Applied Soft Computing 57 (2017) 60–73

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Full Length Article

An adaptive differential evolution algorithm with an aging leader and challengers mechanism

C.M. Fu [a], C. Jiang [a,*], G.S. Chen [b], Q.M. Liu [a]

[a] State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body, College of Mechanical and Vehicle Engineering, Hunan University, Changsha 410082, PR China
[b] School of Mechanical Engineering, Nanjing University of Science and Technology, Nanjing City 210094, PR China

ARTICLE INFO

Article history:
Received 19 August 2016
Received in revised form 13 February 2017
Accepted 18 March 2017
Available online 30 March 2017

Keywords:
Global optimization
Evolutionary algorithms
Differential evolution
Adaptive parameter control
Aging leader

ABSTRACT

An adaptive differential evolution algorithm with an aging leader and challengers mechanism, called ADE-ALC, is proposed to solve optimization problems. In ADE-ALC algorithm, the aging mechanism is introduced into the framework of differential evolution to maintain diversity of the population. The key control parameters are adaptively updated based on given probability distributions which could learn from their successful experiences to generate the promising parameters at the next generation. One of the two local search operators is randomly selected to generate challengers which are beneficial for increasing the diversity of population. Finally, the effectiveness of the ADE-ALC algorithm is verified by the numerical results of twenty-five benchmark test functions.

1. Introduction

Differential evolution (DE) algorithm proposed by Storn and Price [1], which shares similar characteristics with other population-based evolutionary algorithms (EAs), is widely utilized to solve global optimization problems. It utilizes the distance and direction information from the current population to guide the evolutionary population to search for the global optimum generation by generation. The main characteristic of DE different from other EAs is the mutant strategy which generates new mutant individuals by adding the weighted difference between two individuals to a third individual. Then, the specific crossover operator is employed to combine the mutant individual with the parent individual to generate offspring individual. Finally, the better individuals from parent and offspring population are selected to the next generation. The above operations repeat to search for the global optimum. DE has been attracted more and more attention because of its simplicity and effectiveness and it has been successfully applied in different fields, such as railway timetable scheduling [2], structural design [3,4], economic dispatch [5], etc.

Since the first publication about DE algorithm, various significant DE variants have been proposed to improve the performance of DE. So far, the existing DE variants are mainly classified into three categories. The first category of DE variants achieves improvements by designing specific mechanisms to tune its control parameters (population size, scaling factor and crossover rate). Choosing suitable parameters for different optimization problems is an arduous task and requires repeatedly tuning. Hence, some adaptive and self-adaptive DE algorithms were developed according to some different feedback information [6,7]. In order to avoid tuning population size, Zhu et al. [8] proposed an adaptive population tuning scheme (APTS) for DE which mainly included two parts: an elite-based population-incremental strategy and an inferior-based population-cut strategy. Ghosh et al. [9] designed the fitness-adaptive scheme to control the scaling factor and crossover rate, which could maintain the strong explorative ability as well as eradicate the maximum probability of premature convergence. Fan and Yan [10] presented the self-adaptive DE with discrete mutation control parameters. Draa et al. [11] designed six sinusoidal-based formulas to automatically adjust the scaling factor and crossover rate. Salehinejad et al. [12] proposed a micro-DE with small population the vectorized random mutation factor which could effectively overcome the premature convergence and high risk of stagnation.

The second category aims to design new mutant strategies or introduce new techniques to adaptively select mutant strategies to enhance performance. Fan and Lampinen [13] proposed the trigonometric mutant strategy to accelerate convergence rate. Zhang and Sanderson [14] revised the DE/current-to-best/1 strat-

* Corresponding author.
E-mail address: jiangc@hnu.edu.cn (C. Jiang).

http://dx.doi.org/10.1016/j.asoc.2017.03.032
1568-4946/

egy to develop the new DE/current-*p*best/1 strategy, and the *p*best individual was selected from one of the best 100*p*% individuals in the current population. Das et al. [15] developed an improved DE/current-to-best/1 strategy which utilized the concept of the global and local neighborhood of each population member. Cai and Wang [16] utilized the neighborhood and direct information to enhance the performance of DE. Qin et al. [17] proposed the SaDE algorithm in which both mutant strategies and their associated control parameter values were adaptively selected by learning from their previous experiences. Similarly, probability matching (PM) and adaptive pursuit (AP) techniques [18] were employed to autonomously select the most suitable mutant strategies from a given strategy pool according to their recent effect on the evolutionary process. Pan et al. [19] presented the SspDE algorithm where each target individual had an associated mutant strategy set, a scaling factor set, and a crossover rate set. Wang et al. [20] developed the composite mutant strategies and control parameters (CoDE) algorithm. Mallipeddi et al. [21] presented the ensemble of mutation strategies and control parameters with the DE algorithm.

The third category introduces the global or local search techniques to construct a hybrid algorithm to improve performance. The hybrid algorithms primarily aim to combine the advantages of two or more different algorithms to create more powerful optimizers. Over the past two decades, DE has been hybridized with several global optimization algorithms, such as genetic operator [22], particle swarm optimization (PSO) [23], estimation of distribution algorithm (EDA) [24], biogeography-based optimization (BBO) [25], artificial bee colony (ABC) [26,27], teaching-learning algorithm [28], etc. At the same time, various local search optimizers are also introduced into the framework of DE to improve its performance [29]. Omran et al. [30] proposed the bare-bones differential evolution (BBDE) algorithm inspired by the concept of bare-bones PSO.

From the above discussions, the crucial aims of different DE variant algorithms to solve optimization problems are keeping the fast-converging speed while obtaining the global optimum. The premature convergence of DE variant can be efficiently avoided by increasing the diversity of evolutionary population. During the last two decades, the concept of aging mechanism was introduced to evolutionary algorithm and has been obtained increasing attention [31–33]. Aging means the act of getting old. Goldsmith [34] stated that aging was a significant and important characteristic for evolution because it increased the diversity of the evolutionary population, which is beneficial for searching global optimal solution.

In this paper, the aging mechanism is designed to the framework of DE algorithm which aims to increase the diversity while retaining the fast-converging rate. Hence, this paper proposes an adaptive differential evolution with an aging leader and challengers (ADE-ALC) algorithm for global optimization problems. The principal merits of the proposed ADE-ALC algorithm are three aspects. Firstly, for the best individual as the leader with strong leadership, it could attract the whole population with the fast-converging feature. Secondly, for the leader with weak leading power, it gets aged quickly and leaves new opportunities for other challengers to lead the population. Hence, this scheme could increase diversity of the population and prevent the premature convergence. Thirdly, two local search operators are utilized to generate the challenger which also brings in diversity, which is beneficial to solve multimodal optimization problems.

The remainder of this paper is organized as follows. Section 2 introduces the basic operations of DE. Section 3 describes the algorithmic details of the proposed ADE-ALC approach. Section 4 studies the effectiveness and performance of the proposed method which is verified by the contrast tests with several evolutionary algorithms on empirical functions. In Section 5, an engineering application is conducted by the proposed method. Finally, conclusions are drawn in Section 6.

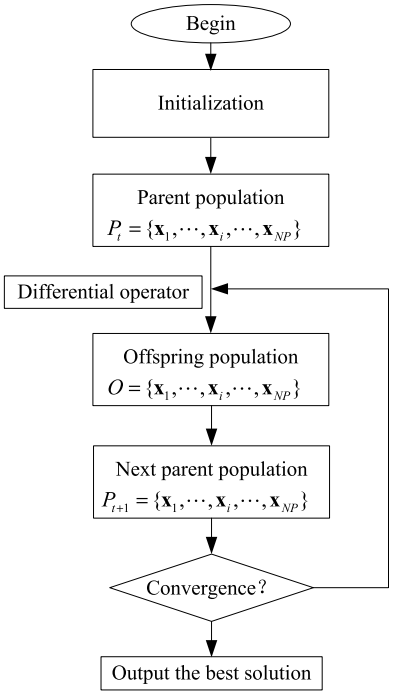

Fig. 1. The flowchart of classical DE.

2. The basics of differential evolution

Without loss of generality, an optimization problem is defined as:

$$\begin{aligned} &\min f(\mathbf{x}) \\ &\text{s.t.}\mathbf{x} \in S \end{aligned} \tag{1}$$

where $f(\mathbf{x})$ is an objective function, $\mathbf{x}$ is an n-dimensional design vector, and S is the search region. DE has been extensively applied to solve different optimization problems because of its simplicity and effectiveness [1]. The flowchart of the classical DE is shown in Fig. 1. The specific operators including initialization, mutation, crossover and selection will be sequentially introduced in the following text. In the initialization step, it generates NP initial population $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,j}, \ldots, x_{i,n})(i = 1,2,\ldots,NP)$, which is defined as:

$$x_{i,j} = x_j^{\min} + rand \times (x_j^{\max} - x_j^{\min}) \tag{2}$$

where *rand* denotes a real number randomly generated between 0 and 1, $x_j^{\min}$ and $x_j^{\max}$ are the lower and upper bounds of the variable x_j, respectively.

