Journal of Uncertain Systems
Vol.12, No.4, pp.256-266, 2018
Online at: www.jus.org.uk

Differential Evolution with Adaptive Parameter Strategy for Continuous Optimization Problems

Gaoji Sun*
College of Economic and Management, Zhejiang Normal University, Jinhua 321004, China

Received 2 June 2018; Revised 15 October 2018

Abstract

Differential evolution (DE) algorithm has remarkable performance in dealing with global optimization problems, but its performance is highly dependent on its mutation operator and control parameters (mutation factor and crossover rate). In this paper, we introduce a modified version of GPDE (one outstanding variant of DE with Gaussian mutation and dynamic parameter) with adaptive parameter strategy, and denote it as AGPDE. Compared to GPDE, AGPDE adopts an individual-dependent parameter based on the individual's fitness information and an individual-independent macro-control rule to set the values of mutation factor and crossover rate, and introduces a decreasing function to adjust the variance of Gaussian mutation operator. In addition, in AGPDE, a more concise rule is applied to coordinate the two adopted mutation operators. To show the effectiveness of the proposed AGPDE, it compares with seven existing DE algorithms on CEC 2014 contest test functions, and the experiments results show that AGPDE obtains the best performance among the eight DE algorithms.

Keywords: differential evolution, Gaussian mutation, adaptive parameter, individual-dependent parameter, global optimization

1 Introduction

Over the last two decades, global optimization problems have attracted a great interest of researchers and many nature-inspired intelligent algorithms have been developed, such as genetic algorithm [7], differential evolution [16], particle swarm optimization [11], ant colony optimization [5], artificial bee colony optimization [10], biogeography-based optimization [15] and joint operations algorithm [20]. These algorithms have been used to deal with many real-life optimization problems, such as operations research, computer science and engineering design [3, 9, 18, 23, 25]. Among the proposed intelligent algorithms, DE has shown significant success in solving different numerical optimization problems. However, the choice of the mutation strategies or control parameters plays a key role in the performance of DE algorithms. Many related works improve the performance of DE significantly. We only briefly review some papers in the following paragraph, the other recent research about DE can be found in the literature reviews [1, 4] and the references therein.

DE usually has three main operators: mutation, crossover and selection, and three important control parameters: population size, mutation factor and crossover rate. In fact, most of the existing variants of DE only focus on setting the mutation operator, mutation factor and crossover rate. It is generally recognized that mutation operator reflects the core essence of DE and has the greatest influence on the performance of DE, hence lots of research work concentrate on proposing novel mutation operators or combining some different kinds of mutation operators together. For example, in SADE [14], the authors adopted four different mutation operators, and the selection strategy for each individual is according to the success rate of each mutation operator; a new mutation operator called "DE/current-to-*p*best" is proposed in JADE [27], which based on a concept of "*p*best" individual (randomly chosen from the top $100 * p\%$ individuals set); in GDE [8], two mutation operators with different characteristics are respectively designated into two disparate groups, and an adaptive tuning strategy based on the well-known 1/5th rule is used to dynamically reassign the size of the two groups; in MGBDE [24], the authors proposed a new kind of Gaussian mutation strategy

*Corresponding author.
Email: gsun@zjnu.edu.cn (G. Sun).

and combined it with a classical mutation operator to handle optimization problems. Mutation factor is an important component of mutation operator and crossover rate is the key parameter of crossover operator, they both have important influence on the performance of DE. Therefore, many researchers pay attention to optimize the setting of those two control parameters. The existing strategies of the control parameters can be briefly divided into two classes according to the generated method of their values: based on a pre-given fixed rule or depended on some valuable information during the optimization process. Liu and Lampinen [13] applied a fuzzy logic controller to adapt the mutation factor and crossover rate. Brest et al. [2] proposed a parameters setting method that the values of mutation factor and crossover rate are randomly chosen within a certain range, and the parameters are updated only if satisfying the pre-given probability constraints. Draa et al. [6] used two sine functions to dynamically adjust the values of mutation factor and crossover rate, respectively. The aforementioned three DE variants belong to the first class. Some DE variants applied the information of successful experience to control mutation factor and crossover rate, such as SADE, JADE, and an improved variant of the JADE algorithm called SHADE [21]. Some other DE variants adopted the fitness information of individual to adjust mutation factor and crossover rate, such as ADE [26], IDE [22], IDDE [19] and so on. In fact, for the control parameters setting, using the valuable information becomes more and more popular.

In this paper, a modified version of GPDE [17], which is one of our previous work, is proposed. In GPDE, a new kind of Gaussian mutation operator and one modified mutation operator called "DE/rand-worst/1" are used to generate mutant vector, and a cosine function and Gaussian function are adopted to adjust the mutation factor and crossover rate, respectively. In addition, a cooperative rule based on the historical success rate and the current success rate of each mutation operator to coordinate the two proposed mutation operators. Compared to GPDE, the modified version (denoted by AGPDE) proposed in this paper mainly has three different points. First, a decreasing control parameter is embedded into the Gaussian mutation operator in GPDE, which can provide a better balance between the exploration ability and exploitation ability during the optimization routine. Second, two adaptive parameter tuning strategies based on different combinations of an individual-dependent parameter and an individual-independent macro-control parameter are used to adjust the mutation factor and crossover rate. Third, a more concise cooperative rule based on the accumulated success rate is applied to coordinate the two adopted mutation operators. To judge the effectiveness of the proposed AGPDE algorithm, one well known set of test functions (from the IEEE CEC 2014 [12] competition problem sets) and seven excellent DE variants are used to carry out the comparison experiment.

The remainder of this paper is organized as follows. Section 2 briefly introduces the basic operators of classical DE algorithm. Section 3 provides a detailed description of the proposed AGPDE algorithm. Section 4 presents the comparison between AGPDE and seven compared DE algorithms. Section 5 draws the conclusions.

