}$, $F_i \to S_F$.
        end if
        Randomly remove solutions from $\mathbf{A}$ so that $|\mathbf{A}| \leq N$.
        Update $\mu CR$ and $\mu F$ according to (7) and (9), respectively.
    end for
end while
```

4. **Numerical Example.** We compare our algorithm (PJADE) to real-coded genetic algorithm (RGA), particle swarm optimization (PSO), differential evolution (DE), adaptive differential evolution with optional external archive (JADE), and differential evolution with perturbation (DEP) by the CEC2017 benchmark functions. CEC2017 benchmark functions are constructed by 29 functions [11]. All test functions are minimization problems and define the same search ranges in $[-100, 100]^D$. F_1 to F_9 are the functions to be shifted and rotated. Especially, F_1 and F_2 are the unimodal functions, and F_3 to F_9 are the multimodal functions. F_{10} to F_{19} are hybrid functions that the variables are randomly divided into some subcomponents and then different basic functions are used for different subcomponents. F_{20} to F_{29} are composition functions that merge the properties of the sub-functions better and maintain continuity around the global/local optima. We set common parameters as follows: number of population $N = 20$, number of dimensions $D = 10$ and 50, maximum number of function evaluation $MaxFEs = 10000D$, and run times $R = 50$. Parameters of each algorithm are shown as follows: parameters $\alpha = 0.5$, $\beta = 0.35$ in RGA, inertia weight $w = 0.9 - 0.5FEs/MaxFEs$, weight parameters $c1 = c2 = 2.0$ in PSO, scale factor $F = 0.8$, crossover rate $C_r = 0.95$, constant parameter $c = 0.2$, $p = 0.2$ in JADE and PJADE, and constant parameter $a = 1.0$, $R = 200.0$ in DEP and PJADE.

Table 1. Experimental results in 10 dimensions

	RGA	PSO	DE	JADE	DEP	PJADE
F_1	2.06×10^{10}	4.97×10^7	1.10×10^2	1.81×10^2	**0.0**	1.74×10^2
F_2	5.39×10^4	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
F_3	2.93×10^3	1.00×10^1	1.54	3.44	**0.0**	2.59
F_4	2.52×10^4	5.35×10^1	8.23×10^1	1.51×10^2	1.86×10^1	$\mathbf{1.07 \times 10^1}$
F_5	1.64×10^{-2}	2.20×10^{-5}	6.17×10^{-4}	1.33×10^{-7}	1.01×10^{-5}	$\mathbf{4.59 \times 10^{-8}}$
F_6	2.48×10^4	2.50×10^1	4.28×10^1	$\mathbf{1.37 \times 10^1}$	3.89×10^1	1.43×10^1
F_7	5.20×10^2	3.90×10^1	5.50×10^1	5.50×10^1	5.11×10^1	$\mathbf{3.34 \times 10^1}$
F_8	1.03	$\mathbf{6.36 \times 10^{-1}}$	$\mathbf{6.36 \times 10^{-1}}$	$\mathbf{6.36 \times 10^{-1}}$	$\mathbf{6.36 \times 10^{-1}}$	$\mathbf{6.36 \times 10^{-1}}$
F_9	3.36×10^3	4.37×10^2	2.68×10^2	7.05×10^1	2.50×10^2	$\mathbf{1.68 \times 10^1}$
F_{10}	1.16×10^9	3.52×10^3	1.06×10^3	3.00×10^2	**3.07**	6.85
F_{11}	1.06×10^9	6.67×10^5	1.63×10^7	1.17×10^4	$\mathbf{6.43 \times 10^1}$	6.38×10^4