After initialization, the mutant strategy is employed to generate a mutant vector $\mathbf{v}_{i,g} = (v_{i,1,g}, \ldots, v_{i,j,g}, \ldots, v_{i,n,g})$ by its corresponding target vector $\mathbf{x}_{i,g} = (x_{i,1,g}, \ldots, x_{i,j,g}, \ldots, x_{i,n,g})$. The following mutant strategies are generally utilized [35]:

DE/rand/1:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r_1,g} + F \times (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}) \tag{3}$$

DE/best/1:

$$\mathbf{v}_{i,g} = \mathbf{x}_{\text{best},g} + F \times (\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}) \tag{4}$$

DE/current-to-rand/1:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \times (\mathbf{x}_{r_1,g} - \mathbf{x}_{i,g}) + F \times (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}) \tag{5}$$

DE/current-to-best/1:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \times (\mathbf{x}_{\text{best},g} - \mathbf{x}_{i,g}) + F \times (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \quad (6)$$

where indices r_1, r_2 and r_3 are mutually exclusive integers randomly selected from [1, *NP*] and are also different from the *i*-th individual; *F* is a real number between 0 and 1; and $\mathbf{x}_{\text{best},g}$ is the best individual in generation *g*. Subsequently, a trial vector $\mathbf{u}_{i,g} = (u_{i,1,g}, ..., u_{i,j,g}, ..., u_{i,n,g})$ is generated by the binomial crossover or exponential crossover. The binomial crossover is defined as:

$$u_{i,j,g} = \begin{cases} v_{i,j,g}, \text{ if}(r\ and_j \le CR) or j = j_{\text{r and}} \\ x_{i,j,g}, \text{ otherwise} \end{cases} \quad (7)$$

where *CR* is the crossover probability rate and j_{rand} is an integer randomly generated from the range [1,*n*]. The binomial crossover operator copies the *j*-th variable of mutant vector $\mathbf{v}_{i,g}$ to its corresponding element in the trial vector $\mathbf{u}_{i,g}$ if it meets the condition. If the *j*-th component $u_{i,j,g}$ of the trial vector is out of the boundary, it should be reset as:

$$u_{i,j,g} = \begin{cases} \left(x_j^{\min} + u_{i,j,g}\right)/2, \text{ if} u_{i,j,g} < x_j^{\min} \\ \left(x_j^{\max} + u_{i,j,g}\right)/2, \text{ if} u_{i,j,g} > x_j^{\max} \\ u_{i,j,g}, \text{ otherwise} \end{cases} \quad (8)$$

Then, the target vector $\mathbf{x}_{i,g}$ is compared with its trial vector $\mathbf{u}_{i,g}$ according to their function values $f(\cdot)$, and the better one denoted as $\mathbf{x}_{i,g+1}$ will survive into the next generation population. For a minimization problem, the selection operation can be expressed as:

$$\mathbf{x}_{i,,g+1} = \begin{cases} \mathbf{u}_{i,g}, \mathit{iff}(\mathbf{u}_{i,g}) \le f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g}, \text{ otherwise} \end{cases} \quad (9)$$

The above processes are repeated generation by generation until the termination criterion is met.

For some typical adaptive DE variants, every individual $\mathbf{x}_{i,g}$ in the evolutionary population is associated with its own parameters F_i and CR_i. If the improvement value $\Delta_{i,g} = f\left(\mathbf{x}_{i,g}\right) - f\left(\mathbf{u}_{i,g}\right)$ is positive, the corresponding parameters F_i and CR_i used to generate $\mathbf{u}_{i,g}$ are called a successful scaling factor and a successful crossover probability, respectively.

3. The formulation of ADE-ALC

For a DE algorithm, it is critical to retain the diversity of evolutionary population when solving complicated multimodal optimization problems. Goldsmith [34] stated that a reasonable aging mechanism could increase the diversity of the evolutionary population, so that it could increase the maximum probability of avoiding local optimum.

Hence, the proposed ADE-ALC with a specific aging mechanism aims to achieve the global optimal solution with fast-converging speed when solving optimization problems. The aging mechanism only acts on the leader of the population rather than every individual in the whole population in order to reduce the complexity. Specifically, an individual chosen from the current population is regarded as the leader which could lead the evolutionary population to search for optimal individual within a limited lifespan. The lifespan of the leader is adaptively changed according to its leading power. An optimal lifespan plays an important role in improving the performance of evolution. When the lifespan of the leader is exhausted, a new individual generated by a local search operator emerge to challenge and claim the leading power of the current leader. This mechanism can provide new opportunities for other individuals to lead the population and thus increase the diversity of the population. ADE-ALC employs the best individual as the leader of the population within a limited lifespan to generate new mutant individuals. The flowchart of ADE-ALC is illustrated in Fig. 2 and the computational steps are given as follows.

Step 1 Initialization. The initial individuals $P_0 = \left\{\mathbf{x}_1, \cdots, \mathbf{x}_i, \cdots, \mathbf{x}_{NP}\right\}$ are randomly generated by Eq. (2) within the search region *S*. The best individual in the current population is selected as the leader. Its initial age A_0 is set to 0 and the lifespan θ is set to an initial value θ_0.

Step 2 Generate the new offspring. In ADE-ALC algorithm, the mutant strategy is defined as DE/current-to-leader/1 strategy described in the following section. Then, the binomial crossover is utilized to generate the trial individuals which form the offspring population. Moreover, an adaptive mechanism is utilized to control the critical parameters *F* and *CR*.

Step 3 Update the leader and select the better individuals to construct the next population. If the fitness value of best individual in current population is better than the fitness value of the leader, the best individual is regarded as the leader in this iteration. Furthermore, the better *NP* individuals are selected from the offspring and parent population to construct the next parent population.

Step 4 Adjust the lifespan. The lifespan θ is adaptively tuned according to its leading power. Then, the age of the leader increases 1. If the condition (i.e.$A \ge \theta_0$) is met, go to step 5. Otherwise, go to step 7.

Step 5 Generate a challenger. A new challenger which is utilized to challenge the leading power of the old leader is generated by one of the two local search operators.

Step 6 Evaluate the leading power of the challenger. If the challenger has a strong leading power, it replaces the old leader. The age and lifespan of the leader are reset to $A = 0$ and $\theta = \theta_0$, respectively. Otherwise, the leader remains unchanged and the age is reset to $A = \theta - 1$.

Step 7 Check the terminal condition. If the number of fitness function evaluations (*FEs*) is larger than the given maximum fitness function evaluations (*Max_FEs*), the algorithm stops and outputs the final solution. Otherwise, return to step 2 for the next generation.

From the above computational steps of the ADE-ALC algorithm, four key steps are required to explain in detail. The first one is to design the adaptive mechanism to control parameters *F* and *CR*, which adaptively change according to their historical experiences. The second one is to adjust the lifespan of the leader based on its leading power. The third one is to generate a challenger to challenge the old leader in order to increase new information to keep the diversity of population. The last one is to design the criterion mechanism to evaluate the leading power of the challenger and to judge whether to replace the old leader. Then, the following four subsections are utilized to explain the four steps one by one.

3.1. Control parameters adaption

In the ADE-ALC algorithm, DE/current-to-leader/1 strategy utilizes the information of the best solution as the leader in the current population to generate the mutant individual $\mathbf{v}_i$, which is defined as:

$$\mathbf{v}_i = \mathbf{x}_i + F_i \times (\mathbf{x}_{\text{leader}} - \mathbf{x}_i) + F_i \times \left(\mathbf{x}_{r_1} - \mathbf{x}_{r_2}\right) \quad (10)$$

where $\mathbf{x}_{\text{leader}}$ is the best individual in the current population and F_i is the scaling factor of individual $\mathbf{x}_i$. The proposed strategy's difference from DE/current-to-best/1 is that here $\mathbf{x}_{\text{leader}}$ represents the best individual during its leader's lifetime. This greedy strategy may lead to premature convergence and is usually less reliable, especially for solving the complicated multimodal problems [36]. Adaptive parameters' mechanism is helpful to increase the diversity of the evolutionary population and improve the robustness of the algorithm. Hence, every individual $\mathbf{x}_i$ owns its different param-

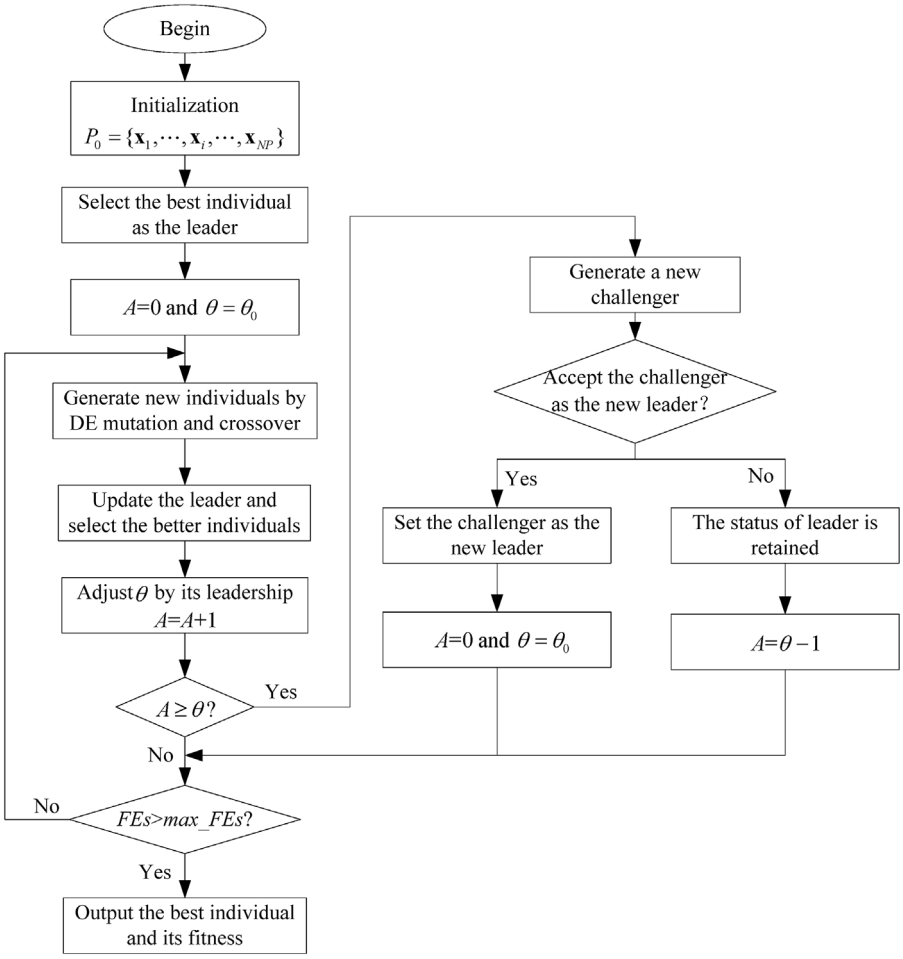

Fig. 2. Flowchart of the proposed ADE-ALC.

eters F_i and CR_i which are adaptively changed according to their recent historical successful experiences.