2 Classical DE

The classical DE algorithms are briefly presented in this section. DE is a population-based stochastic search algorithm. It starts with an initial population, which consists of NP D-dimensional real-valued vectors $\boldsymbol{x}_i = [x_{i,1}, x_{i,2}, \cdots, x_{i,D}], i = 1, 2, \ldots, NP$, where NP is the population size and D is the problem's dimension. Usually, the initial individuals are randomly generated via the following formula,

$$x_{i,j} = x_{j,\min} + \text{rand}_{i,j}[0,1] \cdot \big(x_{j,\max} - x_{j,\min}\big), \tag{1}$$

where $\text{rand}_{i,j}[0,1]$ represents a uniformly distributed number generated between 0 and 1. There are three main operations in the classical DE: mutation, crossover, and selection, which willbe described in detail in the following subsections.

2.1 Mutation Operation

For each target vector $\boldsymbol{x}_i$, DE employs a mutation operation to create a corresponding mutant vector $\boldsymbol{v}_i = \big[v_{i,1}, v_{i,2}, \cdots, v_{i,D}\big]$ based on the current parent population. The following are five well-known and widely used mutation strategies.

1) DE/rand/1

$$\boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F \cdot (\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}). \tag{2}$$

2) DE/best/1

$$\boldsymbol{v}_i = \boldsymbol{x}_{best} + F \cdot (\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2}). \tag{3}$$

3) DE/current-to-best/1

$$\boldsymbol{v}_i = \boldsymbol{x}_i + F \cdot (\boldsymbol{x}_{best} - \boldsymbol{x}_{r_1}) + F \cdot (\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}). \tag{4}$$

4) DE/best/2

$$\boldsymbol{v}_i = \boldsymbol{x}_{best} + F \cdot (\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2}) + F \cdot (\boldsymbol{x}_{r_3} - \boldsymbol{x}_{r_4}). \tag{5}$$

5) DE/rand/2

$$\boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F \cdot (\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}) + F \cdot (\boldsymbol{x}_{r_4} - \boldsymbol{x}_{r_5}). \tag{6}$$

The indices r_1, r_2, r_3, r_4 and r_5 are mutually different random integers randomly chosen from the set $\{1, 2, \dots, NP\}$ and all are different from the base index i. F is the mutation factor, which is used to control the difference vectors. The vector $\boldsymbol{x}_{best} = (x_{best,1}, x_{best,2}, \cdots, x_{best,D})$ is the best individual in the current population.

2.2 Crossover Operation

Generally, after the mutation operation, a crossover operation is employed on the target vector $\boldsymbol{x}_i$ and its corresponding mutant vector $\boldsymbol{v}_i$ to generate a trial vector $\boldsymbol{u}_i = \big[u_{i,1}, u_{i,2}, \cdots, u_{i,D}\big]$, and the crossover operation can be described as

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } \big(j = j_{rand} \text{ or } \text{rand}_{i,j}[0,1] \le CR\big) \\ x_{i,j}, & \text{otherwise} \end{cases} \tag{7}$$

where parameter $CR \in (0,1)$ is the predefined crossover rate, and j_{rand} is a random index chosen from the set $\{1, 2, \dots, D\}$.

2.3 Selection Operation

After the crossover operation, a selection operation is used to determine whether the trail vector or the target vector can be saved into the next iteration according to their fitness values $f(\cdot)$. The selection operation for solving minimization problems is performed as follow:

$$\boldsymbol{x}_i = \begin{cases} \boldsymbol{u}_i, & \text{if } f(\boldsymbol{u}_i) \le f(\boldsymbol{x}_i) \\ \boldsymbol{x}_i, & \text{otherwise.} \end{cases} \tag{8}$$

3 Description of AGPDE

In this section, we firstly review the Gaussian mutation operator in GPDE and then describe its modified version. Secondly, we review the DE/rand-worst/1 adopted in GPDE, and introduce a new adaptive strategy for the mutation factor. Thirdly, we describe the cooperative rule between the two mutation operators. Lastly, we summarize the overall procedure of AGPDE.

3.1 Gaussian Mutation Operator

Since the 3-σ rule of Gaussian distribution (denoted by $N(\mu, \sigma^2)$) provides a wonderful chance to control the hunting zone, in GPDE [17], a novel mutation operator, which combined the crossover operator (7), is proposed to directly produce the new trial vector $\big($denoted by $\boldsymbol{u}_i^g = \big(u_{i,1}^g, u_{i,2}^g, \cdots, u_{i,D}^g\big)\big)$ for the i-th individual, $i = 1, 2, \dots, NP$,

$$u_{i,j}^g = \begin{cases} N\left(x_{r_1,j}, \big(x_{r_2,j} - x_{r_3,j}\big)^2\right), & \text{if } \big(j = j_{rand} \text{ or } \text{rand}_{i,j}[0,1] \le CR_t^i\big), \\ x_{i,j}, & \text{otherwise,} \end{cases} \tag{9}$$

where the indices r_1, r_2, r_3 are randomly generated from the set $\{1, 2, \dots, NP\}$ and $r_1 \ne r_2 \ne r_3 \ne i$. It should be point out that the novel Gaussian mutation operator always takes the r_1-th individual (the **best** one among the three randomly selected individuals) as the base vector, which means that the Gaussian mutation operator always focuses on exploit the most promising region around the r_1-th individual, and the range of the searching region is controlled by the distance $|x_{r_2,j} - x_{r_3,j}|$. In order to enhance the exploitation ability of Gaussian mutation operator (9) in GPDE, we bring a decreasing function F_t^g to gradually shrinkage the searching region. The modified Gaussian mutation operator can be described as follow:

$$u_{i,j}^g = \begin{cases} N\left(x_{r_1,j}, \left(F_t^g \times \left(x_{r_2,j} - x_{r_3,j}\right)\right)^2\right), & \text{if } \left(j = j_{rand} \text{ or } \text{rand}_{i,j}[0,1] \le CR_t^i\right), \\ x_{i,j}, & \text{otherwise,} \end{cases} \tag{10}$$

where $F_t^g = (F_t)^2$ and $F_t = (T - t + 1)/T$. The indexes t and T represent the current and the maximum allowable generation, respectively. It is easy to see that the value of F_t^g is decreased from 1 to $1/T^2$ during the optimization process. In addition, the Gaussian mutation operator will only be executed when meeting the triggering condition $\left(j = j_{rand} \text{ or } \text{rand}_{i,j}[0,1] \le CR_t^i\right)$.

