Soft Computing (2019) 23:3113–3128
https://doi.org/10.1007/s00500-017-2958-z

METHODOLOGIES AND APPLICATION

Self-adaptive parameters in differential evolution based on fitness performance with a perturbation strategy

Chen-Yang Cheng[1] · Shu-Fen Li[2] · Yu-Cheng Lin[1]

Published online: 1 December 2017

Abstract
Differential evolution (DE) algorithms have been used widely to solve optimization problems and practical cases and have demonstrated high efficiency, performing favorably using only a few parameters. Compared with other traditional algorithms, DE algorithms perform well when used to solve continuous problems. To obtain an approximate solution using DE, it is critical that appropriate parameter values are selected. However, selecting and dynamically tuning the parameter values during evolution are not easy tasks because the values depend significantly on the problem to be solved. To address these issues, this study presents an enhanced DE algorithm with self-adaptive adjustable parameters and a perturbation strategy based on individual fitness performance. Compared with two existing DE algorithms, the proposed algorithm can solve six benchmark functions and has both high efficiency and stability.

Keywords Self-adaptive parameters · Differential evolution · Perturbation strategy · Parameter adjusting · Fitness performance

1 Introduction

Evolutionary algorithms (EAs) are inspired by biological evolution. Proposed by Storn and Price (1997) and originally used to solve the Chebyshev polynomial problem, differential evolution (DE) algorithms are among the popular metaheuristic EAs. A population-based stochastic search technique, a DE algorithm is simpler to execute, requires fewer parameters and retains the more of the diversity of a population than other metaheuristic EAs. Furthermore, compared with other metaheuristic EAs, DE is more reliable and converges more quickly for optimization problems (Hu et al. 2013; Salman et al. 2007). DE has been applied in various areas (Storn and Price 1997), such as energy planning (Rajesh et al. 2016), scheduling (Tang et al. 2014), and supportive decision-making (Xue et al. 2009). In addition, it has been used to solve the traveling salesman problem (TSP) (Mi et al. 2010; Sauer and Coelho 2008) and other optimization problems (Iacca et al. 2012).

In classic DE, each new candidate solution is a combination of the parent individual and several other individuals in the population. A candidate that has better fitness than its parent replaces the parent. In addition, the search mechanism of a DE algorithm includes the mutation, crossover, and selection operations present in evolution. As with other EAs, a DE algorithm has only three crucial parameters: the scaling factor (F) of the difference vector, crossover rate (CR), and population size (NP) (Jia et al. 2011; Lee and Chiang 2011). DE does have some weaknesses, major among which are unstable convergence, slow convergence in the late stages of evolution, and a tendency to become stuck at a local optimal solution. Nonetheless, the use of suitable parameter values obviates these weaknesses such that a DE algorithm converges faster and yields superior solutions than when the parameters are not well selected. In addition, the properties of the parameters are highly correlated with the problems—that is, the objective functions—to be solved.

In the past few decades, researchers have improved many DE algorithms in three crucial ways: first, by modifying

Communicated by V. Loia.

✉ Chen-Yang Cheng
cycheng@ntut.edu.tw

Shu-Fen Li
fennieli@thu.edu.tw

Yu-Cheng Lin
t104378028@ntut.edu.tw

[1] Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei, Taiwan

[2] Department of Industrial Engineering and Enterprise Information, Tunghai University, Taichung, Taiwan