F_{12}	1.55×10^9	7.44×10^3	3.16×10^1	1.86×10^2	$\mathbf{3.01 \times 10^1}$	1.33×10^2
F_{13}	9.95×10^5	7.45×10^1	1.60×10^1	**4.12**	2.13×10^1	4.91
F_{14}	1.37×10^9	4.25×10^4	6.88×10^4	3.10×10^1	**2.26**	3.71
F_{15}	2.81×10^5	2.27×10^2	1.34×10^2	$\mathbf{7.34 \times 10^1}$	1.65×10^2	7.94×10^1
F_{16}	3.38×10^{11}	1.93×10^1	3.77×10^1	5.01×10^1	1.20×10^1	**1.86**
F_{17}	5.83×10^8	1.93×10^4	3.57×10^3	1.23×10^2	1.01×10^1	**7.98**
F_{18}	2.44×10^{12}	1.58×10^3	3.72×10^3	1.42×10^3	1.97	$\mathbf{9.68 \times 10^{-1}}$
F_{19}	2.33×10^3	3.81×10^1	3.51×10^1	2.80	2.63×10^1	$\mathbf{4.83 \times 10^{-1}}$
F_{20}	4.43×10^3	2.06×10^2	2.41×10^2	1.16×10^2	1.16×10^2	$\mathbf{1.07 \times 10^2}$
F_{21}	2.21×10^2	1.03×10^2	1.04×10^2	1.01×10^2	$\mathbf{1.00 \times 10^2}$	$\mathbf{1.00 \times 10^2}$
F_{22}	3.72×10^3	2.97×10^2	2.72×10^2	2.27×10^2	1.36×10^2	$\mathbf{1.08 \times 10^2}$
F_{23}	6.06×10^3	3.23×10^2	6.78×10^2	3.09×10^2	1.46×10^2	$\mathbf{1.13 \times 10^2}$
F_{24}	4.35×10^9	9.98×10^4	$\mathbf{4.00 \times 10^2}$	4.56×10^2	$\mathbf{4.00 \times 10^2}$	4.05×10^2
F_{25}	9.72×10^8	5.64×10^8	1.01×10^9	7.39×10^8	6.73×10^8	$\mathbf{2.95 \times 10^8}$
F_{26}	4.32×10^2	4.27×10^2	4.16×10^2	4.00×10^2	3.99×10^2	$\mathbf{3.89 \times 10^2}$
F_{27}	9.17×10^8	3.31×10^5	2.54×10^4	3.43×10^4	3.09×10^2	$\mathbf{1.39 \times 10^2}$
F_{28}	2.33×10^7	6.10×10^4	9.96×10^2	3.84×10^3	$\mathbf{7.40 \times 10^2}$	1.18×10^3
F_{29}	1.74×10^{10}	1.78×10^6	1.20×10^6	8.33×10^4	9.63×10^5	$\mathbf{2.70 \times 10^4}$

Table 2. Comparison results of PJADE with other algorithms in 10 dimensions

Algorithms	+	=	−
RGA	29	0	0
PSO	23	5	1
DE	15	6	8
JADE	9	17	3
DEP	12	10	7

Table 3. Experimental results in 50 dimensions

	RGA	PSO	DE	JADE	DEP	PJADE
F_1	2.73×10^{11}	1.61×10^9	8.48×10^3	1.35×10^3	4.23×10^3	$\mathbf{3.55 \times 10^2}$
F_2	4.74×10^5	2.09×10^3	**0.0**	7.30×10^1	**0.0**	1.28×10^2
F_3	1.22×10^5	3.37×10^2	8.53×10^1	5.90×10^1	9.26×10^1	$\mathbf{3.44 \times 10^1}$
F_4	2.90×10^5	2.92×10^3	7.65×10^2	1.86×10^2	$\mathbf{1.14 \times 10^2}$	2.03×10^2
F_5	1.90×10^{-1}	9.92×10^{-4}	3.89×10^{-3}	2.00×10^{-4}	7.65×10^{-5}	$\mathbf{2.78 \times 10^{-5}}$
F_6	3.86×10^5	4.06×10^3	9.25×10^2	2.41×10^2	2.96×10^2	$\mathbf{2.26 \times 10^2}$
F_7	1.35×10^4	$\mathbf{3.41 \times 10^2}$	5.44×10^2	6.28×10^2	6.37×10^2	7.46×10^2
F_8	5.02	4.73	4.70	4.70	4.69	**4.57**
F_9	4.89×10^4	4.19×10^3	5.62×10^3	4.08×10^1	7.18×10^3	$\mathbf{2.27 \times 10^1}$
F_{10}	3.93×10^{10}	3.52×10^6	1.90×10^5	9.41×10^3	9.26×10^3	$\mathbf{1.93 \times 10^2}$
F_{11}	1.22×10^{11}	1.41×10^9	1.35×10^8	3.38×10^6	$\mathbf{1.56 \times 10^5}$	3.40×10^6
F_{12}	2.81×10^{11}	1.15×10^9	5.06×10^7	6.74×10^3	1.17×10^4	$\mathbf{3.98 \times 10^3}$