For the i-th individual, the corresponding value of crossover rate (CR_t^i) in the t-th generation can be computed by

$$CR_t^i = \sqrt{0.5 \times (F_t \times F_t + (1 - F_t) \times I_t^i)}, \quad i = 1, 2, \ldots, NP, \ t = 1, 2, \ldots, T, \tag{11}$$

where $I_t^i = (F_i - F_w)/(F_w - F_b + \varepsilon)$. The symbols F_i, F_w, F_b respectively represent the fitness values of the i-th individual, the worst individual and the best individual in the current population, and ε indicates a very small number, which is used to avoid the case of $F_w - F_b = 0$ and is set to 1.0×10^{-99} in our paper. Actually, I_t^i reflects the fitness value's state of the i-th individual in the current population. Crossover rate (CR_t^i) in formula (11) implies three meanings: 1) its value is determined by an individual-dependent parameter I_t^i and an individual-independent macro-control parameter F_t; 2) its value mainly depends on the macro-control parameter F_t in the earlier stage, but mainly depends on the individual-dependent parameter I_t^i in the later stage; 3) the crossover rate of different individuals only have minor variation in the earlier stage, but the variation becomes more and more large during the running process. In fact, the above-mentioned three phenomena just are the reasons that why we take the form of formula (11) to compute the crossover rate of each individual. Generally speaking, in the earlier stage of the searching process, although the individuals have different fitness value, we don't know which one can produce the most competitive offspring, hence the crossover rate of different individuals should have no considerable difference, but in the later stage, the better individuals should have minor change and the worse ones should have major changes, which can provide a better balance between the solution's precision and the population diversity.

3.2 DE/rand-worst/1

Based on the existing knowledge about the base vector and difference vector, GPDE modified one of the most popular mutation operator (DE/rand/1) and proposed a new mutation operator (denoted by DE/rand-worst/1). In GPDE, the proposed DE/rand-worst/1 and crossover operator (7) are combined to directly produce the new trial vector (denoted by $\boldsymbol{u}_i^m = \left(u_{i,1}^m, u_{i,2}^m, \ldots, u_{i,D}^m\right), i = 1, 2, \ldots, NP\right)$ of each individual, the corresponding formula can be described as follow,

$$u_{i,j}^m = \begin{cases} x_{r_1',j} + F_t^i \cdot (x_{r_2',j} - x_{r_3',j}), & \text{if } \left(j = j_{rand} \text{ or } \text{rand}_{i,j}[0,1] \le CR_t^i\right), \\ x_{i,j}, & \text{otherwise,} \end{cases} \tag{12}$$

where the indices r_1', r_2' and r_3' are randomly chosen from the set $\{1, 2, \ldots, NP\}$ and $r_1' \ne r_2' \ne r_3' \ne i$. Moreover, the r_3'-th individual is the **worst** one among the three randomly selected individuals. In addition, note that the mutation operator DE/rand-worst/1 in formula (12) has the same triggering condition with the modified Gaussian mutation operator in formula (10).

For each individual, the value of its current mutation factor F_t^i can be calculated via the following formula,

$$F_t^i = \frac{F_t + I_t^i}{2}. \tag{13}$$

Obviously, the value of mutation factor F_t^i also depends on the individual-dependent parameter I_t^i and the individual-independent macro-control parameter F_t. From the formula (13), we can see that in the earlier stage, every individual has a relatively large mutation factor, leading to that the population has more powerful exploration ability, but all of the individuals have relatively small mutation factor in the later stage, resulting in that the population has more powerful exploitation ability at that moment. Furthermore, in the same generation, the better individuals always have smaller mutation factor than the worse ones, which implies the better individuals mainly are in charge of the exploitation work and the worse individuals mainly hold the post of exploration work.

3.3 Cooperative Rule

In AGPDE, we will adopt the modified Gaussian mutation operator and DE/rand-worst/1 to produce the new trial vector, but for one specific individual, the first confronting problem is that how to choose the two mutation operators. Therefore, it is imperative to set an appropriate cooperative rule between the two mutation operators. It's a natural selection that setting the cooperative rule based on the two mutation operators' historical success rate. More specifically, we take the modified Gaussian mutation operator (10) as an example, until to the t-th generation, the current historical success rate of Gaussian mutation operator (denoted by SR_t^g) is equal to S_t^g/R_t^g, where S_t^g and R_t^g represent the cumulative success times and cumulative run times. Meanwhile, we adopt $SR_t^m = S_t^m/R_t^m$ to compute the historical success rate of DE/rand-worst/1. At the beginning of the optimization process, we set all the initial values of S_0^g, R_0^g, S_0^m and R_0^m to 1. During the running process, for each given individual, if it adopts the Gaussian mutation operator to produce its offspring, then executing $R_t^g = R_t^g + 1$, otherwise executing $R_t^m = R_t^m + 1$; if the generated offspring via Gaussian mutation operator is better than its parent, then executing $S_t^g = S_t^g + 1$, and if the offspring is better than the best individual, executing $S_t^g = S_t^g + 1$ again. The same rule is also implemented to the DE/rand-worst/1. As a result, at the beginning of the t-th generation, the value of a parameter involved in cooperative rule can be derived in terms of the two mutation operators' historical success rate, which can be calculated by,

$$SR_t = \frac{SR_t^g}{SR_t^g + SR_t^m}, \tag{14}$$

where parameter SR_t is applied to control the selection probability of Gaussian mutation operator in the next generation. Furthermore, the detailed cooperative rule can be described as follows,

$$\boldsymbol{u}_i = \begin{cases} \boldsymbol{u}_i^g, & \text{if } \ \mathrm{rand}_i[0,1] < SR_t, \\ \boldsymbol{u}_i^m, & \text{otherwise.} \end{cases} \tag{15}$$

The cooperative rule (14) shows that the chance of executing the adopted two mutation operators relies on their own historical success rate, and the one with higher cumulative success rate has the more chance to produce the trial vectors. After the new trial vector $\boldsymbol{u}_i$ produced, comparing the two fitness values of $\boldsymbol{u}_i$ and its target vector $\boldsymbol{x}_i$, then determining the offspring via the selection operator (8).