In the g-th iteration, the scaling factor F_i of the individual $\mathbf{x}_{i,g}$ is generated by a Cauchy distribution function Cauchy(·)with the location parameter μ_F and scale parameter 0.1. The Cauchy distribution is defined as [14]:

$$F_i = \text{Cauchy}_i(\mu_F, 0.1) \tag{11}$$

If the value of F_i goes beyond the interval [0, 1], it should be regenerated. The location parameter μ_F is initialized to be 0.5 and is updated by its historical experiences as:

$$\mu_F = (1 - c) \cdot \mu_F + c \cdot \text{mean}_L(S_F) \tag{12}$$

where $\text{mean}_L(\cdot)$ is the Lehmer mean, S_F means the set of all successful scaling factors in the g-th generation.

Similarly, in the g-th generation, the crossover rate CR_i is separately generated by the normal distribution function Normal(·) with the mean μ_{CR} and standard deviation 0.1:

$$CR_i = \text{Normal}_i(\mu_{CR}, 0.1) \tag{13}$$

If the values of CR_i exceeds the interval [0, 1], it should be regenerated. The mean μ_{CR} is initialized to be 0.5 and then updated as:

$$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot \text{mean}_A(S_{CR}) \tag{14}$$

where $\text{mean}_A(\cdot)$ is the arithmetic mean, S_{CR} is the set of all successful crossover probabilities in the g-th generation. The aim of the adaptive mechanism is to record the recent successful parameters F_i and CR_i to guide the next generation of generating new good parameters. The fundamental principle of the adaptive scheme is that suitable parameters tend to generate better offspring individuals, and good individuals help to inherit their associated parameters.

3.2. Lifespan mechanism

The lifespan control for the leader is adaptively tuned according to the leader's leading power. Goldsmith [34] pointed out that a suitable lifespan was a significant and important characteristic for evolution because it could increase the diversity of the population. In this paper, the lifespan scheme mainly depends on its ability to challenge the best individual in the current population. More specifically, when the leader leads the current population to find a better individual than the best individual in the parent population, the lifespan of the leader θ increases one. If a better individual than the best in the parent population cannot be found in the offspring population, then the lifespan of the leader θ decreases one. Otherwise, the lifespan remains to be unchanged.

3.3. Challengers mechanism

When the lifespan of the old leader is exhausted, the current population may be trapped in a local optimum. Hence, it is necessary to add new information to the population to help the current population escaping from the local optimum. One of the two local search operators is randomly selected to generate a chal-

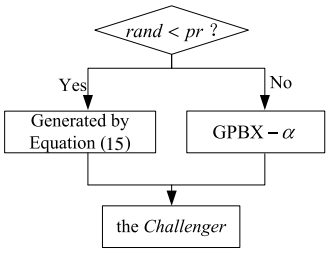

Fig. 3. Generation of the challenger.

lenger as shown in Fig. 3 to increase the diversity of the current population. Specifically, if a random number generated from the interval [0,1] is smaller than the value of parameter *pr*, a new challenger $\mathbf{x}_{\text{challenger}} = \left(x_{\text{challenger},1}, \cdots x_{\text{challenger},j}, \cdots, x_{\text{challenger},n}\right)$ is generated by:

$$x_{\text{challenger},j} = \begin{cases} x_j^{\min} + rand\,(0,1)\left(x_j^{\max} - x_j^{\min}\right), \text{if} rand < 1/n \\ x_{\text{leader},j}, \text{ otherwise} \end{cases} \quad (15)$$

Otherwise, another local search named Gaussian PBX-α (GPBX-α) [37] is utilized to generate a promising challenger. It should be noted that the old leader is regarded as the base vector of GPBX-α operator. The GPBX-α operator is a parent centric recombination operator which creates a new offspring individual from two parents. It aims to introduce more new information to increase the diversity of the current population.

These two local search operators generate the challenger around the region of the old leader. These techniques can remain the good information of the old leader and add new information to the current population to get rid of a local optimum, specifically for solving multimodal functions.

3.4. Criterion for accepting the new leader

The process of generating a new challenger is stochastic, thus its leading power should be checked. The challenger should be verified whether it has enough leading power to lead the current population. The challenger is utilized as a temporary leader for T generations. Specifically, the mutant strategy DE/current-to-leader/1 is replaced by DE/current-to-challenger/1 strategy to generate mutant individuals for running T generations, and then combined with the binominal crossover to generate offspring individuals. During the T iterations, if the fitness value of the best individual is improved, the challenger is regarded as the leader in the following generations. The age and lifespan of the new challenger is reset as 0 and θ_0, respectively. Otherwise, the old leader remains unchanged but its age is reset to $A = \theta - 1$. If there is still no improvement, another challenger is added into the current population to challenge the old leader again. Based on some preliminary study, the parameter T is set to 2 for ADE-ALC.

4. Numerical examples

Twenty-five benchmark function tests from IEEE CEC2005 are utilized to study the performance of the proposed ADE-ALC algorithm. A detailed description about these functions can be found in [38]. These test functions can be divided into four parts: (1) unimodal functions F_1—F_5, (2) basic multimodal functions F_6—F_{12}, (3) expanded multimodal functions F_{13}—F_{14}, (4) hybrid composition functions F_{15}—F_{25}. The number of the decision variables n is set to 30 for all functions. The termination criterion is 300 000 fitness function evaluations. Every function runs at 30 independent times to record its average and standard deviation of the function error value $(f(\mathbf{x}) - f(\mathbf{x}^*))$, where $\mathbf{x}$ is the best solution found by ADE-ALC approach and $\mathbf{x}^*$ is the known global optimum. To evaluate the statistical significance among the compared algorithms, a Wilcoxon's rank sum test at a 5% significance level is employed to judge whether the specific algorithm differs from the other contrasted algorithms.

4.1. Comparison with five DE algorithms

ADE-ALC algorithm is compared with five important DE variants, i.e. EPSDE [21], SaDE [17], jDE [39], CoBiDE [40] and SLADE [41]. In these algorithms, the maximum number of fitness function evaluations as the termination criterion is set to 300 000 and their relevant control parameters settings are the same as their original papers. The numerical simulation results are summarized in Table 1. "−", "+" and "≈" mean that the performance of the contracted algorithm is worse than, better than, and similar to that of ADE-ALC algorithm, respectively. "Mean Error" and "Std. Dev." mean that the average and standard deviation of the function error values obtained at 30 independent runs, respectively. The following subsections separately illustrate the performance of ADE-ALC by four different types of test functions.

4.1.1. Unimodal functions

From the results in Table 1, ADE-ALC algorithm is the best one among the six DE variant algorithms on these five unimodal functions. The good performance mainly depends on its greedy mutant strategy (DE/current-to-leader/1), which leads to the fast convergence rate. Furthermore, the convergence curves obtained by EPSDE, SaDE, jDE, CoBiDE, SLADE and ADE-ALC algorithms are shown in Fig. 4. Specifically, the convergence speed solved by ADE-ALC algorithm for functions F_2_F_5 achieves the fastest speed among the five DE algorithms.

4.1.2. Basic multimodal functions

For these seven multimodal functions from the results in Table 1, the statistical results of ADE-ALC algorithm is apparently better than results of EPSDE, SaDE and jDE on four, five and six test functions, respectively. The EPSDE and SaDE can only outperform ADE-ALC on two and two functions, respectively. The jDE cannot be significantly better than ADE-ALC on any functions. The CoBiDE and SLADE compared to ADE-ALC could obtain the similar performance for these seven tests. The computational results of ADE-ALC mainly depend on the aging mechanism which could balance the global and local search ability well. Moreover, the convergence curves derived by EPSDE, SaDE, jDE, CoBiDE, SLADE and ADE-ALC are plotted in Fig. 5. Specifically, the convergence curves of F_{10} obtained by ADE-ALC are the fastest among the six algorithms. For the functions F_9, the convergence curve obtained by ADE-ALC is faster than the curve obtained by EPSDE, SaDE and CoBiDE algorithms. For the F_{11} and F_{12}, the convergence curves obtained by ADE-ALC are faster than the curve obtained by EPSDE and jDE algorithms. The CoBiDE achieves good performance due to its ability to establish an appropriate coordinate system for crossover operator.

4.1.3. Expanded multimodal functions

For these two expanded multimodal functions, ADE-ALC is better than EPSDE, and wins jDE, SaDE, CoBiDE and SLADE on one function. Their convergence curves are plotted in Fig. 6. For the function F_{13}, the convergence curve obtained by ADE-ALC is similar with the curve obtained by jDE and EPSDE, but is faster than SaDE, CoBiDE and SLADE. For the function F_{14}, the ADE-ALC and SLADE achieves the similar convergence speed which are faster than the other four algorithms.

Table 1
Results of EPSDE, SaDE, jDE, CoBiDE, SLADE and ADE-ALC algorithms on 25 benchmark functions with 300 000 Fes.

