Adaptive Population Differential Evolution with Dual Control Strategy for Large-Scale Global Optimization Problems

Xin Zhang, Zhi-Hui Zhan (Corresponding Author), *Senior Member, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*—In the greedy selection operator of differential evolution (DE), the trial solution will be selected into the new population only if it is better than the original target solution. Otherwise, the generated solution is simply eliminated. However, in most cases, these eliminated solutions may still be promising, and it will waste the computing resources to directly ignore them. Especially for the large-scale global optimization (LSGO) problems, it is important to make full use of all generated solutions and to enhance the population diversity in the limited fitness evaluation budget. To address this issue, an adaptive population DE, termed as APDE, is proposed with dual control strategy. Firstly, a population increasing (pop_inc) strategy is proposed for giving the opportunity to the generated trial solutions to survive in the population even though they are not good enough. Secondly, to avoid the gradual expansion of the population due to the pop_inc strategy, a population decreasing (pop_dec) strategy is proposed based on the "degradation value" designed for solutions. In the end of every iteration, if the degradation value of a solution is large, it represents the solution has a worse fitness value or has no improvement for a long time, and this solution will be deleted. In this way, the population size can be kept within a certain range. The test functions in CEC'2013 on LSGO are used to verify the performance of APDE. The experiment shows that APDE generally outperforms the original DE and two state-of-the-art LSGO algorithms.**

Keywords—Differential evolution, selection operator, population control, large-scale global optimization

I. INTRODUCTION

Differential evolution (DE) is an efficient heuristic algorithm for the global optimization problems. DE was proposed by Storn and Price in 1997 [1], and since then, there have been a series of researches to improve this algorithm [2]-[4].

X. Zhang and Z.-H. Zhan are with the School of Computer Science and Engineering, South China University of Technology, 510006 Guangzhou, China and also with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information and the State Key Laboratory of Subtropical Building Science, South China University of Technology, 510006 Guangzhou, China. Zhi-Hui Zhan is the corresponding author, e-mail: zhanapollo@163.com.

J. Zhang is with the Hanyang University, Ansan 15588, South Korea.

This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2102102, in part by the Outstanding Youth Science Foundation under Grant 61822602, in part by the National Natural Science Foundations of China (NSFC) under Grant 61772207 and Grant 61873097, in part by Key-Area Research and Development of Guangdong Province under Grant 2020B010166002, in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003, and in part by the Ministry of Science and ICT through the National Research Foundation of Korea (NRF-2019H1D3A2A01101977).

The optimization process of DE is driven by the evolutionary operators including mutation, crossover, and selection, and is controlled by parameters like the scale factor *F* and the crossover rate *CR*. As the mutation operator and the related parameters are significant to the performance of the algorithm, the adaptive strategies of mutation and parameter settings have been most frequently researched to improve the performance of DE. For example, some adaptive DE algorithms put these mutation strategies with different characteristics into a pool, and choose one from the pool with the preference information, such as SaDE [5] and SaMDE [6]. As for the parameter settings, the two important parameters *F* and *CR* can be updated adaptively by the historical information of the population, such as SHADE [7]. Liu *et al.* [8] combined the historical and heuristic information to adaptively control the parameters. Another parameter, the population size *NP*, is usually updated according to the evolutionary state, such as DE-APTS [9]. In a recent adaptive distributed DE (ADDE) proposed by Zhan *et al.* [10], all the parameters *F*, *CR*, and *NP* are adaptively controlled. Besides, new mutation operators and parameter settings are proposed to improve the performance of the DE algorithm. For example, Li *et al.* [11] proposed a DE with the evolution path (DEEP) which used the centralized model of evolution strategy to design the new mutation operator and to generate new solutions. Yu *et al.* [12] proposed an adaptive DE (ADE) and updated the parameters in two levels including the population-level parameters and the individual-level parameters.

Although many works have been conducted on the mutation operator, there are still less concerns on the selection operator. Moreover, the researches in adaptive control of *NP* are still much fewer than those in the parameters *F* and *CR*. In the classical DE, after the mutation and crossover operators, a trial vector $\boldsymbol{u}$ is generated. The trial solution will be selected into the new population only if it has a better fitness value than its original target solution $\boldsymbol{x}$. Otherwise, the generated trial solution $\boldsymbol{u}$ will be simply eliminated and directly ignored. However, such a greedy selection scheme actually does not fully utilize the effort of the mutation and crossover operators in obtaining the trial vector $\boldsymbol{u}$ because it is eliminated and contributes nothing to the evolution. In fact, these generated solutions may be promising to have some effects in the later evolution even though they are not better than their parents in the current time. Especially in the large-scale global optimization (LSGO) problems, the satisfactory solutions need to be obtained in the limited fitness evaluation budget, and therefore it is important to make full use of the evolutionary information of solutions and to improve the population diversity [13][14].

Therefore, this paper focuses on the selection operator of DE to make the best use of the solutions that have been generated by mutation and crossover. Firstly, even though the generated solution ***u*** is not better than ***x***, it will be added into the population as a new solution if the best solution of the population has not updated in continuously several iterations. This operation is termed as population increasing (pop_inc) strategy. The pop_inc strategy helps the new generated solutions to obtain more opportunity to survive in the population, which can also increase the population diversity. Due to the pop_inc strategy, the population size *NP* will gradually become large, which may on the other hand slow down the evolution. Therefore, a population decreasing (pop_dec) strategy is also proposed as the complementation of the pop_inc strategy. The pop_dec strategy is to delete some hopeless solutions in the end of every iteration during the evolution. In this strategy, a new factor is designed for each solution, named as "degradation value" (termed as the *dg* value). If a solution has a worse fitness value in the population or not changed in several iterations, its *dg* value will be high and this solution will be regarded as hopeless and will be deleted from the current population. Therefore, both the pop_inc strategy and the pop_dec strategy work cooperatively to form a dual control strategy (DCS). Due to the DCS, the value of *NP* can be controlled in a certain range. Moreover, the population size can be adaptively controlled, and an adaptive population DE (APDE) is proposed.