3.4 The Overall Procedure of AGPDE

We have provided a detailed description of Gaussian mutation operator, DE/rand-worst/1, and the cooperative rule between them. Now we summarize the overall procedure of AGPDE into Algorithm 1.

4 Experimental Setup and Results

4.1 Benchmark Functions

In order to verify the performance of AGPDE algorithm, a test bed of well-known 30 benchmark functions are used in the following experiments. These functions are presented for IEEE CEC 2014 competition. Among these benchmark functions, $f_1 - f_3$ are unimodal functions with only one minima; $f_4 - f_{16}$ are simple multimodal functions with many local minima; $f_{17} - f_{22}$ are hybrid functions, which have many different sub-functions with different properties; $f_{23} - f_{30}$ are composition functions, which are constructed by many different basic functions. The detail description of these functions can be found in [12].

Algorithm 1 The overall procedure of AGPDE

```
1: Set the values of parameters $NP$ and $T$;
2: Initialize $NP$ individuals with random positions via the formula (1);
3: for ($t = 1; t <= T; t++$), do
4:    Compute the value of parameter $SR_t$ in the $t$-th generation via the formula (14);
5:    Find out the best individual and the worst individual in the current population;
6:    for ($i = 1; i <= NP; i++$), do
7:       For the $i$-th individual, compute its current crossover rate $CR_t^i$ via the formula (11);
8:       if rand[0, 1] < $SR_t$ then
9:          Generate the new trial vector via the formula (10) included modified Gaussian mutation operator;
10:      else
11:         Generate the new trial vector via the formula (12) contained DE/rand-worst/1;
12:      end if;
13:      Update the $i$-th individual via the selection operator (8);
14:      Replace the best individual $\boldsymbol{x}_{best}$ by the new individual $\boldsymbol{x}_i$ if $\boldsymbol{x}_i$ is better than $\boldsymbol{x}_{best}$;
15:      Update the success rates $SR_t^g$ and $SR_t^m$ of the two adopted mutation operators;
16:   end for
17: end for
18: Output the position of the best individual as the global optimal solution.
```

4.2 Comparison to State-of-the-art DE Variants

In this section, we compare AGPDE with GPDE [17] and six other existing DE variants: SADE [14], JADE [27], GDE [8], MGBDE [24], SinDE [6] and SHADE [21]. It should be pointed out that for all the aforementioned compared algorithms, except the population sizes $NP = D = 30$, the other involved control parameters keep the same with their corresponding literature. In our experiments, we respectively report the average error (denoted by "Mean") function values $\left(f\left(\boldsymbol{x}_{best}\right) - f\left(\boldsymbol{x}^*\right)\right)$, the corresponding standard deviation (Std.) and the statistical conclusion of comparative results based on 50 independent runs, where $\boldsymbol{x}_{best}$ is the best solution found by the algorithm in one run and $\boldsymbol{x}^*$ is the global optimum of test function. In addition, for all the compared algorithms, the maximum allowable function evaluation are set to $10000 \times D$ for all the test functions, which means the maximum allowable generation are set to 10000, and Wilcoxon's rank sum test at a 0.05 significance level is conducted on the experimental results to have statistically sound conclusions. The symbols "=", "−" and "+" in the following tables separately means the result of AGPDE equals to, worse than and better than the corresponding comparison algorithms. The achieved results is reported in Tables 1–2, and the best results are shown in **boldface**. The comparison results among AGPDE and the other seven algorithms are summarized in the last row of the Table 2. Furthermore, we apply Figures 1–2 to shows the performance graphs of the compared algorithms.

Based on the results reported in Tables 1-2, we can see that our proposed AGPDE achieves better results than other algorithms on the majority of test functions. In details, compared to SADE, JADE, GDE, MGBDE, SinDE, SHADE and GPDE, AGPDE outperforms them on 23, 21, 21, 24, 12, 17 and 12 test functions, similar as them on 1, 2, 5, 1, 11, 5 and 11 test functions, and slightly worse than them on 6, 7, 4, 5, 7, 8 and 7 test functions. In addition, AGPDE obtains 13 best results and takes the first place, SinDE and SHADE are tied for second with scores of 9 best results. It should be mentioned that AGPDE has no losing on 16 test functions ($f_6, f_9, f_{11} - f_{16}, f_{19}, f_{22} - f_{26}$ and f_{28}), but it is intersting that AGPDE has no winning on 3 test functions (f_1, f_4 and f_{29}).

In summary, compared with GPDE, AGPDE has a better performance on multimodal functions and composition functions, but has an unsatisfactory performance on unimodal functions and hybrid functions (which usually take one unimodal functions as a subcomponent). Figures 1–2 show that AGPDE exhibits a medium convergence speed among the compared algorithms and converges slower than GPDE on most functions, but AGPDE exhibits a continuous improvement characteristic during the whole optimization routine, maybe that is why AGPDE has different performance on different categories of test functions.