fun.	EPSDE Mean Error ± Std. Dev.	SaDE Mean Error ± Std. Dev.	jDE Mean Error ± Std. Dev.	CoBiDE Mean Error ± Std. Dev.	SLADE Mean Error ± Std. Dev.	ADE-ALC Mean Error ± Std. Dev.
F_1	0.00E+0 ± 0.00E+0≈	0.00E+0 ± 0.00E+0≈	0.00E+0 ± 0.00E+0≈	0.00E+0 ± 0.00E+0≈	0.00E+0 ± 0.00E+0≈	0.00E+0 ± 0.00E+0
F_2	7.28E–26 ± 1.63E-25–	8.26E–6 ± 1.36E-5–	1.36E–6 ± 4.55E-6–	1.58E−14 ± 2.39E−14–	3.23E–27 ± 7.38E-27–	1.12E–28 ± 7.41E-29
F_3	9.59E+5 ± 4.81E+6–	4.72E+5 ± 2.10E+5–	1.81E+5 ± 8.47E+4–	7.46E+4 ± 3.53E+4–	6.81E+4 ± 4.59E+4–	8.89E+03 ± 7.01E+3
F_4	2.16E+1 ± 7.27E+1–	2.47E+2 ± 4.60E+2–	2.60E–2 ± 3.91E-1–	6.53E−4 ± 1.46E−3–	6.71E−3 ± 9.31E−3–	8.71E–16 ± 2.01E-15
F_5	1.34E+3 ± 6.62E+2–	3.45E+3 ± 5.76E+2–	5.02E+2 ± 4.01E+2–	1.34E+2 ± 1.70E+2–	1.15E+2 ± 1.28E+2–	3.31E–7 ± 1.51E-6
F_6	2.66E–1 ± 1.01E+0+	5.83E+1 ± 3.28E+1–	2.74E+1 ± 2.71E+1–	1.98E−1 ± 7.28E−1+	9.33E+0 ± 1.35E+1+	1.67E+1 ± 3.73E+1
F_7	1.87E–2 ± 1.52E-02–	1.67E–2 ± 1.45E-02–	2.28E–2 ± 5.23E-3–	2.09E−3 ± 7.56E−3≈	2.71E–2 ± 8.13E-3–	6.36E–3 ± 9.45E-3
F_8	2.09E+1 ± 4.91E-2–	2.09E+1 ± 4.57E-2–	2.09E+1 ± 5.14E-2–	2.08E+1 ± 4.08E−1–	2.08E+1 ± 2.08E−1≈	2.08E+1 ± 1.89E-1
F_9	0.00E+0 ± 0.00E+0≈	1.33E–1 ± 3.44E-1–	0.00E+0 ± 0.00E+0≈	0.00E+0 ± 0.00E+0≈	0.00E+0 ± 0.00E+0≈	0.00E+0 ± 0.00E+0
F_{10}	4.32E+1 ± 7.65E+0+	5.15E+1 ± 1.15E+1–	5.58E+1 ± 8.30E+0–	4.89E+1 ± 1.04E+1–	4.30E+1 ± 1.04E+1–	2.51E+1 ± 5.02E+0
F_{11}	3.40E+1 ± 3.81E+0–	1.75E+1 ± 2.50E+0+	2.79E+1 ± 1.54E+0–	7.80E+0 ± 3.10E+0+	1.13E+1 ± 3.76E+0+	2.63E+1 ± 1.15E+0
F_{12}	3.50E+4 ± 7.86E+3–	4.38E+3 ± 5.36E+3+	9.30E+3 ± 8.43E+3–	3.84E+3 ± 3.93E+3+	4.63E+3 ± 5.21E+3+	4.89E+3 ± 4.36E+3
F_{13}	1.94E+0 ± 2.32E-1–	3.92E+0 ± 4.06E-1–	1.69E+0 ± 1.85E-1–	1.91E+0 ± 6.14E-1–	1.68E+0 ±7.31E-1–	1.39E+0 ± 1.54E-1
F_{14}	1.35E+1 ± 2.75E-1–	1.26E+1 ± 2.71E-1≈	1.24E+1 ± 1.72E-1≈	1.24E+1 ± 3.57E−1≈	1.24E+1 ±3.31E−1≈	1.23E+1 ± 3.47E-1
F_{15}	4.24E+2 ± 4.41E+1–	3.79E+2 ± 6.93E+1≈	3.87E+2 ± 8.11E+1≈	4.12E+2 ± 4.79E+1–	3.79E+2 ± 2.70E+1≈	3.80E+2 ± 7.63E+1
F_{16}	1.53E+2 ± 1.39E+1–	8.51E+1 ± 6.89E+1+	7.95E+1 ± 2.89E+1+	7.41E+1 ± 3.65E+1+	9.11E+1 ± 4.10E+1+	1.03E+2 ± 1.56E+2
F_{17}	1.71E+2 ± 1.02E+2–	7.82E+1 ± 3.89E+1≈	1.43E+2 ± 3.79E+1–	7.73E+1 ± 1.98E+1≈	7.84E+1 ± 2.11E+1–	7.81E+1 ± 2.74E+1
F_{18}	8.20E+2 ± 4.52E+0+	8.68E+2 ± 6.34E+1+	9.04E+2 ± 1.07E+1–	9.03E+2 ± 1.01E+1≈	9.04E+2 ± 1.05E+0≈	9.04E+2 ± 9.05E-1
F_{19}	8.21E+2 ± 2.73E+0+	8.53E+2 ± 6.32E+1+	9.04E+2 ± 1.23E+0–	9.03E+2 ± 1.02E+1≈	9.04E+2 ± 1.16E+0≈	9.04E+2 ± 6.01E-1
F_{20}	8.21E+2 ± 4.35E+0+	8.91E+2 ± 5.93E+1+	9.04E+2 ± 1.09E+0≈	9.04E+2 ± 6.34E−1≈	9.04E+2 ±9.91E−1≈	9.04E+2 ± 1.17E+0
F_{21}	8.43E+2 ± 1.03E+2–	5.53E+2 ± 1.73E+2–	5.00E+2 ± 4.67E-13+	5.00E+2 ± 5.76E−13+	5.41E+2 ± 1.60E+2–	5.12E+2 ± 5.83E+1
F_{22}	5.05E+2 ± 8.32E+0+	9.42E+2 ± 1.91E+1–	8.83E+2 ± 1.78E+1≈	8.71E+2 ± 2.87E+1–	9.01E+2 ± 1.89E+1–	8.67E+2 ± 1.95E+1
F_{23}	8.49E+2 ± 6.81E+1–	5.32E+2 ± 3.90E-3≈	5.32E+2 ± 2.91E-4≈	5.34E+2 ± 1.52E−4–	5.32E+2 ± 5.71E−9≈	5.32E+2 ± 3.75E-13
F_{24}	2.13E+2 ± 1.54E+0–	2.00E+2 ± 6.18E-13≈	2.00E+2 ± 5.67E-14≈	2.00E+2 ± 2.93E−14≈	2.00E+2 ± 2.96E−14≈	2.00E+2 ± 2.89E-14
F_{25}	2.13E+2 ± 2.65E+0–	2.14E+2 ± 2.00E+0–	2.11E+2 ± 9.31E-1–	2.10E+2 ± 7.68E−1≈	2.13E+2 ± 2.17E+0–	2.10E+2 ± 7.99E-1
–	**17**	**13**	**15**	**10**	**11**	
+	**6**	**6**	**2**	**5**	**4**	
≈	**2**	**6**	**8**	**10**	**10**	

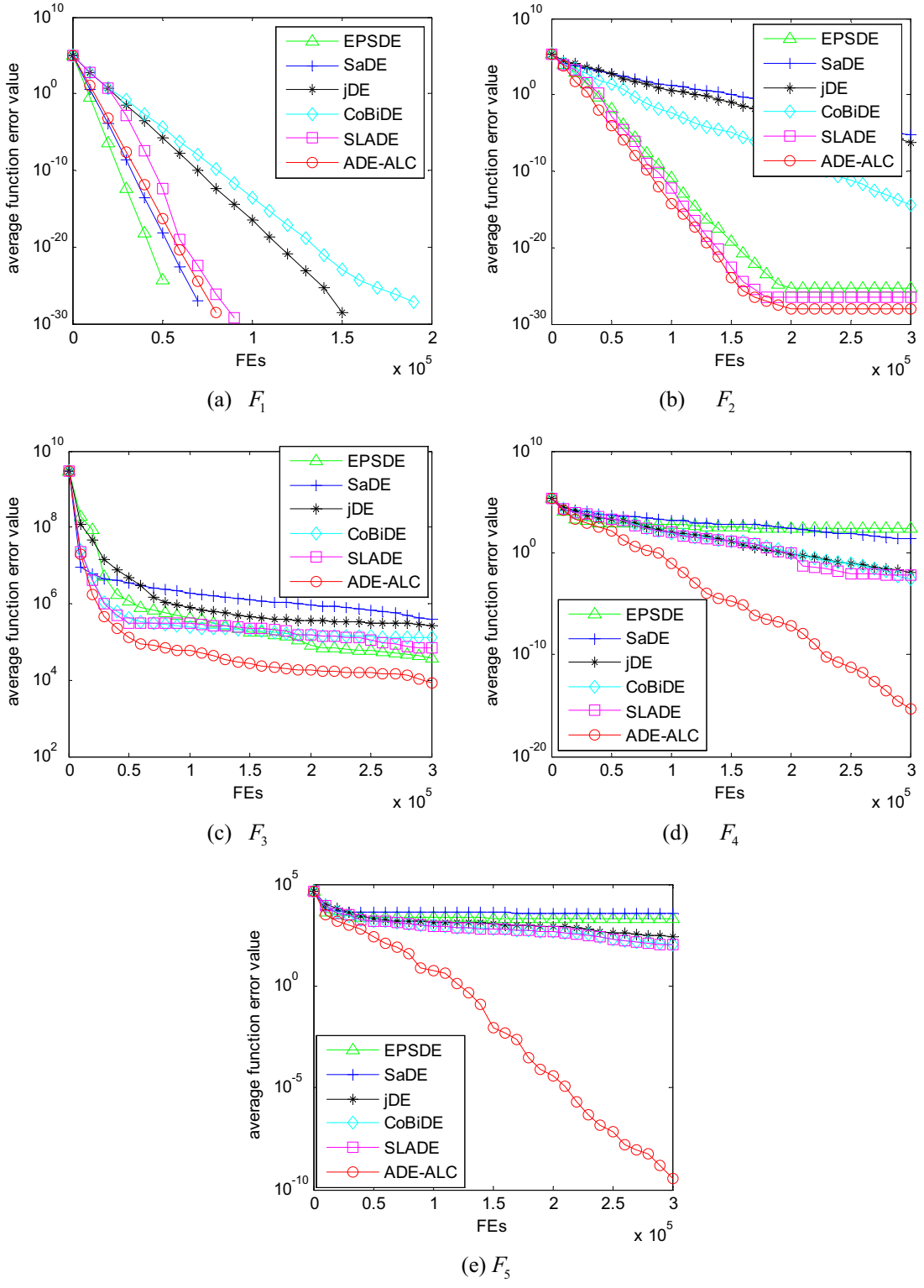

Fig. 4. Error values derived from EPSDE, SaDE, jDE, CoBiDE, SLADE and ADE-ALC versus FEs on five unimodal functions.

4.1.4. Hybrid composition functions

For the hybrid composition functions, the ADE-ALC outperforms EPSDE, SaDE, jDE, CoBiDE and SLADE on seven, three, four, three and four test functions, respectively. It ties SaDE, jDE CoBiDE and SLADE on four, five, six and six test functions, respectively. Some typical convergence curves are plotted as shown in Fig. 7. For the function F_{16}, the convergence curve achieved by ADE-ALC is the fastest among the five DE algorithms. For the function F_{21}, the convergence curve obtained by ADE-ALC is similar with the curve obtained by jDE, CoBiDE and SaDE, but is faster than EPSDE.