The contributions of this paper lie in that we focus on a less concerned evolutionary operator (the selection operator) and a less concerned parameter (the population size *NP*) to enhance the performance of DE via a novel DCS way. This makes our research different from most of the existing researches. Specially, DCS is proposed with both the pop_inc and pop_dec strategies, which can make full and best use of all the generated solutions to increase the population diversity, being much helpful for solving the LSGO problems. The experiments are conducted on the LSGO test suites of CEC'2013 and the results show that our proposed APDE algorithm is generally superior to the classical DE and two state-of-the-art LSGO algorithms.

The rest of this paper is organized as follows. Section II describes the classical DE algorithm. Section III shows the details of the proposed APDE algorithm. Then, Section IV presents the exhaustive experiments on the benchmark functions from CEC'2013 on the LSGO problems. Finally, Section V gives a conclusion.

II. The DE Algorithm

DE is a population-based algorithm, and the first step is the initialization of the population ***pop*** with *NP* solutions ***x*** as:

$$\boldsymbol{pop}^t = \{\boldsymbol{x}_1^t, \boldsymbol{x}_2^t, \ldots, \boldsymbol{x}_{NP}^t\} \tag{1}$$

$$\boldsymbol{x}_i^t = \{x_{i,1}^t, x_{i,2}^t, \ldots, x_{i,dim}^t\}, i = 1, 2, \ldots, NP \tag{2}$$

where *t* is the iteration index, and *dim* is the dimension of the problems. The solution ***x*** in the current iteration is also named as the target vector.

After the initialization, the iterative evolution process is executed. This process mainly includes three steps, such as mutation, crossover, and selection [1].

A. Mutation

The solution generated by mutation is called as the donor vector ***v***. There are five common mutation strategies as (3) to (7) with different difference vectors [4].

$$\text{DE/rand/1: } \boldsymbol{v}_i^t = \boldsymbol{x}_{r1}^t + F\cdot(\boldsymbol{x}_{r2}^t - \boldsymbol{x}_{r3}^t) \tag{3}$$

$$\text{DE/best/1: } \boldsymbol{v}_i^t = \boldsymbol{x}_{\text{best}}^t + F\cdot(\boldsymbol{x}_{r1}^t - \boldsymbol{x}_{r2}^t) \tag{4}$$

$$\text{DE/current-to-best/1: } \boldsymbol{v}_i^t = \boldsymbol{x}_i^t + F\cdot(\boldsymbol{x}_{\text{best}}^t - \boldsymbol{x}_i^t) + F\cdot(\boldsymbol{x}_{r1}^t - \boldsymbol{x}_{r2}^t) \tag{5}$$

$$\text{DE/best/2: } \boldsymbol{v}_i^t = \boldsymbol{x}_{\text{best}}^t + F\cdot(\boldsymbol{x}_{r1}^t - \boldsymbol{x}_{r2}^t) + F\cdot(\boldsymbol{x}_{r3}^t - \boldsymbol{x}_{r4}^t) \tag{6}$$

$$\text{DE/rand/2: } \boldsymbol{v}_i^t = \boldsymbol{x}_{r1}^t + F\cdot(\boldsymbol{x}_{r2}^t - \boldsymbol{x}_{r3}^t) + F\cdot(\boldsymbol{x}_{r4}^t - \boldsymbol{x}_{r5}^t) \tag{7}$$

where *r*1, *r*2, *r*3, *r*4, and *r*5 are chosen from [1, *NP*] randomly, and $\boldsymbol{x}_{\textbf{best}}{}^t$ is the best solution in the t^{th} generation. The convention of mutation can be denoted as DE/a/b, where 'a' represents which vector to be perturbed, and 'b' represents the number of difference vectors to perturb the vector 'a'.

B. Crossover

The vector generated by crossover is called as the trial vector ***u***, and there are two common crossover operators to generate ***u***, including the binomial crossover and the exponential crossover [4].

In the binomial crossover, the decision variables of the donor vector ***v*** and the target vector ***x*** are randomly chosen with the probability *CR*. For its simplicity, the binomial crossover is used in this paper.

$$u_{i,j}^t = \begin{cases} v_{i,j}^t, & \text{if } j = jrand \text{ or } rd \le CR \\ x_{i,j}^t, & \text{otherwise} \end{cases} \tag{8}$$

where *jrand* is randomly chosen from [1, *dim*] to ensure that there is at least a variable from ***v***, and *rd* is a random number from [0,1].

In the exponential crossover, a consecutive variables of the donor vector ***v*** are randomly chosen.

$$u_{i,j}^t = \begin{cases} v_{i,j}^t, & \text{if } j = \langle n\rangle_{dim}, \langle n+1\rangle_{dim}, \ldots, \langle n+L-1\rangle_{dim} \text{ and } rd \le CR \\ x_{i,j}^t, & \text{otherwise} \end{cases} \tag{9}$$

where *n* and *L* are randomly chosen from [1, *dim*], and $\langle a\rangle_b$ is a modular operation and equal to *a* modulo *b*.