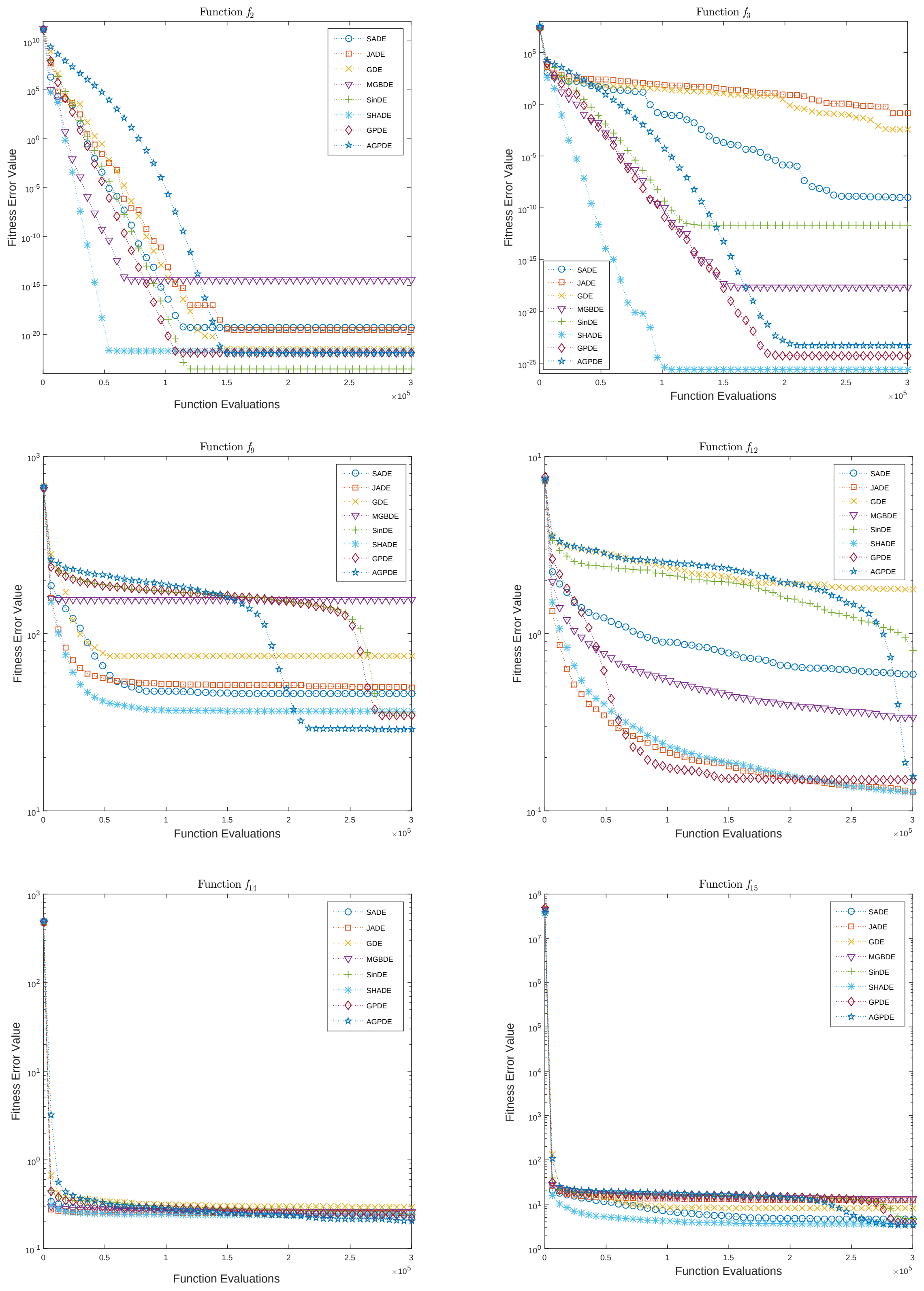

Figure 1: Convergence graphs (mean curves) for the involved eight algorithms on functions $f_2, f_3, f_9, f_{12}, f_{14}$ and f_{15} over 50 independent runs

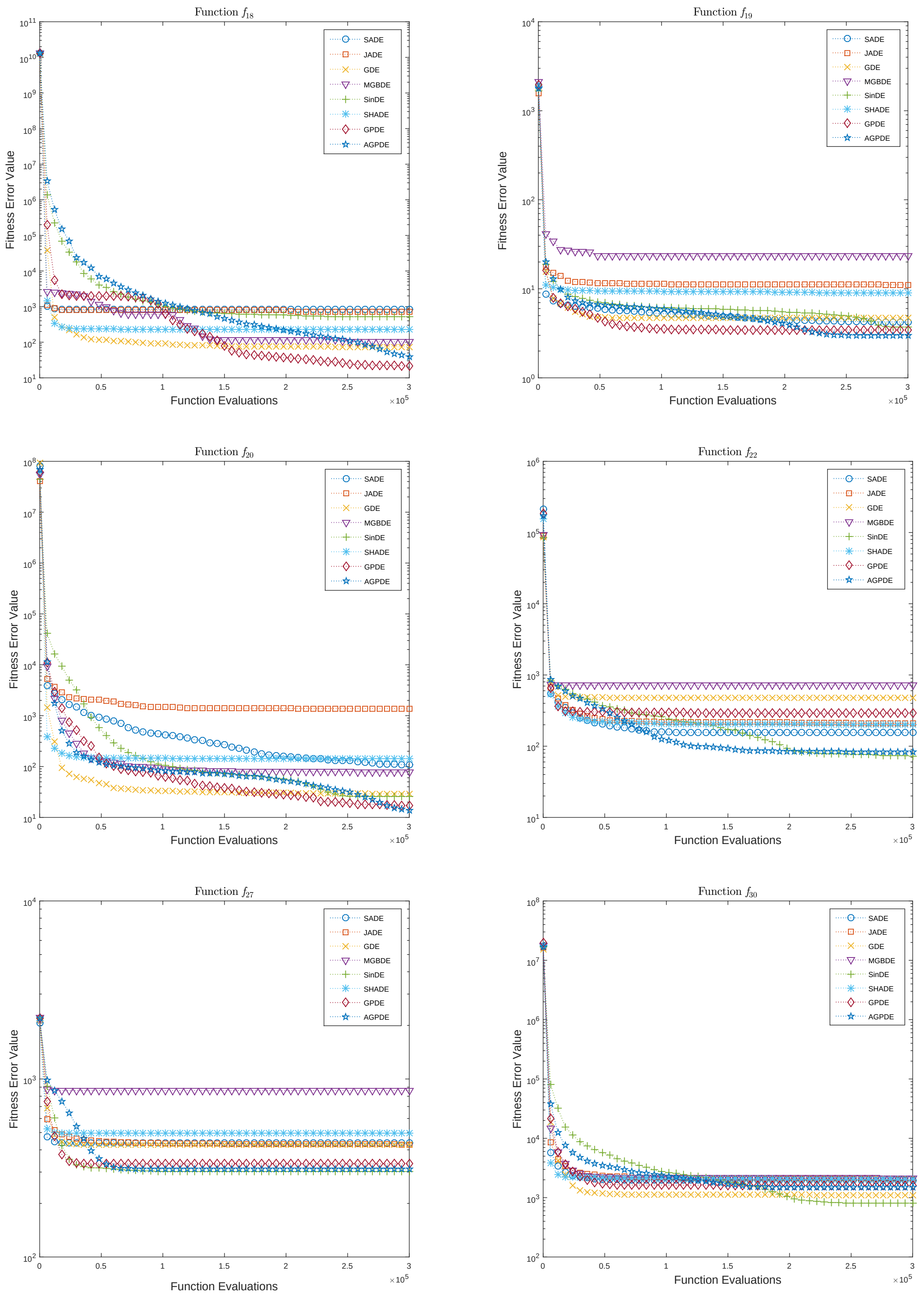

Figure 2: Convergence graphs (mean curves) for the involved eight algorithms on functions $f_{18}, f_{19}, f_{20}, f_{22}, f_{27}$ and f_{30} over 50 independent runs