the procedure of implementing DE; next, by applying classic DE in combination with other search strategies such as the vector to be perturbed selection strategy, number of difference vectors considered for perturbation, and type of crossover being used (exponential or binomial) (Storn and Price 1997; Price et al. 2005), even with strategy adaptation (Qin et al. 2009); and finally, by proposing new parameters for the tuning mechanism (Das et al. 2005; Das and Suganthan 2011; Lee and Chiang 2011; Trivedi et al. 2015). First, with regard to the modified procedure for DE, Chuan-Kang and Chih-Hui (2009) proposed Poisson differential evolution using more than one difference vector based on a Poisson distribution, which significantly improved on DE in terms of solution quality and convergence speed. Second, a few researchers published studies focusing on DE in combination with other search strategies. Zhang and Sanderson (2007) implemented a strategy called "DE/current-to-p-best" that both improved the rate and reliability of convergence performance and employed scaling factors that had a distribution that was two-thirds the Gaussian distribution and one-third a uniform distribution. Wang et al. (2016) proposed a self-adaptive parameter dynamics DE algorithm and modified the algorithm's framework by implementing a new mutation strategy: DE/Current-to lbest/1. This modified framework algorithm maintains a balance between exploitation and exploration capabilities during DE. Zhang and Sanderson (2007) showed that the *CR* in DE obeys a normal distribution. Lee and Chiang (2011) proposed the use of a perturbation operator in classic DE to improve the DE algorithm's exploration and exploitation abilities, enabling individuals to more easily escape the local optimum. Lastly, numerous studies have verified that the values of both the mutation and crossover factors critically influence the performance of a DE algorithm (Chen and Chiang 2015). Mezura-Montes et al. (2006) empirically identified the DE algorithms that had highest performance for various global optimization problems. However, the researchers noted that parameter tuning is time-consuming. Brest et al. (2006) presented an improved DE algorithm with dynamic parameter values, and this algorithm randomly obtained values of the scaling factor and *CR* within the range of each parameter. Likewise, Dexuan and Liqun (2012) concluded that establishing suitable parameter values is crucial for efficiently solving complex problems. Therefore, they proposed a method that modifies the scaling factor, modifies the *CR*, and obeys a uniform distribution according to a linearly increasing strategy. Moreover, Li and Yin (2016) introduced self-adaptive parameter setting by using uniform random numbers that follow the Gaussian distribution; this enhanced the diversity based on the two new parameters proposed in a previous period. Chen and Chiang (2015) created a series of DE algorithms with adaptive parameters and summarized the six characteristics of a favorable adaptive scheme for controlling the parameters of DE algorithms: (1) adaptable to different problems, stages, and individuals; (2) revisable parameter values based on iterations or other factors; (3) considers population diversity to prevent premature convergence; (4) considers reusing successful parameter values; (5) applies a small perturbation to individuals with favorable fit; and (6) relies on a random distribution, in which case, both the mean/location and deviation/range are important.

As noted, adjusting or controlling parameters manually is time-consuming. Therefore, the purpose of the present paper is to propose an enhanced DE algorithm with self-adaptive adjustable parameters based on individual fitness performance (SADE-FP). The adjustment mechanism of SADE-FP relies on determining the mutation factor based on the fitness of individuals and generating crossover factors from the Gaussian distribution, as proposed by Lee and Chiang (2011). In addition, we compare the performance of SADE-FP with PsDE (Lee et al. 2011) and classic DE (Storn and Price 1997) for the solution of seven benchmark problems; the results of this comparison verify that the proposed SADE-FP performs well and is efficient. The rest of this paper is organized as follows. Section 2 reviews classic DE and recent adaptive DE variants. Section 3 describes the proposed SADE-FP algorithm. Section 4 presents the experimental investigation and a discussion of the results and their implications. Finally, our conclusions and suggestions for future research are presented in Sect. 5.

2 Related work

The classic DE algorithm is a population-based metaheuristic algorithm proposed by Storn and Price (1997). In this algorithm, the dimension vectors of an individual in a population represent a combination of decision variables as a reasonable solution of the objective function. An individual in the population of each generation is represented by a D-dimensional vector, expressed as $X_{i,G} = (X_{1,G}, X_{2,G}, \ldots, X_{i-1,G}, X_{i,G})$. A population consists of NP individuals: $i = 1, 2, \ldots, NP$. In classic DE, NP does not change during the evolution. G denotes one generation, and the maximum number of generations ($G_{\max}$) is the stopping criterion for the algorithm's search procedure.