The last three rows of Table 1 summarize the numerical simulation results. Overall, the ADE-ALC achieves the better performance

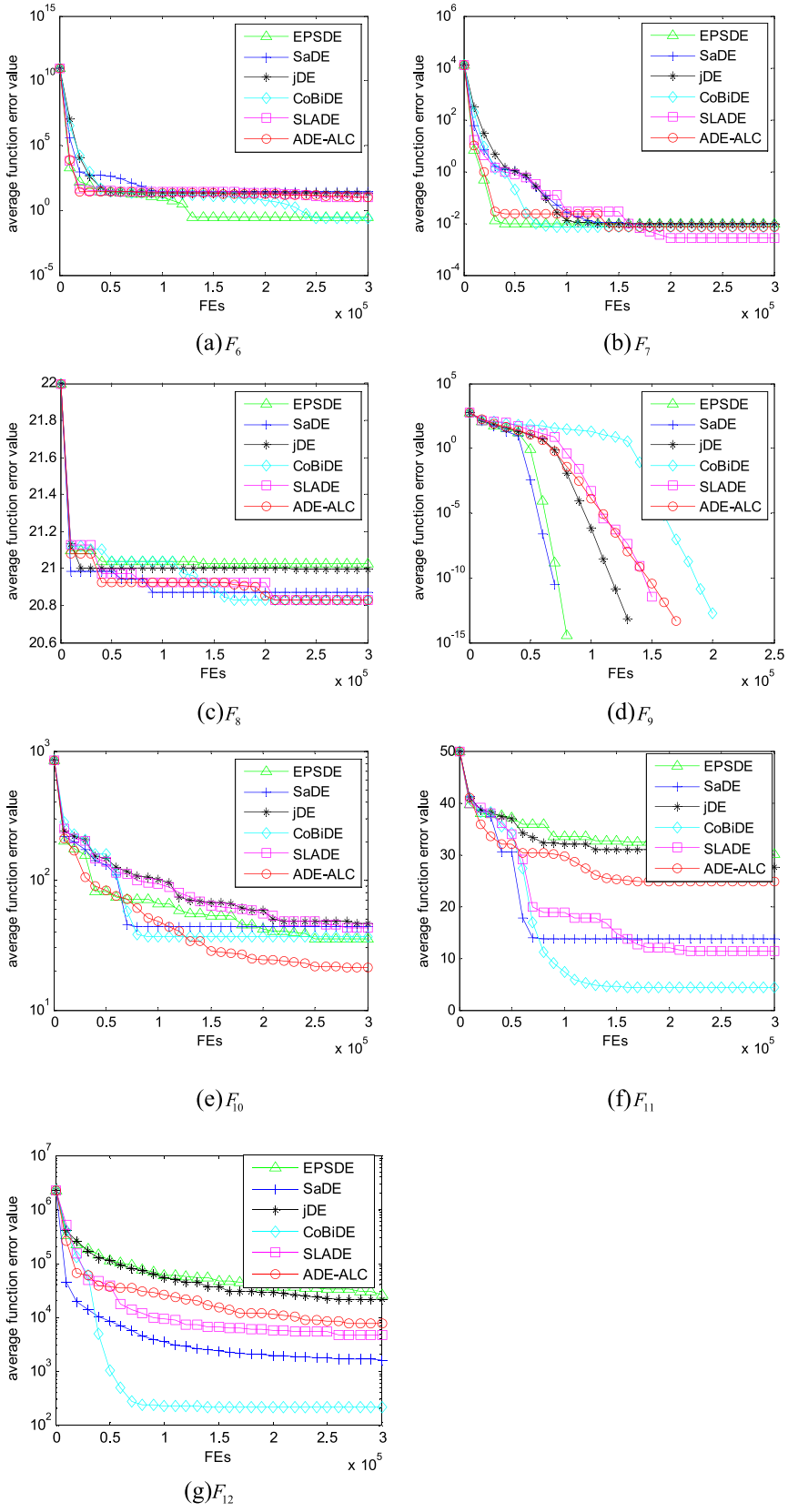

Fig. 5. Convergence curves derived by EPSDE, SaDE, jDE, CoBiDE, SLADE and ADE- ALC for seven basic multimodal functions(g)F_{12}.

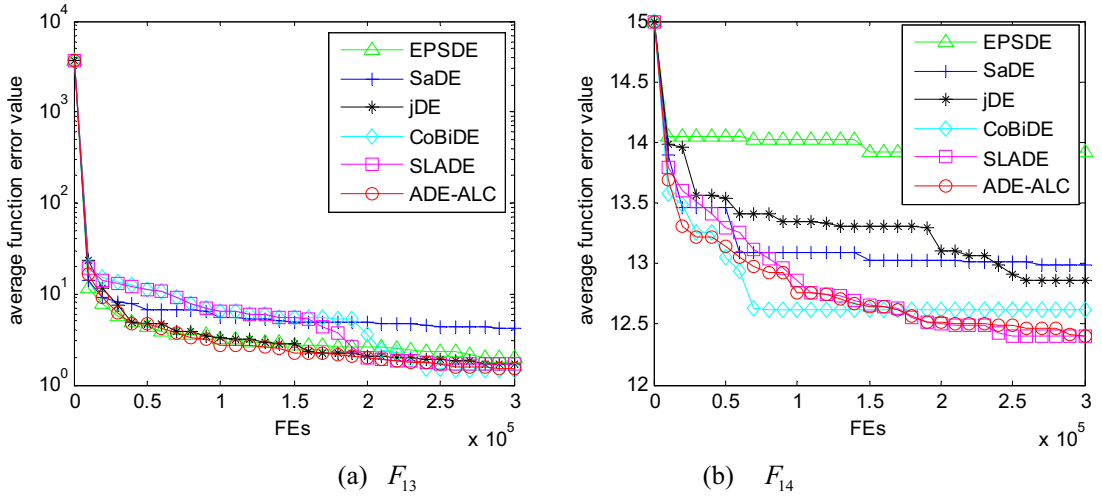

Fig. 6. Convergence curves derived by EPSDE, SaDE, jDE, CoBiDE, SLADE and ADE- ALC for F_{13} and F_{14}.

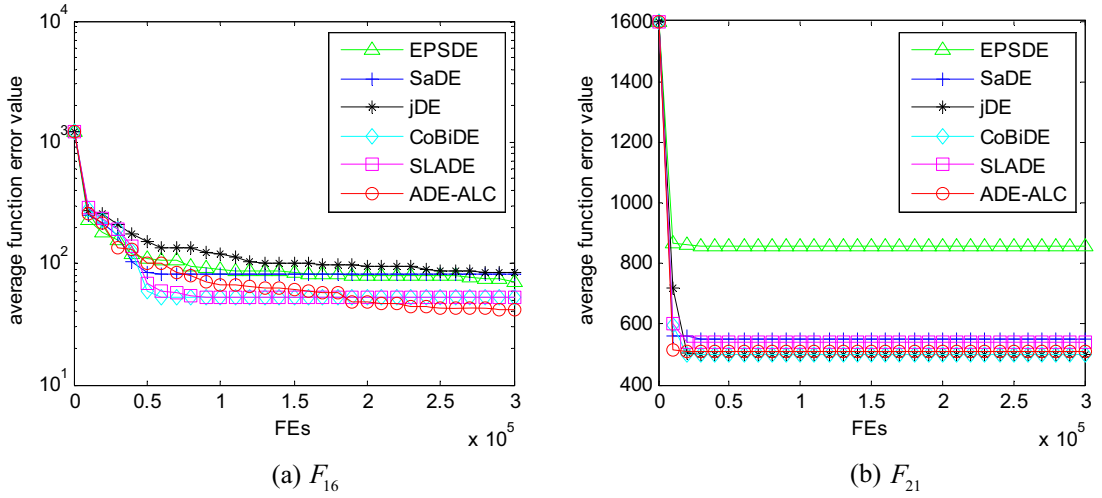

Fig. 7. Convergence curves derived by EPSDE, SaDE, jDE, CoBiDE, SLADE and ADE- ALC for $F_{16}$and F_{21}.

than EPSDE, SaDE, jDE, CoBiDE and SLADE. It wins EPSDE, SaDE, jDE, CoBiDE and SLADE on 17, 13, 15, 10 and 11 functions, respectively. And it ties the compared algorithms on 2, 6, 8 10 and 10 test functions, respectively. From Figs. 4–7, the convergence speed of ADE-ALC is the fastest at the most test functions, specifically for the unimodal and basic multimodal functions.

4.2. Comparison with three non-DE algorithms

In order to further verify the performance of ADE-ALC, it is compared with three non-DE variant approaches, i.e. CLPSO, CMA-ES and GL-25. CLPSO algorithm [42] is one of the typical PSO. It utilized all other particles' historical best information to update a particle's velocity and had the strong ability to solve multimodal functions. CMA-ES approach [43] was a typical efficient evolution strategy with the covariance matrix adaptation technique to solve optimization problems. GL-25 approach [44] was an efficient and important hybrid real-coding genetic algorithm with parent-centric crossover operators. Their relevant setting parameters are same as the original papers. The comparison results about the four algorithms are summarized in Table 2.

The last three rows in Table 2 summarize the numerical simulation results. Overall, ADE-ALC achieves better performance than CLPSO, CMA-ES and GL-25. Specifically, it wins CLPSO, CMA-ES and GL-25 on 15, 14 and 21 functions, respectively. It ties the compared algorithms on 5, and 2 test functions, respectively.