C. Selection

The selection operator determines whether the trial vector ***u*** or the target vector ***x*** can be added into the population $\boldsymbol{pop}^{t+1}$ of the next iteration.

$$x_i^{t+1} = \begin{cases} u_i^t, & \text{if } f(u_i^t) \le f(x_i^t) \\ x_i^t, & \text{otherwise} \end{cases} \tag{10}$$

where $f(\cdot)$ represents the objective function in the minimization problems.

III. The Proposed APDE Algorithm

To avoid directly eliminating and ignoring the worse trial vectors in the selection of DE, a DCS strategy is designed, including pop_inc and pop_dec. The pop_inc strategy is proposed to give the worse trial vectors the chance to be added into the new population. On the other hand, the pop_dec strategy is designed to delete some hopeless solutions to avoid the gradual expansion of the population and to ensure the population size within a certain range

Algorithm 1: APDE

```
Begin
1:  Initialize pop and best;
2:  best_notChange = 0;
3:  For i = 1 : NP Do
4:      notChange_i = 0;
5:  While the terminal condition is not satisfied Do
6:      NPtmp = NP;
7:      For i = 1 : NPtmp Do
8:          Generate $v_i$ by the mutation operator as (4);
9:          Generate $u_i$ by the crossover operator as (8);
10:         Evaluate $u_i$;
11:         If $f(u_i) < f(x_i)$ Then            // selection
12:             $x_i \leftarrow u_i$, $notChange_i = 0$;
13:         Else
14:             $notChange_i = notChange_i + 1$;
15:             If $best\_notChange \geq T$ and $NP < NPmax$ Then
16:                 pop_inc($u_i$, pop);
17:         Update $x_{best}$ in the current iteration;
18:         Update $x_{worst}$ in the current iteration;
19:         If $f(x_i) < f(best)$ Then    // update best found so far
20:             $best \leftarrow x_i$, $f(best) = f(x_i)$;
21:             best_notChange = 0;
22:         Else
23:             best_notChange = best_notChange + 1;
24:     pop_dec(pop);
End
```

[*NPmin*, *NPmax*], where *NPmin* and *NPmax* are the lower and upper bound of the population size, respectively.

This section describes the APDE algorithm detailedly. Firstly, the framework of APDE algorithm is presented. Then, the two population control strategies, including pop_inc and pop_dec, are introduced.

A. The Framework of APDE

The details of APDE are shown in **Algorithm 1**. Firstly, the population ***pop***, ***best***, *best_notChange*, and ***notChange*** are initialized in lines 1 to 4, where ***best*** is the best solution found so far, and *best_notChange* records how many iterations ***best*** has not changed, and $notChange_i$ records how many iterations the solution $\boldsymbol{x_i}$ has not changed. In every iteration, *NP* needs to be saved in *NPtmp* before the **For** loop because the *NP* will change in the loop (pop_inc in line 16). The new solution $\boldsymbol{u_i}$ is generated and evaluated after mutation (in line 8) and crossover (in line 9). If $\boldsymbol{u_i}$ is better than the target vector $\boldsymbol{x_i}$, $\boldsymbol{u_i}$ will replace the old $\boldsymbol{x_i}$, and $notChange_i$ will be reset to zero in line 12; otherwise, $notChange_i$ will increase by one because no better solution is found, and we still give chance for the $\boldsymbol{u_i}$ entering the population. Therefore, if the best solution during the iteration has not been updated for a long time ($best\_notChange \geq T$), and the current population size *NP* is smaller than *NPmax*, then the pop_inc strategy will be used. After this, the best solution $\boldsymbol{x_{best}}$ in the current generation (in line 17), the worst solution in the current generation $\boldsymbol{x_{worst}}$ (in line 18), and the best solution found so far ***best***, and *best_notChange* (in lines 19 to 23) are updated, respectively. After the generation of the whole population, the pop_dec strategy is used to delete some hopeless solutions in line 24. Then, the process iterates until the terminal condition is satisfied. pop_inc and pop_dec are introduced in the following sections.

Algorithm 2: pop_inc(u_i, ***pop***)

```
Begin
1:  Add $u_i$ into pop;                // $pop_{NP+1} \leftarrow u_i$
2:  NP = NP + 1;
3:  $notChange_{NP} = 0$;
End
```

Algorithm 3: pop_dec(***pop***)

```
Begin
1:  If NP > NPmin Then
2:      NPtmp = NP;
3:      For i = 1 : NPtmp Do
4:          Calculate $dg(x_i)$;
5:          If $dg(x_i) > T$ and $x_i$ is not equal to $x_{best}$ Then
6:              Delete $x_i$ from pop;
7:              NP = NP − 1;
8:              If $NP \leq NPmin$ Then
9:                  Break;
End
```

B. The pop_inc Strategy

The classical selection operator in DE directly eliminates and ignores the trial vector ***u*** if it is not better than the target vector ***x***. However, these ignored vectors ***u***, generated by the mutation and crossover operations, will be helpful to increase the population diversity and may become the promising solutions in the later evolution. Therefore, pop_inc is designed to give these worse vectors ***u*** the opportunity to be added into the population, shown as **Algorithm 2**. For a minimization problem, if the solution ***best*** has been not changed for *T* iterations and *NP* is smaller than *NPmax* in line 15 of **Algorithm 1**, the trial vector $\boldsymbol{u_i}$ will be added to ***pop*** in line 1 of **Algorithm 2**, and *NP* and $notChange_{NP}$ are updated.

If there has been no improvement on ***best*** for a long time, it shows that the algorithm may have trapped into stagnation and fallen into the local optima. Then, the addition of some worse vectors may be helpful to disturb the population and jump from the local optima, which can also increase the population diversity.