The classic DE algorithm was proposed in 1996, and since then, many researchers have attempted to increase its effectiveness; specifically, to address its weaknesses, such as unstable convergence, slow convergence in late evolution stages, and tendency to become stuck in the local optimal solution (Jia et al. 2011; Lee and Chiang 2011). DE algorithms have improved in three aspects (Lee and Chiang 2011): by applying classic DE in concert with other strategies, by improving the control parameters (Brest et al. 2006; Liu and Lampinen 2005), and by forming a hybrid of classic

DE with other algorithms (Yildiz 2013; Ponsich and Coello 2013).

With regard to improving the control parameters, there are two ways in which parameters can be set: parameter tuning and parameter control (Eiben et al. 1999). In parameter tuning, a common approach, favorable parameter values are identified before running the algorithm, and the algorithm is then implemented using these values, which are not changed during the implementation. Parameter control, also commonly employed, is similar to parameter tuning but allows the parameter values to change during the implementation (Brest et al. 2007). Given that the parameters must be adjusted manually before it can be used, the classic DE algorithm employs parameter tuning as its strategy for parameter identification. Storn and Price (1997) considered the scaling factor to be most effective in the range $F = [0.5, 1.0]$; they noted that if parameter F is smaller than 0.4 or larger than 1, the algorithm performs more poorly. The crossover rate should be set within the range $CR = [0.9, 1.0]$ to ensure high convergence speed. To increase divergence and prevent arrival at the optimal solution prematurely, Zaharie (2007) found that when a binomial crossover strategy is used, F should be greater than 0.18 (under the conditions $NP = 50$ and $D = 30$). However, in recent research (De Falco et al. 2014; Jiang et al. 2013), the use of parameter control has been shown to improve the performance of DE. Parameter control can be divided into adaptive and self-adaptive parameter control. In self-adaptive parameter control, the processes of searching for optimal parameter values and optimal solutions are combined. By contrast, adaptive parameter control separates the optimal solution search process from the ideal parameter values search process. Unlike in self-adaptive parameter control, the mechanism for updating the ideal parameter values in adaptive parameter control is not part of the optimization cycle (Aleti and Moser 2016). Brest et al. (2006) proposed a self-adaptive parameter mechanism that generates F and CR randomly. When F is relatively small, each individual searches its neighboring area, and thus convergence occurs quickly. Omran et al. (2005) made F self-adaptive by randomly selecting the scaling factor from the current individuals and then multiplying the selected scaling factor by a number generated from the Gaussian distribution. This self-adaptive method, which is similar to the method whereby DE involves mutation, generates a nonfixed scaling factor and improves the quality of the solution compared with classic DE. Furthermore, Omran et al. (2005) commented that each individual can select from the target vector ($X_{i,G}$) and mutation vector ($V_{i,G}$) equally when the CR follows the normal distribution $N(0.5, 0.15)$. According to all these studies (Brest et al. 2006; Omran et al. 2005), each individual can have its own scaling factor and CR. Nonfixed and self-adaptive parameter setting avoids the time-consuming process of manual parameter tuning and allows for a high-quality solution to be obtained easily and with rapid convergence.

In the present study, we propose a method that relies on self-adaptive adjustment of parameters. We combine self-adaptive adjustable parameters with other strategies to avoid the time-consuming process of parameter setting, thus increasing the speed at which a stable solution is found. Specifically, our approach adjusts the scaling factor according to the procedure described by Brest et al. (2006) so that each individual has its own parameter set. Individuals in the search space can become more diverse and extend their search areas when F is large. When F is small, individuals focus on searching their neighboring areas and prioritize convergence. From this setting perspective, we propose the ideal scaling factor value, which should be small to obtain favorable solutions. Individuals with poor fitness should explore more, whereas individuals with favorable fitness should exploit more. Adjusting the concept of the crossover rate—referring to Omran et al. (2005) and Lee and Chiang (2011)—we employ a normal distribution. Each individual selects from the target vector ($X_{\mathrm{i},G}$) and mutation vector ($V_{i,G}$) equally when we use a normal distribution with an average equal to 0.5. We also use the perturbation strategy (Lee and Chiang 2011) before mutation to ensure that the algorithm does not become stuck at a local optimum. Owing to use of the self-adaptive adjustable parameters and a perturbation strategy, the proposed algorithm is efficient and effective at solving problems.