4.3. Comparison with ALC-PSO

The concept of aging mechanism utilized in ADE-ALC is similar to that of ALC-PSO proposed by Chen et al. [33]. But their differences are as follows: (1) ALC-PSO has introduced the aging mechanism to the framework of PSO. ADE-ALC introduces the concept of aging mechanism to DE, but employs a different way to control the lifespan of leader. The idea is partially inspired by genetic algorithm with aging structure proposed by Kubota et al. [31] and Ghosh et al. [45]; (2) the lifespan of the leader in ALC-PSO is adaptively adjusted by the three unique types of non-ascending sequences. In ADE-ALC, the lifespan of leader is adjusted according to its ability to change the best solution in the current evolutionary population; (3) we randomly utilize one of two local methods to generate challenger which could be better to keep the diversity of the evolutionary pop-

Table 2
Results of CLPSO, CMA-ES, GL-25 and ADE-ALC algorithms on 25 benchmark functions with 300 000 Fes.

prob.	CLPSO Mean Error ± Std. Dev.	CMA-ES Mean Error ± Std. Dev.	GL-25 Mean Error ± Std. Dev.	ADE-ALC Mean Error ± Std. Dev.
F_1	0.00E+0 ± 0.00E+0≈	2.63E–25 ± 1.75E-26–	6.11E–27 ± 2.76E-26–	0.00E+0 ± 0.00E+0
F_2	8.41E+2 ± 1.83E+2–	1.51E–24 ± 4.91E-25–	4.04E+1 ± 6.41E+1–	1.12E–28 ± 7.41E-29
F_3	1.51E+7 ± 3.11E+6–	4.79E–21 ± 2.10E-21+	2.19E+6 ± 1.08E+6–	8.89E+03 ± 7.01E+3
F_4	6.98E+3 ± 1.71E+3–	9.14E+5 ± 2.22E+6–	9.07E+2 ± 4.16E+2–	8.71E–16 ± 2.01E-15
F_5	3.88E+3 ± 4.43E+2–	2.78E–10 ± 5.14E-11+	2.51E+3 ± 1.96E+2–	3.31E–7 ± 1.51E-6
F_6	4.19E+0 ± 3.43E+0+	4.77E–1 ± 1.28E+0+	2.15E+1 ± 1.18E+0–	1.67E+1 ± 3.73E+1
F_7	4.89E–1 ± 1.10E-3–	6.84E–3 ± 4.54E-3≈	2.83E–2 ± 3.43E-2–	6.36E–3 ± 9.45E-3
F_8	2.09E+1 ± 4.51E-2≈	2.07E+1 ± 6.73E-1≈	2.09E+1 ± 5.60E-2≈	2.08E+1 ± 1.89E-1
F_9	0.00E+0 ± 0.00E+0≈	4.47E+2 ± 7.20E+1–	2.45E+1 ± 7.31E+0–	0.00E+0 ± 0.00E+0
F_{10}	1.01E+2 ± 1.62E+1–	4.66E+1 ± 1.21E+1–	1.42E+2 ± 6.45E+1–	2.51E+1 ± 5.02E+0
F_{11}	2.62E+1 ± 1.59E+0≈	7.16E+0 ± 2.21E+0+	3.27E+1 ± 7.78E+0–	2.63E+1 ± 1.15E+0
F_{12}	1.79E+4 ± 5.38E+3–	1.26E+4 ± 1.74E+4–	6.53E+4 ± 4.69E+4–	4.89E+3 ± 4.36E+3
F_{13}	2.06E+0 ± 2.41E-1–	3.43E+0 ± 7.59E-1–	6.23E+0 ± 4.89E+0–	1.39E+0 ± 1.54E-1
F_{14}	1.28E+1 ± 2.48E-1–	1.47E+1 ± 3.28E-1–	1.31E+1 ± 1.75E-1–	1.23E+1 ± 3.47E-1
F_{15}	5.78E+1 ± 2.84E+1+	5.55E+2 ± 3.40E+2–	3.04E+2 ± 2.01E+1+	3.80E+2 ± 7.63E+1
F_{16}	1.69E+2 ± 2.74E+1–	2.98E+2 ± 2.10E+2–	1.32E+2 ± 6.93E+1–	1.03E+2 ± 1.56E+2
F_{17}	2.45E+2 ± 4.77E+1–	4.43E+2 ± 3.31E+2–	1.61E+2 ± 6.74E+1–	7.81E+1 ± 2.74E+1
F_{18}	9.13E+2 ± 1.46E+0–	9.04E+2 ± 2.99E-1≈	9.07E+2 ± 1.41E+0–	9.04E+2 ± 9.05E-1
F_{19}	9.14E+2 ± 1.48E+0–	9.16E+2 ± 6.11E+1–	9.06E+2 ± 1.23E+0–	9.04E+2 ± 6.01E-1
F_{20}	9.14E+2 ± 3.54E+0–	9.04E+2 ± 2.68E-1≈	9.07E+2 ± 1.35E+0–	9.04E+2 ± 1.17E+0
F_{21}	5.00E+2 ± 4.71E-13+	5.00E+2 ± 3.91E-12+	5.00E+2 ± 5.18E-13+	5.12E+2 ± 5.83E+1
F_{22}	9.71E+2 ± 1.27E+1–	8.31E+2 ± 1.43E+1+	9.28E+2 ± 7.05E+1–	8.67E+2 ± 1.95E+1
F_{23}	5.34E+2 ± 2.21E-4≈	5.36E+2 ± 5.28E+0–	5.34E+2 ± 4.72E-4–	5.32E+2 ± 3.75E-13
F_{24}	2.00E+2 ± 5.71E-12≈	2.12E+2 ± 5.92E+1–	2.00E+2 ± 5.81E-11≈	2.00E+2 ± 2.89E-14
F_{25}	2.00E+2 ± 1.96E+0+	2.07E+2 ± 6.13E+0≈	2.17E+2 ± 1.36E-1–	2.10E+2 ± 7.99E-1
–	**15**	**14**	**21**	
+	**4**	**6**	**2**	
≈	**6**	**5**	**2**	

Table 3
Results of ALC-PSO and ADE-ALC algorithms on 12 benchmark functions with 200 000 FEs.

prob.	ALC-PSO Mean Error ± Std. Dev.	ADE-ALC Mean Error ± Std. Dev.
f_1	1.68E−161 ± 8.21E-161−	2.47E−162 ± 1.91E-161
f_2	1.16E−90 ± 4.15E-90−	7.48E−91 ± 2.43E-90
f_3	1.79E−11 ± 3.54E-11−	1.29E−26 ± 4.57E-26
f_4	7.61E ± 6.66+	8.37 ± 7.60
f_5	0 ± 0≈	0 ± 0
f_6	21.03 ± 54.10−	14.62 ± 35.38
f_7	2.53E−14 ± 1.38E-14+	6.41E−13 ± 1.86E-13
f_8	1.25E−11 ± 6.75E-11+	7.77E−10 ± 6.42E-13
f_9	1.15E−14 ± 2.94E-15−	7.17E−15 ± 5.82E-13
f_{10}	1.22E−2 ± 1.56E-2−	4.38E−4 ± 5.22E-4
f_{11}	4.39E−32 ± 7.60E-32−	1.19E−32 ± 3.21E-32
f_{12}	3.17E−31 ± 6.92E-31−	1.47E−32 ± 4.19E-32
−	8	
+	3	
≈	1	

Table 4
The design variables of the cab and their relevant parts.

design variables	Part name	Search space
x_1	Thickness of the floor stringers	[1.2,2.5]
x_2	Thickness of the front pillar outer panel	[1.0,2.2]
x_3	Thickness of the floor	[0.6,1.8]
x_4	Thickness of the door inner panel	[0.6,1.8]
x_5	Thickness of the front wall inner panel	[1.0,2.2]

ulation. In ALC-PSO, only one local method is used to generate new challenger which may occur premature convergence.

The numerical results of ALC-PSO and ADE-ALC on 12 benchmark functions selected from the literature of ALC-PSO are shown in Table 3. The functions f_1-f_5 are unimodal and f_6-f_{12} are multimodal with many local optima. These functions are solved by the ALC-PSO and ADE-ALC and the maximum FEs is set as 200 000. As a result, the ADE-ALC achieves a better performance than ALC-PSO on 8 functions, and it ties on 1 function. From these functions, we can conclude that the performance of ADE-ALC is better than ALC-PSO under this condition.

4.4. Analysis of parameters for ADE-ALC

4.4.1. The lifespan θ

The effect of lifespan mechanism is studied by two typical benchmark functions: the unimodal function F_3 and multimodal function F_6. Computational results about the search behavior of the ADE-ALC on these two functions are plotted in Fig. 8 and 9, respectively. From Fig. 8, the ADE-ALC remains a fast convergence speed for unimodal function because the greedy mutant strategy "DE/current-to-leader" utilizes the information of the best individual to generate new mutant individuals. And the leader has a strong ability to lead the population, thus the lifespan still keep increasing. In contrast, from Fig. 9, the lifespan of the leader is shortened because it is very easy to trap in a local optimum. A challenger emerges to replace the old leader and brings in diversity. Hence, the ADE-ALC has strong ability to jump out of a local optimum.

4.4.2. The parameter pr

In ADE-ALC, the parameter *pr* controls that which one of the two local search operator is selected to generate the challenger. Fig. 10 shows the function error values under different *pr*values. The best value of *pr* for unimodal function F_3 is 0.5, and the values for multimodal function F_6 can achieve good performance under interval [0.3, 0.6]. Overall, based on this observations, $pr = 0.5$ appears to achieve a good balance for ADE-ALC algorithm. Hence, the two local search operators have the same probability to generate the challenger.

4.4.3. The parameters μ_F and μ_{CR}

In the basic DE, parameters *F* and *CR* are required to tune for different optimization problems. In ADE-ALC algorithm, the two parameters are adaptively updated by the given normal and Cauchy distributions. The key parameters of the two distributions are mainly controlled by the μ_F and μ_{CR}. The evolution of μ_F and μ_{CR} for solving unimodal and multimodal functions are illustrated in Fig. 11. For unimodal functions F_1 and F_4, the mean values of μ_F and μ_{CR} are firstly tuned to the reasonable values and then change little for subsequent generations. For multimodal functions F_8 and F_9, the mean values of μ_F and μ_{CR} are updated to different values with the increasing generation. Different values of the parameters *F* and *CR* are required to optimize different multimodal functions.