C. The pop_dec Strategy

Due to the pop_inc strategy, the population size will increase gradually in the evolution. The evolution of the whole population will cost larger computational burden, and it will slow down the convergence speed of the APDE algorithm. Therefore, pop_dec is proposed to delete some inefficient solutions from the current population and ensure that *NP* is smaller than *NPmax*. As shown in line 24 of **Algorithm 1**, the pop_dec procedure is executed after an iteration of the evolution.

To decide which solutions to be deleted, a factor named "degradation value" (the *dg* value) of solutions is designed as

$$dg(\boldsymbol{x}_i) = \frac{f(\boldsymbol{x}_i) - f(\boldsymbol{best}) + 1.0}{f(\boldsymbol{x}_{worst}) - f(\boldsymbol{best}) + 1.0} \times notChange_i \tag{11}$$

The addition of the constant "1.0" is to avoid the divisor of zero. If $dg(\boldsymbol{x_i})$ is large, it represents that $f(\boldsymbol{x_i})$ is worse and close to $f(\boldsymbol{x_{worst}})$, or the solution $\boldsymbol{x_i}$ has not changed for a long time. Then, this solution $\boldsymbol{x_i}$ may be hopeless and will be deleted from the population.

TABLE I. Comparison results of APDE and All Compared Algorithms

Function	Statistics	APDE	DE$_{NP50}$	DE$_{NP100}$	DECC-DG	CCPSO2
f_1	Median	**2.80E-20**	5.05E-20	3.22E-14	1.51E+06	2.21E+02
	Mean	**1.42E-19**	1.25E-19	3.37E-14	3.63E+06	2.38E+02
	Stdev	**3.52E-19**	2.26E-19	1.32E-14	3.88E+06	1.40E+02
	p-value	-	9.14E-02$^{=}$	1.42E-09^{+}	1.41E-09^{+}	1.41E-09^{+}
f_2	Median	2.64E+03	2.33E+03	5.90E+03	1.27E+04	**7.74E+01**
	Mean	2.56E+03	2.34E+03	5.96E+03	1.30E+04	**7.92E+01**
	Stdev	2.76E+02	3.15E+02	2.55E+02	1.96E+03	**1.63E+01**
	p-value	-	4.46E-03^{-}	1.41E-09^{+}	1.41E-09^{+}	**1.41E-09^{-}**
f_3	Median	**2.00E+01**	2.13E+01	2.13E+01	2.07E+01	**2.00E+01**
	Mean	**2.00E+01**	2.13E+01	2.13E+01	2.07E+01	**2.00E+01**
	Stdev	**3.04E-02**	5.91E-03	5.25E-03	6.67E-03	**4.12E-04**
	p-value	-	9.67E-12^{+}	9.67E-12^{+}	9.67E-12^{+}	**8.10E-02$^{=}$**
f_4	Median	1.11E+11	3.25E+11	3.43E+11	9.33E+10	**6.29E+10**
	Mean	1.26E+11	3.40E+11	3.32E+11	1.05E+11	**6.80E+10**
	Stdev	1.03E+11	6.13E+10	5.99E+10	7.04E+10	**2.80E+10**
	p-value	-	5.54E-08^{+}	6.46E-07^{+}	2.37E-01$^{=}$	**1.65E-01$^{=}$**
f_5	Median	**2.75E+06**	6.43E+06	6.87E+06	3.00E+06	1.50E+07
	Mean	**2.94E+06**	6.34E+06	6.95E+06	2.85E+06	1.52E+07
	Stdev	**5.14E+05**	7.24E+05	6.58E+05	9.39E+05	2.38E+06