Table 1: Comparative results on functions $f_1 - f_{15}$

Func.	Metric	SADE	JADE	GDE	MGBDE	SinDE	SHADE	GPDE	AGPDE
	Mean	3.22e+04	3.76e+04	6.85e+03	5.57e+03	1.92e+06	**5.83e+02**	5.21e+04	2.17e+06
f_1	Std.	2.17e+04	3.66e+04	6.41e+03	3.45e+03	1.11e+06	**2.24e+03**	3.51e+04	1.15e+06
	+/=/−	−	−	−	−	=	−	−	−−
	Mean	4.84e-20	3.08e-20	3.06e-22	3.92e-15	**2.90e-24**	2.06e-22	1.35e-23	1.21e-22
f_2	Std.	2.39e-19	9.38e-20	3.80e-22	1.95e-14	**1.01e-23**	1.69e-22	2.17e-22	1.69e-22
	+/=/−	+	+	+	+	−	+	=	−−
	Mean	9.82e-10	1.34e-01	3.56e-03	2.08e-18	2.18e-12	**2.29e-26**	5.42e-25	5.07e-24
f_3	Std.	4.91e-09	6.05e-01	9.98e-03	9.54e-18	9.18e-12	**1.03e-25**	1.83e-24	2.00e-23
	+/=/−	+	+	+	+	+	−	−	−−
	Mean	1.39e+01	1.94e+01	5.48e+00	**9.95e-05**	1.43e+01	2.76e+00	2.99e+00	1.63e+01
f_4	Std.	2.72e+01	3.09e+01	1.89e+01	**4.48e-04**	2.28e+01	1.30e+01	1.49e+01	2.66e+01
	+/=/−	−	−	−	−	=	−	−	−−
	Mean	2.04e+01	**2.00e+01**	2.09e+01	2.02e+01	2.06e+01	**2.00e+01**	**2.00e+01**	2.01e+01
f_5	Std.	4.74e-02	**3.99e-03**	1.48e-01	3.92e-02	4.77e-02	**9.03e-03**	**6.53e-06**	1.80e-01
	+/=/−	+	−	+	+	+	=	−	−−
	Mean	8.93e+00	1.14e+01	8.25e+00	2.22e+01	**1.93e-02**	6.77e+00	1.33e+00	2.26e-01
f_6	Std.	2.03e+00	1.75e+00	2.85e+00	3.64e+00	**9.27e-02**	1.90e+00	1.16e+00	4.91e-01
	+/=/−	+	+	+	+	=	+	+	−−
	Mean	1.89e-02	2.69e-02	1.30e-02	1.12e-02	**0.00e+00**	1.13e-02	2.17e-03	2.17e-03
f_7	Std.	2.10e-02	2.17e-02	1.75e-02	1.44e-02	**0.00e+00**	1.67e-02	4.17e-03	3.58e-03
	+/=/−	+	+	+	+	−	+	=	−−
	Mean	4.78e+00	3.98e-02	6.38e+01	1.29e+02	4.70e-01	**2.84e-16**	9.79e+00	6.96e+00
f_8	Std.	2.54e+00	1.99e-01	1.58e+01	3.17e+01	5.69e-01	**8.39e-16**	3.72e+00	2.22e+00
	+/=/−	−	−	+	+	−	−	+	−−
	Mean	4.59e+01	4.97e+01	7.46e+01	1.56e+02	3.58e+01	3.66e+01	3.46e+01	**2.88e+01**
f_9	Std.	1.09e+01	8.71e+00	2.74e+01	2.95e+01	7.51e+00	7.79e+00	9.26e+00	**6.92e+00**
	+/=/−	+	+	+	+	+	+	+	−−
	Mean	3.40e+00	6.98e+00	1.80e+03	1.25e+03	9.31e+00	**1.27e-01**	1.25e+02	3.15e+01
f_{10}	Std.	2.09e+00	2.40e+01	6.43e+02	7.96e+02	4.77e+00	**2.30e-01**	9.65e+01	4.62e+01
	+/=/−	−	−	+	+	−	−	+	−−
	Mean	2.42e+03	2.04e+03	5.06e+03	2.85e+03	2.35e+03	1.78e+03	1.97e+03	**1.57e+03**
f_{11}	Std.	5.45e+02	2.43e+02	1.60e+03	6.29e+02	4.08e+02	3.47e+02	4.71e+02	**4.16e+02**
	+/=/−	+	+	+	+	+	=	+	−−
	Mean	5.91e-01	1.28e-01	1.77e+00	3.36e-01	8.04e-01	**1.27e-01**	1.49e-01	1.56e-01
f_{12}	Std.	8.56e-02	2.61e-02	8.70e-01	3.67e-02	1.24e-01	**2.65e-02**	7.84e-02	7.24e-02
	+/=/−	+	=	+	+	+	=	=	−−
	Mean	2.80e-01	3.09e-01	3.64e-01	4.40e-01	2.06e-01	2.78e-01	2.40e-01	**1.92e-01**
f_{13}	Std.	5.43e-02	5.90e-02	7.23e-02	7.72e-02	5.27e-02	6.42e-02	6.84e-02	**4.18e-02**
	+/=/−	+	+	+	+	=	+	+	−−
	Mean	2.41e-01	2.50e-01	2.95e-01	2.58e-01	2.42e-01	2.38e-01	2.22e-01	**2.04e-01**
f_{14}	Std.	4.47e-02	1.01e-01	9.12e-02	5.78e-02	2.60e-02	4.78e-02	3.40e-02	**3.75e-02**
	+/=/−	+	+	+	+	+	+	+	−−
	Mean	4.57e+00	1.23e+01	8.12e+00	1.33e+01	4.81e+00	3.55e+00	3.75e+00	**3.37e+00**
f_{15}	Std.	1.31e+00	6.69e+00	2.95e+00	2.37e+00	9.80e-01	1.35e+00	9.44e-01	**8.34e-01**
	+/=/−	+	+	+	+	+	=	=	−−