5. Engineering application

The commercial vehicles are the hinges of traffic system and play a very important role in our national economic and daily life. The

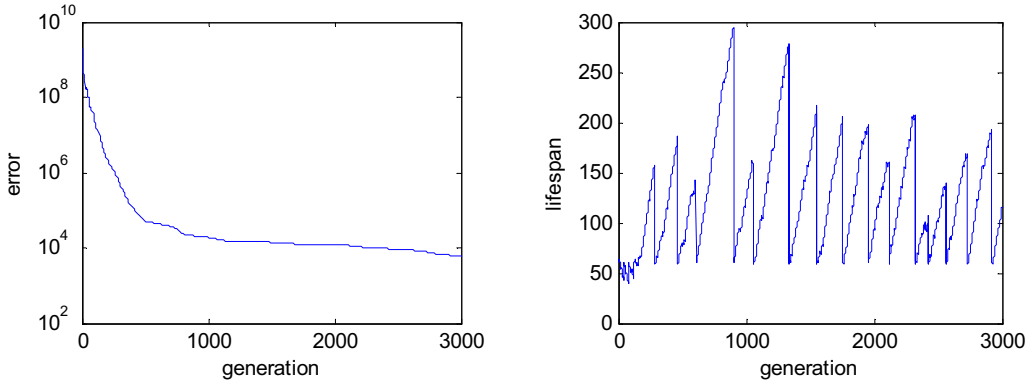

Fig. 8. Search behavior of the ADE-ALC on unimodal function F_3.

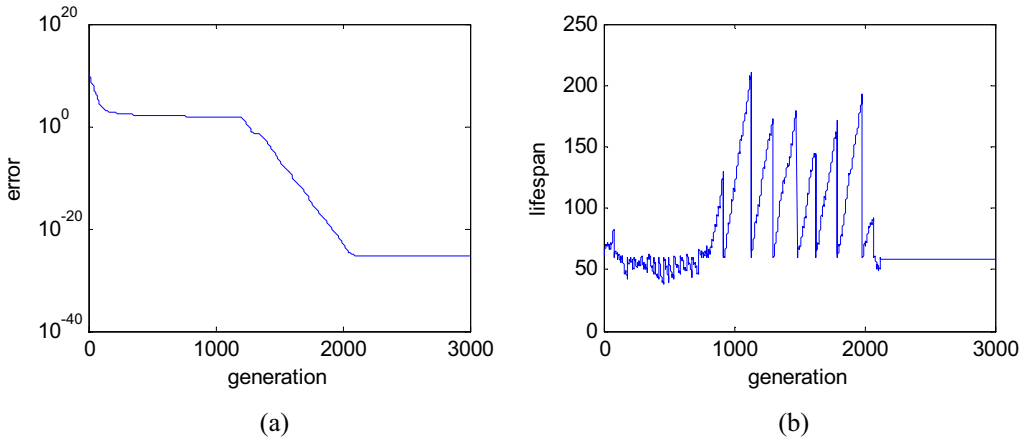

Fig. 9. Search behavior of the ADE-ALC on multimodal functionF_6.

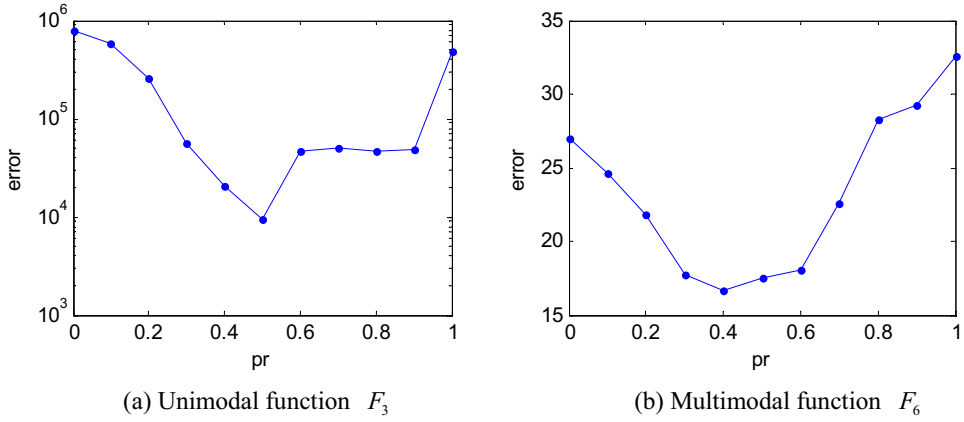

Fig. 10. Performance of the ADE-ALC under different *pr* values.

designers of the vehicles should try their best to reduce the mass of the commercial vehicle under the meeting shipping requirements. Moreover, the design of commercial vehicle cab should be capable of protecting the occupants' safety when the crash of the cab occurs inevitably. In order to achieve enough living space for the occupants [46], the horizontal distance between the steering column and the front seat L_1 should be greater than 110 mm, and horizontal distance between the steering column and the back seat L_2 should be greater than 230 mm what's more, the vertical distance between the steering wheel and the seat L_3 should be greater than 120 mm which could protect the thign of the driver well. So we construct an optimization problem which requires the maximum absorbing energy of cab $E(\mathbf{x})$ under the requirements of the living space. The converted optimization problem is defined as:

$$\begin{aligned} &\max E(\boldsymbol{x}) \\ &\text{s.t.} L_1(\boldsymbol{x}) \geq 110 \\ &L_2(\boldsymbol{x}) \geq 230 \\ &L_3(\boldsymbol{x}) \geq 120 \end{aligned} \tag{16}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_5]$ denotes the design variables of the commercial vehicle cab as shown in Fig. 12. The design variables of the cab and their relevant meaning are given in Table 4.

In the simulation of collision, the vehicle cab is fixed, and then the removable rigid barrier with 1500 kg simulating a pendulum offers an active weight to crash the fixed cab. The objective $E(\mathbf{x})$ denotes the absorbing energy of the selected parts when the vehicle cab collides. The optimization objective $E(\mathbf{x})$ and the constraints L_1, L_2 and L_3 are constructed by finite element model (FEM) of a typical commercial vehicle cab as shown in Fig. 12. The FEM model consists of 210 047 nodes, and 194 174 elements. The feasibility-based criterion proposed by Deb [47] is one of the most convenient and effective method to handle constraints. Hence, the proposed ADE-ALC with the feasibility-based criterion is utilized to solve this constrained optimization problem. The parameters are the same as these used to solve the numerical examples except that the parameters *NP* and *max_FEs* are reset as 30 and 2 000, respectively. As a result, the optimal solution is 1.87 mm, 2.2 mm, 1.8 mm, 1.05 mm, and 2.2 mm. Under this circumstance, the absorbing energy of the optimal vehicle cab is 5.2535×10^4 J which can absorb 301J more than the original design. Moreover, the results of L_1, L_2 and L_3 are 120.9 mm, 240.6 mm and 120.1 mm, respectively. Thus, this optimal design could improve the maximum absorbing energy meeting the requirements of the living space.

6. Conclusions

By introducing the aging mechanism into the framework of DE, a new adaptive differential evolution algorithm with an aging leader and challengers mechanism, namely ADE-ALC, is proposed to solve optimization problems. The control parameters are adaptively adjusted according to their cumulative successful experiences. The general performance of ADE-ALC is demonstrated by 25 benchmark test functions. It shows better or at least competitive optimiza-

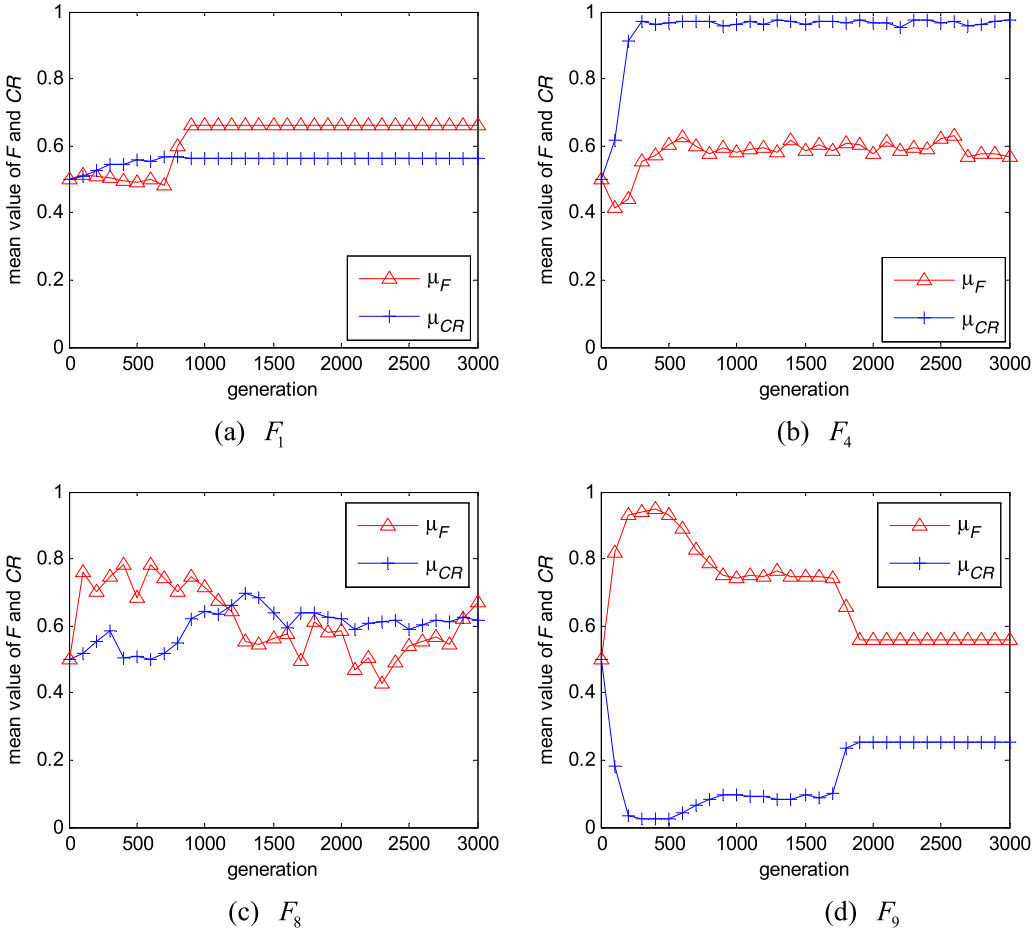

Fig. 11. Evolution of μ_F and μ_{CR} for F_1, F_4, F_8, and F_9.