F_{13}	7.15×10^7	1.91×10^5	1.88×10^5	6.38×10^5	$\mathbf{8.24 \times 10^3}$	1.77×10^5
F_{14}	1.35×10^{11}	3.13×10^6	1.31×10^4	8.98×10^3	9.53×10^3	$\mathbf{7.93 \times 10^3}$
F_{15}	2.28×10^{10}	7.85×10^5	1.15×10^4	$\mathbf{2.79 \times 10^2}$	7.21×10^2	3.16×10^2
F_{16}	7.85×10^{16}	2.30×10^4	2.63×10^3	4.15×10^2	6.64×10^2	$\mathbf{3.90 \times 10^2}$
F_{17}	3.63×10^8	6.96×10^5	6.48×10^4	1.06×10^6	$\mathbf{4.14 \times 10^4}$	7.49×10^5
F_{18}	6.07×10^{15}	5.06×10^9	3.15×10^8	3.59×10^6	1.76×10^4	$\mathbf{3.10 \times 10^3}$
F_{19}	4.88×10^4	6.52×10^2	2.63×10^2	1.45×10^2	1.83×10^2	$\mathbf{1.02 \times 10^2}$
F_{20}	2.41×10^5	3.04×10^3	8.27×10^2	3.66×10^2	$\mathbf{2.59 \times 10^2}$	2.94×10^2
F_{21}	7.03×10^3	6.93×10^2	3.04×10^2	1.15×10^2	1.66×10^2	$\mathbf{1.00 \times 10^2}$
F_{22}	2.19×10^5	1.01×10^4	7.75×10^2	7.73×10^2	$\mathbf{2.94 \times 10^2}$	2.96×10^2
F_{23}	1.37×10^5	7.07×10^3	3.45×10^2	5.71×10^2	2.99×10^2	$\mathbf{2.13 \times 10^2}$
F_{24}	2.69×10^{11}	4.91×10^7	5.42×10^2	1.08×10^6	4.96×10^2	$\mathbf{4.43 \times 10^2}$
F_{25}	4.96×10^{10}	3.27×10^{10}	2.69×10^{10}	1.78×10^{10}	3.01×10^9	$\mathbf{1.13 \times 10^8}$
F_{26}	1.72×10^3	1.12×10^3	8.72×10^2	5.70×10^2	5.73×10^2	$\mathbf{5.62 \times 10^2}$
F_{27}	5.50×10^{10}	9.08×10^7	8.37×10^7	3.00×10^5	7.74×10^2	$\mathbf{3.60 \times 10^2}$
F_{28}	2.95×10^{15}	1.05×10^8	6.87×10^6	1.85×10^7	$\mathbf{6.31 \times 10^4}$	9.74×10^4
F_{29}	4.47×10^{16}	4.19×10^{10}	2.38×10^8	6.35×10^9	$\mathbf{1.57 \times 10^6}$	2.81×10^6

Table 4. Comparison results of PJADE with other algorithms in 50 dimensions

Algorithms	+	=	−
RGA	29	0	0
PSO	26	0	3
DE	19	4	6
JADE	15	12	2
DEP	16	6	7

Averages of the best solution for each algorithm in 10 and 50 dimensions are shown in Tables 1 and 3, respectively. Tables 2 and 4 show Wilcoxon signed rank test with a significance level of 5 percent between PJADE and other algorithms in 10 and 50 dimensions, respectively. Three symbols "+, −, =" indicate that PJADE exhibits significantly better, worse than, and equal to the competitor, respectively. In Tables 1 and 3, PJADE has the best solution for 18 functions in both 10 and 50 dimensions. In Tables 2 and 4, PJADE shows the better performance than RGA for all functions in both 10 and 50 dimensions, and than PSO for 23 and 26 functions in 10 and 50 dimensions, respectively. PJADE shows the better performance than DE for 15 and 19 functions in 10 and 50 dimensions, respectively, and than JADE for 9 and 15 functions in 10 and 50 dimensions, respectively. PJADE shows the better performance than DEP for 12 and 16 functions in 10 and 50 dimensions, respectively. As a result, PJADE performs equal to or better than the existing algorithms and shows relatively superior performances for high dimensional problems.