5 Conclusions

In this paper, to improve the performance of GPDE, we propose an adaptive parameter adjustment strategy based on an individual-dependent parameter and an individual-independent macro-control rule to set the values of mutation factor and crossover rate at the individual-level. In fact, two user-specified constants, which is used to produce the concrete values of mutation factor and crossover rate during the optimization routine in GPDE, are eliminated by using the new adaptive parameter adjustment strategy. Furthermore, we introduce a decreasing function to the Gaussian mutation operator to enhance the accuracy of the optimal solution, meanwhile adopt a more simple and efficient rule based on each mutation operator's successful experiences to coordinate the selection decision of the two adopted mutation operator for every individual. Numerical experiments on 30 benchmark functions in the CEC 2014 special session have been conducted. The performance of AGPDE is also compared with seven popular state-of-the-art DE variants, and it is concluded that AGPDE outperforms all of the other DE variants.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant No.71701187, and Research Project of Zhejiang Education Department under Grant No. Y201738184.

Table 2: Comparative results on functions $f_{16} - f_{30}$

Func.	Metric	SADE	JADE	GDE	MGBDE	SinDE	SHADE	GPDE	AGPDE
	Mean	1.03e+01	1.02e+01	1.10e+01	1.07e+01	1.00e+01	9.51e+00	9.64e+00	**8.05e+00**
f_{16}	Std.	4.16e-01	3.56e-01	1.12e+00	6.49e-01	5.22e-01	5.70e-01	7.85e-01	**7.06e-01**
	+/=/−	+	+	+	+	+	+	+	−−
	Mean	5.95e+03	6.29e+04	1.91e+04	1.71e+03	1.25e+05	**1.45e+03**	3.84e+03	1.28e+05
f_{17}	Std.	4.17e+03	6.21e+04	3.03e+04	7.57e+02	1.20e+05	**5.04e+02**	3.53e+03	1.45e+05
	+/=/−	−	−	−	−	=	−	−	−−
	Mean	8.34e+02	7.06e+02	7.43e+01	1.03e+02	5.15e+02	2.30e+02	**2.16e+01**	3.96e+01
f_{18}	Std.	1.22e+03	9.62e+02	1.90e+02	3.60e+01	6.94e+02	5.21e+02	**9.20e+00**	2.28e+01
	+/=/−	+	+	=	+	+	+	−	−−
	Mean	4.23e+00	1.11e+01	4.73e+00	2.33e+01	3.71e+00	8.96e+00	3.45e+00	**2.99e+00**
f_{19}	Std.	1.23e+00	1.66e+01	1.20e+00	2.55e+01	7.18e-01	1.18e+01	1.18e+00	**7.95e-01**
	+/=/−	+	+	+	+	+	+	=	−−
	Mean	1.08e+02	1.37e+03	2.88e+01	7.80e+01	2.57e+01	1.43e+02	1.71e+01	**1.37e+01**
f_{20}	Std.	1.42e+02	2.06e+03	2.19e+01	4.38e+01	2.85e+01	6.21e+01	1.12e+01	**3.45e+00**
	+/=/−	+	+	+	+	+	+	=	−−
	Mean	4.84e+03	5.74e+03	3.29e+03	8.50e+02	9.23e+03	**8.36e+02**	3.63e+03	2.04e+03
f_{21}	Std.	4.42e+03	7.17e+03	5.36e+03	3.90e+02	7.52e+03	**2.60e+02**	4.61e+03	1.91e+03
	+/=/−	+	+	=	−	+	−	=	−−
	Mean	1.55e+02	2.10e+02	4.82e+02	7.17e+02	**7.25e+01**	2.03e+02	2.90e+02	8.26e+01
f_{22}	Std.	8.61e+01	7.94e+01	2.09e+02	2.74e+02	**6.28e+01**	1.09e+02	1.41e+02	7.18e+01
	+/=/−	+	+	+	+	=	+	+	−−
	Mean	**3.15e+02**	**3.15e+02**	**3.15e+02**	**3.15e+02**	**3.15e+02**	**3.15e+02**	**3.15e+02**	**3.15e+02**
f_{23}	Std.	**1.38e-13**	**3.48e-13**	**6.13e-13**	**1.21e-12**	**1.24e-13**	**1.77e-13**	**1.04e-13**	**2.32e-13**
	+/=/−	=	=	=	=	=	=	=	−−
	Mean	2.28e+02	2.30e+02	2.35e+02	2.42e+02	2.23e+02	2.38e+02	2.27e+02	**2.10e+02**
f_{24}	Std.	5.41e+00	3.87e+00	7.27e+00	1.18e+01	8.57e-01	6.05e+00	4.51e+00	**1.10e+01**
	+/=/−	+	+	+	+	+	+	+	−−
	Mean	2.10e+02	2.12e+02	**2.04e+02**	2.20e+02	**2.04e+02**	2.08e+02	**2.04e+02**	**2.04e+02**
f_{25}	Std.	2.07e+00	1.47e+00	**1.19e+00**	6.22e+00	**6.64e-01**	3.93e+00	**7.80e-01**	**7.96e-01**
	+/=/−	+	+	=	+	=	+	=	−−
	Mean	1.12e+02	1.56e+02	**1.00e+02**	1.56e+02	**1.00e+02**	1.20e+02	1.08e+02	**1.00e+02**
f_{26}	Std.	3.31e+01	5.05e+01	**9.32e-02**	5.04e+01	**3.99e-02**	4.07e+01	2.76e+01	**3.60e-02**
	+/=/−	+	+	+	+	=	+	+	−−
	Mean	4.36e+02	4.29e+02	4.28e+02	8.61e+02	**3.02e+02**	4.96e+02	3.34e+02	3.13e+02
f_{27}	Std.	5.74e+01	6.28e+01	5.93e+01	3.11e+02	**7.08e+00**	8.00e+01	3.49e+01	2.35e+01
	+/=/−	+	+	+	+	−	+	+	−−
	Mean	9.12e+02	9.19e+02	9.56e+02	2.60e+03	7.94e+02	9.52e+02	7.93e+02	**7.88e+02**
f_{28}	Std.	4.37e+01	6.51e+01	6.23e+01	7.61e+02	3.23e+01	1.23e+02	2.60e+01	**3.82e+01**
	+/=/−	+	+	+	+	=	+	=	−−
	Mean	6.83e+02	7.84e+02	**4.80e+02**	7.18e+02	1.32e+03	7.43e+02	6.32e+02	1.54e+03
f_{29}	Std.	2.64e+02	2.69e+02	**2.73e+02**	1.26e+02	2.39e+02	9.24e+01	1.96e+02	3.18e+02
	+/=/−	−	−	−	−	−	−	−	−−
	Mean	1.96e+03	2.05e+03	1.11e+03	2.14e+03	**8.10e+02**	2.05e+03	1.62e+03	1.48e+03
f_{30}	Std.	6.22e+02	5.50e+02	3.00e+02	7.18e+02	**1.69e+02**	8.74e+02	7.04e+02	7.40e+02
	+/=/−	+	+	=	+	−	+	=	−−
Sum	+/=/−	23/1/6	21/2/7	21/5/4	24/1/5	12/11/7	17/5/8	12/11/7	−−