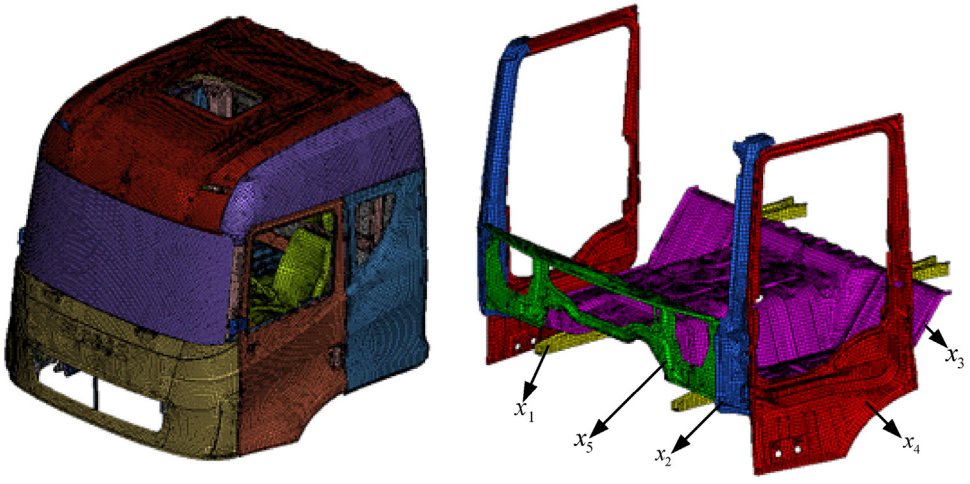

Fig. 12. The FEM of the commercial cab and the design variables.

tion performance in terms of the statistical performance, compared with some classic DE variants and three other significant evolutionary algorithms. Moreover, the convergence curves of ADE-ALC are faster than the ones of DE variants at most functions, especially for the unimodal and basic multimodal functions. For future work, the similar aging mechanism could be extended to solve the multi-objective optimization problem.

Acknowledgements

The authors would like to express sincere appreciations to the anonymous reviewers for their constructive and valuable suggestions, which are really helpful to guarantee the quality of this paper. This study is supported by the Major Program of National Natural Science Foundation of China (Grant No. 51490662), the Funds for Distinguished Young Scientists of Hunan Province (Grant No.

14JJ1016) and the Fok Ying-Tong Education Foundation of China (Grant No. 131005).

References

[1] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[2] F.S. Al-Anzi, A. Allahverdi, A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times, Eur. J. Oper. Res. 182 (2007) 80–94.
[3] S. Talatahari, A.H. Gandomi, X. Yang, S. Deb, Optimum design of frame structures using the eagle strategy with differential evolution, Eng. Struct. 91 (2015) 16–25.
[4] S. Kitayama, M. Arakawa, K. Yamazaki, Differential evolution as the global optimization technique and its application to structural optimization, Appl. Soft Comput. 11 (2011) 3792–3803.
[5] S. Sayah, A. Hamouda, A hybrid differential evolution algorithm based on particle swarm optimization for nonconvex economic dispatch problems, Appl. Soft Comput. 13 (2013) 1608–1619.
[6] S.M. Islam, S. Das, S. Ghosh, S. Roy, P.N. Suganthan, An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization, Systems, Man, and Cybernetics Part B: Cybernetics, IEEE Trans. 42 (2012) 482–500.
[7] W. Yu, M. Shen, W. Chen, Z. Zhan, Y. Gong, Y. Lin, O. Liu, J. Zhang, Differential evolution with two-level parameter adaptation, Cybernetics, IEEE Trans. 44 (2014) 1080–1099.
[8] W. Zhu, Y. Tang, J. Fang, W. Zhang, Adaptive population tuning scheme for differential evolution, Inf. Sci. 223 (2013) 164–191.
[9] A. Ghosh, S. Das, A. Chowdhury, R. Giri, An improved differential evolution algorithm with fitness-based adaptation of the control parameters, Inf. Sci. 181 (2011) 3749–3765.
[10] Q. Fan, X. Yan, Self-adaptive differential evolution algorithm with discrete mutation control parameters, Expert Syst. Appl. 42 (2015) 1551–1572.
[11] A. Draa, S. Bouzoubia, I. Boukhalfa, A sinusoidal differential evolution algorithm for numerical optimisation, Appl. Soft Comput. 27 (2015) 99–126.
[12] H. Salehinejad, S. Rahnamayan, H.R. Tizhoosh, Micro-differential evolution: diversity enhancement and a comparative study, Appl. Soft Comput. 52 (2017) 812–833, http://dx.doi.org/10.1016/j.asoc.2016.1009.1042.
[13] H. Fan, J. Lampinen, A trigonometric mutation operation to differential evolution, J. Global Optim. 27 (2003) 105–129.
[14] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (2009) 945–958.
[15] S. Das, A. Abraham, U.K. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Trans. Evol. Comput. 13 (2009) 526–553.
[16] Y. Cai, J. Wang, Differential evolution with neighborhood and direction information for numerical optimization, Cybernetics, IEEE Trans. 43 (2013) 2202–2215.
[17] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2009) 398–417.
[18] W. Gong, Á. Fialho, Z. Cai, H. Li, Adaptive strategy selection in differential evolution for numerical optimization: an empirical study, Inf. Sci. 181 (2011) 5364–5386.
[19] Q. Pan, P.N. Suganthan, L. Wang, L. Gao, R. Mallipeddi, A differential evolution algorithm with self-adapting strategy and control parameters, Comput. Oper. Res. 38 (2011) 394–408.
[20] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (2011) 55–66.
[21] R. Mallipeddi, P.N. Suganthan, Q.-K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2011) 1679–1696.
[22] R. Mukherjee, S. Debchoudhury, S. Das, Modified differential evolution with locality induced genetic operators for dynamic optimization, Eur. J. Oper. Res. 253 (2016) 337–355.
[23] B. Xin, J. Chen, J. Zhang, H. Fang, Z. Peng, Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: a review and taxonomy, Systems, Man, and Cybernetics Part C: Applications and Reviews, IEEE Trans. 42 (2012) 744–767.
[24] J. Sun, Q. Zhang, E.P. Tsang, DE/EDA: a new evolutionary algorithm for global optimization, Inf. Sci. 169 (2005) 249–262.
[25] I. Boussaïd, A. Chatterjee, P. Siarry, M. Ahmed-Nacer, Hybridizing biogeography-based optimization with differential evolution for optimal power allocation in wireless sensor networks, Vehicular Technology, IEEE Trans. 60 (2011) 2347–2353.
[26] X. Li, M. Yin, Parameter estimation for chaotic systems by hybrid differential evolution algorithm and artificial bee colony algorithm, Nonlinear Dynam. 77 (2014) 61–71.
[27] W.F. Gao, L.L. Huang, J. Wang, S.Y. Liu, C.D. Qin, Enhanced artificial bee colony algorithm through differential evolution, Appl. Soft Comput. 48 (2016) 137–150.
[28] M. Ghasemi, M.M. Ghanbarian, S. Ghavidel, S. Rahmani, E.M. Moghaddam, Modified teaching learning algorithm and double differential evolution algorithm for optimal reactive power dispatch problem: a comparative study, Inf. Sci. 278 (2014) 231–249.
[29] D. Jia, G. Zheng, M.K. Khan, An effective memetic differential evolution algorithm based on chaotic local search, Inf. Sci. 181 (2011) 3175–3187.
[30] M.G. Omran, A.P. Engelbrecht, A. Salman, Bare bones differential evolution, Eur. J. Oper. Res. 196 (2009) 128–139.
[31] N. Kubota, T. Fukuda, Genetic algorithms with age structure, Soft Comput. 1 (1997) 155–161.
[32] L.A. Gavrilov, N.S. Gavrilova, Evolutionary theories of aging and longevity, Sci. World J. 2 (2002) 339–356.
[33] W.-N. Chen, J. Zhang, Y. Lin, N. Chen, Z.-H. Zhan, H.S.-H. Chung, Y. Li, Y.-H. Shi, Particle swarm optimization with an aging leader and challengers, IEEE Trans. Evol. Comput. 17 (2013) 241–258.
[34] T. Goldsmith, The Evolution of Aging, Azinet Press, Crownsville, 2006.
[35] K. Price, R.M. Storn, J.A. Lampinen, Differential evolution: a practical approach to global optimization, Nat. Comput. 141 (2005) 1–24.
[36] E. Mezura-Montes, J. Velázquez-Reyes, C.A. Coello, A comparative study of differential evolution variants for global optimization, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, ACM, 2006, pp. 485–492.
[37] B. Dorronsoro, P. Bouvry, Improving classical and decentralized differential evolution with new mutation operator and population topologies, IEEE Trans. Evol. Comput. 15 (2011) 67–98.
[38] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, Nanyang Technological University Singapore and Kanpur Genetic Algorithms Laboratory, IIT, Kanpur, India, 2005, pp. 1–50.
[39] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (2006) 646–657.
[40] Y. Wang, H.X. Li, T. Huang, L. Li, Differential evolution based on covariance matrix learning and bimodal distribution parameter setting, Appl. Soft Comput. 18 (2014) 232–247.
[41] Z. Zhao, J. Yang, Z. Hu, H. Che, A differential evolution algorithm with self-adaptive strategy and control parameters based on symmetric Latin hypercube design for unconstrained optimization problems, Eur. J. Oper. Res. 250 (2016) 30–45.
[42] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (2006) 281–295.
[43] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evol. Comput. 9 (2001) 159–195.
[44] C. García-Martínez, M. Lozano, F. Herrera, D. Molina, A.M. Sánchez, Global and local real-coded genetic algorithms based on parent-centric crossover operators, Eur. J. Oper. Res. 185 (2008) 1088–1113.
[45] A. Ghosh, S. Tsutsui, H. Tanaka, Individual aging in genetic algorithms, in: Intelligent Information Systems, Australian and New Zealand Conference on, 1996 (1996) 276–279.
[46] W. Qi, X.L. Jin, X.Y. Zhang, Improvement of energy-absorbing structures of a commercial vehicle for crashworthiness using finite element method, Int. J. Adv. Manuf. Technol. 30 (2006) 1001–1009.
[47] K. Deb, An efficient constraint handling method for genetic algorithms, Comput. Methods Appl. Mech. Eng. 186 (2000) 311–338.