5. **Conclusions.** In this paper, we have presented perturbation of adaptive differential evolution with optional external archive for continuous optimization problem. By introducing perturbation to adaptive differential evolution with optional external archive, our algorithm tended to be able to explore the solution efficiently and to avoid to local optima. Our algorithm showed the effectiveness in comparison with real-coded genetic algorithm, particle swarm optimization, differential evolution, adaptive differential evolution with optional external archive, and differential evolution with perturbation using CEC2017 benchmark functions. Theoretical considerations of our algorithm remain as a future study.

REFERENCES

[1] E. Aiyoshi and K. Yasuda, *Metaheuristics and Applications*, Ohmsha, Japan, 2007.

[2] M. Kubo and J. P. Pedroso, *Metaheuristics: A Programing Guide*, Kyoritsu Pub., 2009.

[3] I. Ono, H. Satoh and S. Kobayashi, A real-coded genetic algorithm for function optimization using the unimodal normal distribution crossover, *Journal of Japanese Society for Artificial Intelligence*, vol.14, pp.1146-1155, 1999.

[4] J. H. Seo, C. H. Im, C. G. Heo, J. K. Kim, H. K. Jung and C. G. Lee, Multimodal function optimization based on particle swarm optimization, *IEEE Trans. Magne.*, vol.42, no.4, 2006.

[5] 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.

[6] U. Maulik and I. Saha, Automatic fuzzy clustering using modified differential evolution for image classification, *IEEE Trans. Geos. Rem. Sens.*, vol.48, no.9, pp.3503-3510, 2010.

[7] A. Slowik, Application of an adaptive differential evolution algorithm with multiple trial vectors to artificial neural network training, *IEEE Trans. Indus. Elec.*, vol.58, no.8, pp.3160-3167, 2010.

[8] Y. J. Gong and Y. Zhou, Differential evolution superpixel segmentation, *IEEE Trans. Image Proc.*, vol.27, no.3, pp.1390-1404, 2018.

[9] J. Zhang and A. C. Sanderson, JADE: Adaptive differential evolution with optional external archive, *IEEE Trans. Evol. Comput.*, vol.13, no.5, pp.945-958, 2009.

[10] T. Muto and M. Maeda, Differential evolution with perturbation for continuous function optimization, *Nonlinear Theory and Its Applications, IEICE*, vol.13, no.2, pp.239-245, 2022.

[11] N. H. Awad, M. Z. Ali, P. N. Suganthan, J. J. Liang and B. Y. Qu, *Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization*, School EEE, Nanyang Technol. Univ., Singapore, School Compt. Inf. Syst., Technol. Ramtha, Jordan, and School Elect. Eng., Zhengzhou Univ., Zhengzhou, China, Rep., 2016.