	p-value	-	1.41E-09^{+}	1.41E-09^{+}	1.43E-01$^{=}$	1.41E-09^{+}
f_6	Median	**1.03E+06**	1.06E+06	1.06E+06	1.06E+06	1.05E+06
	Mean	**1.03E+06**	1.06E+06	1.06E+06	1.06E+06	1.05E+06
	Stdev	**1.11E+04**	1.25E+03	1.13E+03	1.40E+03	3.54E+03
	p-value	-	9.11E-10^{+}	9.11E-10^{+}	9.11E-10^{+}	1.16E-07^{+}
f_7	Median	5.03E+08	1.90E+09	2.09E+09	**1.59E+08**	8.15E+08
	Mean	6.44E+08	1.94E+09	2.15E+09	**1.67E+08**	1.34E+09
	Stdev	4.73E+08	4.40E+08	6.64E+08	**6.82E+07**	1.07E+09
	p-value	-	4.21E-08^{+}	9.28E-09^{+}	**2.90E-09^{-}**	3.39E-03^{+}
f_8	Median	**4.76E+14**	6.05E+15	8.92E+15	1.28E+15	2.45E+15
	Mean	**5.18E+14**	6.14E+15	8.50E+15	1.38E+15	2.76E+15
	Stdev	**2.40E+14**	2.62E+15	2.58E+15	9.84E+14	1.56E+15
	p-value	-	1.41E-09^{+}	1.41E-09^{+}	3.84E-03^{+}	5.20E-09^{+}
f_9	Median	**2.44E+08**	4.70E+08	5.12E+08	3.20E+08	1.10E+09
	Mean	**2.47E+08**	4.70E+08	5.16E+08	3.27E+08	1.08E+09
	Stdev	**4.27E+07**	4.39E+07	4.51E+07	9.51E+07	2.61E+08
	p-value	-	1.41E-09^{+}	1.41E-09^{+}	1.27E-02^{+}	1.41E-09^{+}
f_{10}	Median	**9.13E+07**	9.32E+07	9.33E+07	9.45E+07	9.33E+07
	Mean	**9.14E+07**	9.32E+07	9.33E+07	9.44E+07	9.32E+07
	Stdev	**4.79E+05**	2.73E+05	2.77E+05	2.53E+05	5.62E+05
	p-value	-	3.24E-09^{+}	1.99E-09^{+}	1.25E-09^{+}	3.92E-09^{+}
f_{11}	Median	**2.24E+09**	3.83E+10	4.33E+10	3.61E+09	1.48E+11
	Mean	**1.09E+10**	4.80E+10	6.83E+10	4.70E+10	1.86E+11
	Stdev	**3.37E+10**	2.11E+10	4.84E+10	9.19E+10	1.42E+11
	p-value	-	2.87E-08^{+}	2.57E-08^{+}	4.45E-04^{+}	8.28E-09^{+}
f_{12}	Median	**1.96E+03**	2.07E+03	2.06E+03	1.38E+11	3.20E+03
	Mean	**1.99E+03**	2.08E+03	2.02E+03	1.33E+11	3.19E+03
	Stdev	**1.55E+02**	1.88E+02	1.74E+02	1.27E+10	3.17E+02
	p-value	-	4.35E-02^{+}	6.00E-03^{+}	1.41E-09^{+}	1.41E-09^{+}
f_{13}	Median	**6.92E+09**	3.28E+10	4.54E+10	8.02E+09	8.88E+09
	Mean	**8.02E+09**	3.40E+10	4.45E+10	7.07E+09	9.01E+09
	Stdev	**4.21E+09**	5.66E+09	6.51E+09	1.91E+09	2.74E+09
	p-value	-	1.41E-09^{+}	1.42E-09^{+}	9.77E-01$^{=}$	1.51E-01$^{=}$
f_{14}	Median	**8.79E+10**	3.35E+11	4.42E+11	2.78E+11	1.57E+11
	Mean	**9.72E+10**	3.42E+11	4.68E+11	3.55E+11	2.29E+11
	Stdev	**5.54E+10**	9.35E+10	1.51E+11	3.05E+11	1.77E+11
	p-value	-	4.63E-09^{+}	1.80E-09^{+}	3.20E-08^{+}	9.61E-05^{+}
f_{15}	Median	1.09E+07	6.10E+07	8.23E+07	**4.93E+06**	2.10E+07
	Mean	1.11E+07	6.07E+07	8.19E+07	**5.51E+06**	2.11E+07
	Stdev	1.48E+06	6.73E+06	5.76E+06	**2.13E+06**	4.13E+06
	p-value	-	1.41E-09^{+}	1.40E-09^{+}	**8.23E-09^{-}**	2.89E-09^{+}
+/−/=		-	13/1/1	15/0/0	10/2/3	11/1/3