3 Proposed SADE-FP algorithm

We propose the self-adaptive adjustable parameters in DE based on fitness performance and the perturbation strategy (SADE-FP) algorithm by referring to the DE/rand/1/binomial crossover variant designed by Chiang et al. (2013). Figure 1 shows the example of perturbation strategy. In addition to proposing the self-adaptive mechanism of adjusting the scaling factor F, we propose a new model of the ideal scaling factor value, as shown in Fig. 2. According to individual fitness performance, we force the scaling factor to obey a cosine distribution. Furthermore, when the current fitness of an individual is poor, the value of their scaling factor is large in the next iteration. By contrast, when the current fitness of an individual is good, the value of their scaling factor is small in the next iteration. The special notation used to describe SADE-FP is as follows:

d:	number of dimensions of an individual ($d = 1, 2, 3, \ldots, D$)
$\lvert F_i\ (G)\rvert$:	ideal scaling factor (F) of the ith individual in the Gth generation
F_s:	ideal value of the scaling factor

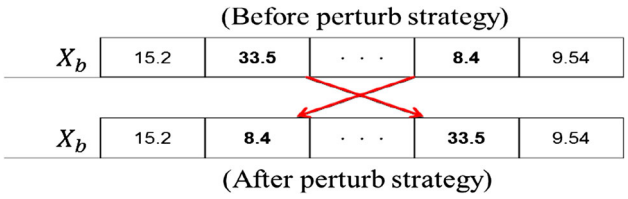

Fig. 1 Conceptual example of the perturbation strategy

F_b: lower bound of the ideal value of the scaling factor function
$A_i\ (G)$: current fitness of the ith individual in the Gth generation: $A_i\ = [0, 1]$
X_b: individual in the population with the highest fitness value
X_{w}: individual in the population with the lowest fitness value

The SADE-FP algorithm applies the perturbation strategy (Lee and Chiang 2011) to the current best individual in the population (X_b) before performing the classic DE search procedure, in which two dimensions of X_b are selected randomly and exchanged to improve a partially evolved problem. For example, as shown in Fig. 1, $X_b = (15.2, 33.5, \ldots, 8.4, 9.54)$ with D dimensions; the two dimensions randomly selected for perturbation are the 2nd and $(D-1)$th dimensions, the values of which are switched. The new current best individual in the population is then $X_b\ = (15.2, 33.5, \ldots, 8.4, 9.54)$. The objective of the perturbation in SADE-FP is for the current best individual to improve the local solution performance.

Furthermore, in line with Chiang et al. (2013) and Lin and Cheng (2015), we determine that the ideal scaling factor value is smaller in favorable solutions and larger in poorer solutions. An individual with poor fitness should explore more; conversely, an individual with favorable fitness should exploit more. Therefore, $F_i\ (G)$ is obtained based on the current fitness (A_i) of $X_{i,G}$. Here, $A_i\ (G)$ is the normalized value of the ith individual's fitness in the Gth generation, as given by Eq. (1). $f\left(X_{i,G}\right)$ is the nonnormalized fitness of individual i in generation G; $f\ (X_{\mathrm{w}})$ is the fitness of the worst individual in the population, whereas $f\ (X_b)$ is the fitness of the best individual. Subsequently, the ideal value of the scaling factor of the individual (F_i^{ideal}) is obtained by substituting $A_i\ (G)$ into Eq. (2):

$$A_i(G) = \frac{f(X_{\mathrm{w}}) - f(X_{i,G})}{f(X_{\mathrm{w}}) - f(X_b)} \tag{1}$$

$$F_i\ (G) = F_s \times \frac{1 + \cos\left(\pi * A_i\ (G)\right)}{2} + F_b. \tag{2}$$

The ideal scaling factor function is a nonlinear cosine-based function with the boundaries $[F_b, F_s]$ and is illustrated in Fig. 2. An individual with poor fitness has a larger scaling factor, whereas an individual with favorable fitness has a small scaling factor. Thus, the ideal scaling factor function is implemented according to the concept of a fitness performance-based search strategy. Given that the new individual scaling factor F is obtained before mutation is performed, it influences the mutation and selection operations of the new vector.