References

[1] Al-Dabbagh, R., Neri, F., Idris, N., and M. Baba, Algorithmic design issues in adaptive differential evolution schemes: review and taxonomy, *Swarm and Evolutionary Computation*, 2018, https://doi.org/10.1016/j.swevo.2018.03.008.

[2] Brest, J., Greiner, S., Boskovic, B., Mernik, M., and V. Zumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, *IEEE Transactions on Evolutionary Computation*, vol.10, pp.646–657, 2006.

[3] Chen, Y., and Y. Liu, A new equilibrium optimization method for hybrid uncertain decision-making system, *Journal of Uncertain Systems*, vol.12, pp.224–240, 2018.

[4] Das, S., Mullick, S.S., and P. Suganthan, Recent advances in differential evolution–an updated survey, *Swarm and Evolutionary Computation*, vol.27, pp.1–30, 2016.

[5] Dorigo, M., and C. Blum, Ant colony optimization theory: a survey, *Theoretical Computer Science*, vol.344, pp.243–278, 2005.

[6] Draa, A., Bouzoubia, S., and I. Boukhalfa, A sinusoidal differential evolution algorithm for numerical optimisation, *Applied Soft Computing*, vol.27, pp.99–126, 2015.

[7] Goldberg, D., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, New York, 1989.

[8] Han, M., Liao, S., Chang, J., and C. Lin, Dynamic group-based differential evolution using a self-adaptive strategy for global optimization problems, *Applied Intelligence*, vol.39, pp.41–56, 2013.

[9] Hu, H., Guo, S., Ma, H., Li, J., and X. Li, A credibilistic mixed integer programming model for time-dependent hazardous materials vehicle routing problem, *Journal of Uncertain Systems*, vol.11, pp.163–175, 2017.

[10] Karaboga, D., and B. Akay, A survey: algorithms simulating bee swarm intelligence, *Artificial Intelligence Review*, vol.31, pp.61–85, 2009.

[11] Kennedy, J., Eberhart, R., and Y. Shi, *Swarm Intelligence*, Morgan Kaufman, San Francisco, 2001.

[12] Liang, J., Qu, B., and P. Suganthan, Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization, Zhengzhou University, China, and Nanyang Technological University, Singapore, 2013.

[13] Liu, J., and J. Lampinen, A fuzzy adaptive differential evolution algorithm, *Soft Computing*, vol.9, pp.448–462, 2005.

[14] Qin, A., Huang, V., and P. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Transactions on Evolutionary Computation*, vol.13, pp.398–417, 2009.

[15] Simon, D., Biogeography-based optimization, *IEEE Transactions on Evolutionary Computation*, vol.12, pp.702–713, 2008.

[16] Storn, R., 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.

[17] Sun, G., Lan, Y., and R. Zhao, Differential evolution with Gaussian mutation and dynamic parameter adjustment, *Soft Computing*, 2017, https://doi.org/10.1007/s00500-017-2885-z.

[18] Sun, G., Liu, Y., and Y. Lan, Optimizing material procurement planning problem by two-stage fuzzy programming, *Computers & Industrial Engineering*, vol.58, pp.97–107, 2010.

[19] Sun, G., Peng, J., and R. Zhao, Differential evolution with individual-dependent and dynamic parameter adjustment, *Soft Computing*, vol.22, pp.5747–5773, 2018.

[20] Sun, G., Zhao, R., and Y. Lan, Joint operations algorithm for large-scale global optimization, *Applied Soft Computing*, vol.38, pp.1025–1039, 2016.

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

[22] Tang, L., Dong, Y., and J. Liu, Differential evolution with an individual-dependent mechanism, *IEEE Transactions on Evolutionary Computation*, vol.19, pp.560–574, 2015.

[23] Tayarani-N., M., Yao, X., and H. Xu, Meta-heuristic algorithms in car engine design: a literature survey, *IEEE Transactions on Evolutionary Computation*, vol.19, pp.609–629, 2015.

[24] Wang, H., Rahnamayan, S., Sun, H., and M. Omran, Gaussian bare-bones differential evolution, *IEEE Transactions on Cybernetics*, vol.43, pp.634–647, 2013.

[25] Xu, X., Hua, C., and Y. Tang, Modeling of the hot metal silicon content in blast furnace using support vector machine optimized by an improved particle swarm optimizer, *Neural Computing & Applications*, vol.27, pp.1451–1461, 2016.

[26] Yu, W., Shen, M., Chen, W., Zhan, Z., Gong, Y., Lin, Y., Liu, O., and J. Zhang, Differential evolution with two-level parameter adaptation, *IEEE Transactions on Cybernetics*, vol.44, pp.1080–1099, 2014.

[27] Zhang, J., and A. Sanderson, JADE: adaptive differential evolution with optional external archive, *IEEE Transactions on Evolutionary Computation*, vol.13, pp.945–958, 2009.