The details of pop_dec are shown in **Algorithm 3**. If *NP* is bigger than *NPmin*, some inefficient solutions should be deleted from the population. For each solution $\boldsymbol{x_i}$, the degradation value $dg(\boldsymbol{x_i})$ is calculated as (11). If $dg(\boldsymbol{x_i})$ is bigger than *T* which is the same as that in line 15 of **Algorithm 1**, and $\boldsymbol{x_i}$ is not equal to $\boldsymbol{x_{best}}$, this solution will be deleted in line 6 of **Algorithm 3**, and *NP* will reduce by one. After deletion, if *NP* reduces to *NPmin*, the **For** loop among lines 3 to 9 will end. Therefore, the population size *NP* can be clamped in a certain range between *NPmin* and *NPmax*.

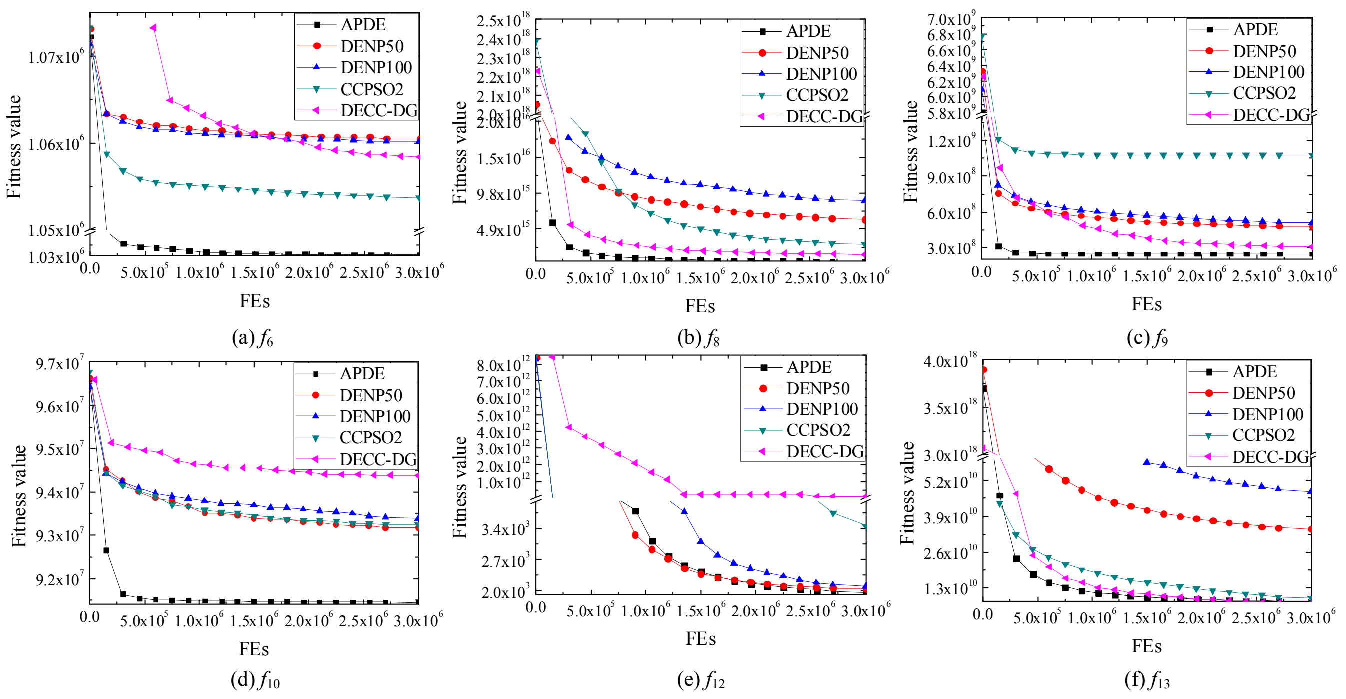

Fig. 1. Convergence curves of all algorithms. (a) f_6. (b) f_8. (c) f_9. (d) f_{10}. (e) f_{12}. (f) f_{13}.

Instead of deleting solutions only by comparing their fitness values, the design of the degradation value $dg(\boldsymbol{x}_i)$ takes two aspects into consideration, including the fitness values and the update frequency of solutions. Besides, there is no need to sort solutions by the fitness values in pop_dec, and it helps to save time.

IV. Experimental Results and Analysis

A. Experimental Settings

In the experiment, the test functions with 1000 dimensions in CEC'2013 on the LSGO problems are used to verify the performance of APDE [15]. The parameters in APDE are set as F=0.5, CR=0.1, $NPmin$=50, $NPmax$=100, and T=15, and DE/best/1 and the binomial crossover are used as the mutaion and crossover operator, respectively. Two traditional DE algorithms are used only with different values of NP (50 and 100), named as DE_{NP50} and DE_{NP100}, respectively. Besides, two LSGO algorithms are compared, including CCPSO2 [16] and DECC-DG [17]. The terminal condition of all algorithms is the maximum fitness evaluations (FEs) of 3×10^6. Each algorithm is executed for 25 times independently on each benchmark function.

B. Comparisons of Experimental Results

The comparative results are shown in Table I, where the data in **bold** represent the best solutions. The Wilcoxon rank-sum test at a 5% significance level is used for the significance test and the statistical comparisons [18]. The last row of the table (+/−/=) represents the number of functions where APDE is superior to, inferior to, and equal to the compared algorithms, respectively. Fig. 1 shows the convergence curves of all algorithms on some test functions.

From Table I, it can be concluded that APDE performs better than other four compared algorithms statistically, either on the fully-separable functions (f_1 and f_3) or the separable functions (f_6, f_8 to f_{12}, and f_{14}). Detailedly, compared with DE_{NP50} and DE_{NP100} with the traditional selection, APDE gives the worse trial vectors the chance to be involved in the new generation, which may help to jump from the local optima. Besides, the APDE deletes the solutions with higher degradation values to assign the limited computing resources to more promising solutions. Compared with the two LSGO algorithms, APDE has a faster convergence speed, as shown in Fig. 1. It may be because the dynamic population size of APDE helps to increase the population diversity and find better solutions.

C. Parameter Investigation

This section studies the parameter T of the proposed algorithm. Five different values are tested (3, 6, 9, 12, and 15), recorded as APDE_{T3}, APDE_{T6}, APDE_{T9}, APDE_{T12}, and APDE (with the default value 15), respectively. The experimental results are shown as Table II.

It can be concluded from Table II that APDE with the higher value of T (T=15) gets general better results than the algorithms with other parameter settings statistically. The reason is that if the value of T is smaller, the pop_inc strategy and the pop_dec strategy will be executed more frequently as shown in **Algorithm 1** and **Algorithm 3**, which costs much time and slows down the evolution.

V. Conclusions

In this paper, APDE with DCS is proposed to make full use of the information of the new generated solutions. The DCS strategy includes pop_inc and pop_dec. Firstly, the pop_inc strategy is designed to give the new generated solutions more chance to join in the new population, which can help the DE algorithm increase the population diversity. If the best solution has not be updated for many times, pop_inc will add the trial vectors (after the crossover operator) even though it is worse than the target vectors (before the mutation operator) into the next generation. Secondly, the pop_dec strategy is designed to delete some inefficient solutions with higher degradation values from the current population, since the population size will increase

TABLE II. COMPARISON RESULTS OF APDE (T = 15) WITH DIFFERENT VALUES OF THE PARAMETER T

Function	Statistics	APDE	APDE$_{T3}$	APDE$_{T6}$	APDE$_{T9}$	APDE$_{T12}$
f_1	Median	2.80E-20	2.03E-15	2.52E-19	3.57E-20	**1.73E-20**
	Mean	1.42E-19	1.52E-12	5.38E-17	3.69E-19	**5.91E-20**
	Stdev	3.52E-19	5.04E-12	1.90E-16	7.50E-19	**1.25E-19**
	p-value	-	1.42E-09^{+}	5.88E-06^{+}	1.71E-01$^{=}$	**1.94E-01**$^{=}$
f_2	Median	**2.64E+03**	5.55E+03	3.39E+03	2.88E+03	2.63E+03
	Mean	**2.56E+03**	5.55E+03	3.40E+03	2.88E+03	2.64E+03
	Stdev	**2.76E+02**	5.71E+02	4.42E+02	2.43E+02	3.19E+02
	p-value	-	1.41E-09^{+}	2.85E-08^{+}	3.30E-04^{+}	5.41E-01$^{=}$
f_3	Median	**2.00E+01**	**2.00E+01**	**2.00E+01**	**2.00E+01**	**2.00E+01**
	Mean	**2.00E+01**	**2.00E+01**	**2.00E+01**	**2.00E+01**	**2.00E+01**
	Stdev	**3.04E-02**	**2.52E-03**	**2.88E-03**	**4.42E-03**	**5.48E-03**
	p-value	-	**8.10E-02**$^{=}$	**8.10E-02**$^{=}$	**8.10E-02**$^{=}$	**8.10E-02**$^{=}$
f_4	Median	1.11E+11	3.02E+10	**1.85E+10**	4.08E+10	3.76E+10
	Mean	1.26E+11	3.48E+10	**3.86E+10**	7.64E+10	7.62E+10
	Stdev	1.03E+11	2.75E+10	**5.91E+10**	8.90E+10	8.60E+10
	p-value	-	1.37E-03^{-}	**5.14E-04**$^{-}$	7.11E-02$^{=}$	7.91E-02$^{=}$
f_5	Median	**2.75E+06**	3.49E+06	3.94E+06	3.49E+06	3.24E+06
	Mean	**2.94E+06**	3.40E+06	4.09E+06	3.67E+06	3.31E+06
	Stdev	**5.14E+05**	5.37E+05	1.24E+06	7.91E+05	7.55E+05
	p-value	-	1.72E-03^{+}	5.52E-04^{+}	7.60E-04^{+}	6.25E-02$^{=}$
f_6	Median	**1.03E+06**	1.06E+06	**1.02E+06**	1.06E+06	1.06E+06
	Mean	**1.03E+06**	1.06E+06	**1.03E+06**	1.05E+06	1.05E+06
	Stdev	**1.11E+04**	1.22E+04	**1.90E+04**	1.58E+04	1.44E+04
	p-value	-	2.68E-07^{+}	**5.69E-01**$^{=}$	9.24E-04^{+}	2.90E-05^{+}
f_7	Median	5.03E+08	9.93E+07	1.20E+08	1.59E+08	**1.63E+08**
	Mean	6.44E+08	3.13E+08	3.16E+08	3.44E+08	**2.14E+08**
	Stdev	4.73E+08	4.38E+08	4.56E+08	3.42E+08	**1.37E+08**
	p-value	-	2.85E-04^{-}	1.67E-04^{-}	1.62E-03^{-}	**9.15E-07**$^{-}$
f_8	Median	**4.76E+14**	4.88E+14	9.71E+14	8.49E+14	6.62E+14
	Mean	**5.18E+14**	8.22E+14	1.03E+15	8.83E+14	8.02E+14
	Stdev	**2.40E+14**	1.25E+15	6.22E+14	4.95E+14	4.72E+14
	p-value	-	9.07E-01$^{=}$	2.03E-03^{+}	1.78E-03^{+}	1.61E-02^{+}
f_9	Median	**2.44E+08**	2.98E+08	4.37E+08	2.67E+08	2.50E+08
	Mean	**2.47E+08**	3.19E+08	4.17E+08	2.69E+08	2.53E+08
	Stdev	**4.27E+07**	5.30E+07	6.92E+07	4.30E+07	4.01E+07
	p-value	-	1.11E-05^{+}	1.63E-08^{+}	1.12E-01$^{=}$	5.41E-01$^{=}$
f_{10}	Median	**9.13E+07**	9.29E+07	9.19E+07	9.17E+07	9.22E+07
	Mean	**9.14E+07**	9.30E+07	9.21E+07	9.19E+07	9.21E+07
	Stdev	**4.79E+05**	9.56E+05	9.02E+05	9.25E+05	6.40E+05
	p-value	-	3.22E-07^{+}	9.41E-03^{+}	1.17E-01$^{=}$	1.08E-03^{+}
f_{11}	Median	2.24E+09	2.24E+09	4.11E+09	3.03E+09	**2.12E+09**
	Mean	1.09E+10	6.41E+09	1.17E+10	1.88E+10	**5.78E+09**
	Stdev	3.37E+10	9.89E+09	1.83E+10	3.03E+10	**1.12E+10**
	p-value	-	8.23E-01$^{=}$	9.91E-02$^{=}$	2.22E-01$^{=}$	**4.61E-01**$^{=}$
f_{12}	Median	1.96E+03	2.11E+03	2.06E+03	2.00E+03	**1.93E+03**
	Mean	1.99E+03	2.07E+03	2.04E+03	1.99E+03	**1.96E+03**
	Stdev	1.55E+02	2.17E+02	1.82E+02	1.20E+02	**1.51E+02**
	p-value	-	1.40E-01$^{=}$	8.58E-02$^{=}$	7.34E-01$^{=}$	**5.60E-01**$^{=}$
f_{13}	Median	6.92E+09	9.50E+09	4.80E+09	**4.49E+09**	9.48E+09
	Mean	8.02E+09	9.92E+09	5.30E+09	**4.91E+09**	9.57E+09
	Stdev	4.21E+09	4.50E+09	1.94E+09	**1.69E+09**	3.51E+09
	p-value	-	1.14E-01$^{=}$	1.79E-02^{-}	**7.85E-03**$^{-}$	1.28E-01$^{=}$
f_{14}	Median	**8.79E+10**	1.24E+11	8.76E+10	7.64E+10	1.05E+11
	Mean	**9.72E+10**	1.44E+11	1.07E+11	1.21E+11	1.05E+11
	Stdev	**5.54E+10**	9.37E+10	8.85E+10	1.05E+11	3.94E+10
	p-value	-	2.20E-02^{+}	8.23E-01$^{=}$	9.77E-01$^{=}$	3.62E-01$^{=}$
f_{15}	Median	1.09E+07	1.40E+07	**1.03E+07**	1.05E+07	1.05E+07
	Mean	1.11E+07	1.42E+07	**1.03E+07**	1.07E+07	1.06E+07
	Stdev	1.48E+06	3.20E+06	**1.30E+06**	1.09E+06	1.40E+06
	p-value	-	3.01E-05^{+}	**7.41E-02**$^{=}$	5.80E-01$^{=}$	3.08E-01$^{=}$
+/−/=		-	8/2/5	6/3/6	4/2/9	3/1/11

gradually after the pop_inc strategy. After the dual population control strategies, the population size can be maintained in a certain range. The experimental results show that APDE performs general better and converges general faster than the traditional DE algorithms and other competent algorithms on the LSGO problems.

We will further improve the DE algorithm to solve the practical problems, such as the supply chain network design [19][20] and the cloud workflow scheduling [21][22], and use the DCS to improve other algorithms, such as particle swarm optimization [23][24].

REFERENCES

[1] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim*., vol. 11, no. 4, pp. 341–359, Dec. 1997.

[2] Y. F. Ge, W. J. Yu, Z. H. Zhan, and J. Zhang, "Competition-based distributed differential evolution," In *Proc. IEEE Congr. Evol. Comput.*, 2018, pp. 1-8.

[3] Z. H. Zhan, *et al.*, "Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 704-716, Mar. 2017.

[4] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution–an updated survey," *Swarm Evol. Comput*., vol. 27, pp. 1-30, April 2016.

[5] A. K. Qin, V. L. Huang, P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, April 2009.

[6] R. C. Pedrosa Silva, R. A. Lopes, and F. G. Guimarães, "Self-adaptive mutation in the differential evolution," In *Proc. ACM Genet. Evol. Comput.*, 2011, pp. 1939-1946.

[7] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," In *Proc. IEEE Congr. Evol. Comput.*, 2013, pp. 71–78.

[8] X. F. Liu, *et al.*, "Historical and heuristic-based adaptive differential evolution," *IEEE Trans. Syst. Man Cybern.: Syst.*, vol. 49, no. 12, pp. 2623-2635, Dec. 2019.

[9] W. Zhu, Y. Tang, J. A. Fang, and W. Zhang, "Adaptive population tuning scheme for differential evolution," *Inf. Sci.*, vol. 223, pp. 164-191, Feb. 2013.

[10] Z. H. Zhan, Z. J. Wang, H. Jin, and J. Zhang, "Adaptive distributed differential evolution," *IEEE Trans. Cybern.*, to be published. DOI: 10.1109/TCYB.2019.2944873, Oct. 2019.

[11] Y. L. Li, Z. H. Zhan, Y. J. Gong, W. N. Chen, J. Zhang, and Y. Li, "Differential evolution with an evolution path: A DEEP evolutionary algorithm," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1798-1810, Sept. 2014.

[12] W. J. Yu, *et al.*, "Differential evolution with two-level parameter adaptation," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1080-1099, July 2013.

[13] D. M. Cabrera, "Evolutionary algorithms for large-scale global optimisation: a snapshot, trends and challenges," *Prog. Artif. Intell.*, vol. 5, no. 2, pp. 85-89, Feb. 2016.

[14] Z. J. Wang, Z. H. Zhan, S. Kwong, H. Jin, and J. Zhang, "Adaptive granularity learning distributed particle swarm optimization for large-scale optimization," *IEEE Trans. Cybern.*, to be published. DOI: 10.1109/TCYB.2020.2977956. Feb. 2020.

[15] X. Li, K. Tang, M. N. Omidvar, Z. Yang, and K. Qin, "Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization," Technical Report, Evolutionary Computation and Machine Learning Group, RMIT University, Australia, 2013.

[16] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 2, pp. 210-224, Apri. 2012.

[17] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Trans.Evol. Comput.*, vol. 18, no. 3, pp. 378-393, June 2013.

[18] *Wilcoxon Rank-sum Test*, Texas Instruments Inc., Dallas, TX, USA, pp. 135-136.

[19] X. Zhang, K. J. Du, Z. H. Zhan, S. Kwong, T. L. Gu, and J. Zhang. "Cooperative coevolutionary bare-bones particle swarm optimization with function independent decomposition for large-scale supply chain network design with uncertainties," *IEEE Trans. Cybern.*, to be published. DOI: 10.1109/TCYB.2019.2937565.

[20] X. Zhang, Z. H. Zhan, and J. Zhang. "A fast efficient local search-based algorithm for multi-objective supply chain configuration problem," *IEEE Access*, to be published. DOI: 10.1109/ACCESS.2020.2983473.

[21] Z. G. Chen, *et al.*, "Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach," *IEEE Trans. Cybern.*,vol. 49, no. 8, pp. 2912-2926, Aug. 2019.

[22] Z. J. Wang, *et al.*, "Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling," *IEEE Trans. Cybern.*, to be published, DOI: 10.1109/TCYB.2019.2933499.

[23] X. Xia, *et al.*, "Triple archives particle swarm optimization," *IEEE Trans. Cybern.*, to be published, DOI: 10.1109/TCYB.2019.2943928.

[24] X. F. Liu, Z. H. Zhan, Y. Gao, J. Zhang, S. Kwong, and J. Zhang, "Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 4, pp. 587-602, Aug. 2018.