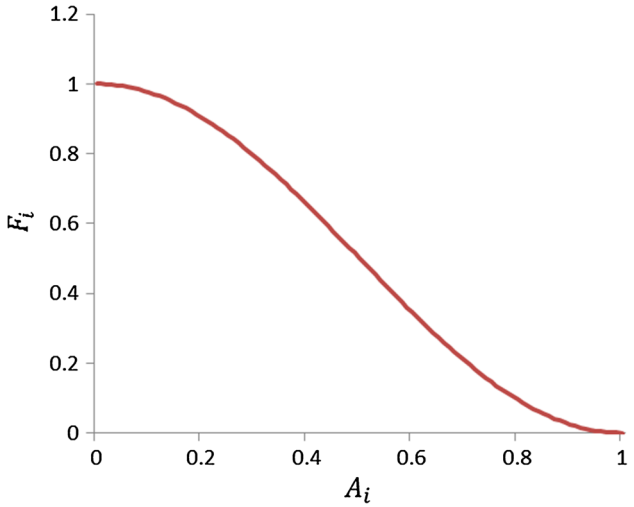

Fig. 2 Distribution of scaling factor (F) values

The procedure of SADE-FP is as follows:

Step 1 Initialization: randomly set the parameter values within the restrictions and initialize the objective function ($X_{i,G}$).

Step 2 Redefine X_b and X_{w}.

Step 3 Perform perturbation (Fig. 1).

Step 4 Calculate each individual's fitness and new scaling factor F [Eqs. (1) and (2)].

Step 5 Perform the mutation operation: randomly choose three individuals to generate the mutant vector ($V_{i,G+1}$):

$$V_{i,G+1} = X_{r1,G} + F \cdot \left(X_{r2,G} - X_{r3,G}\right),\ r1 \neq r2 \neq r3, \tag{3}$$

where $r1, r2, r3 \in \{1, 2, \ldots NP\}$ are random integers, each of which differs in accordance with index i. The scaling factor F is a real and constant factor $\in [0, 2]$ that controls the scale of difference between $X_{r2,G}$ and $X_{r3,G}$.

Step 6 Perform the crossover operation. Mix the target individual with the mutant vector to yield the trial solution ($U_{i,G+1}$). CR is set such that it obeys the Gaussian distribution (0.5, 0.1) [Eq. (4)] and is generated for every individual.

$$U_{i,j,G+1} = \begin{cases} V_{i,j,G+1}, & \text{if rand}_j \leq CR \\ X_{i,j,G}, & \text{otherwise} \end{cases}, \tag{4}$$

where j denotes the dimension of individual i in the Gth iteration. The larger the CR, the larger the movement of the

Table 1 Benchmark test functions employed

Function	Mathematical representation	Range	Dimension	$f(x^*)$
$f_1(x)$ Sphere	$\sum_{i=1}^{n} x_i^2$	$[-100, 100]$	30, 60	0
$f_2(x)$ Schwefel's problem 2.22	$\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]$	30, 60	0
$f_3(x)$ Step	$\sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	$[-100, 100]$	30, 60	0
$f_4(x)$ Rastrigin	$\sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12, 5.12]$	30, 60	0
$f_5(x)$ Ackley's	$-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)+20+e$	$[-32, 32]$	30, 60	0
$f_6(x)$ Griewank	$\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]$	30, 60	0
$f_7(x)$ Six-hump Camel-back	$4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$[-5, 5]$	2	-1.0316285

individual in one generation, where the movement of the individual is $\left(U_{i,G+1} - X_{i,G}\right)$.

Step 7 Compare the fitness of the target individual $(X_{i,G})$ and the trial solution $(U_{i,G+1})$ and retain the better of the two as the new individual in the next iteration $(X_{i,G+1})$:

$$X_{i,G+1} = \begin{cases} U_{i,G+1}, & if\ f\left(U_{i,G+1}\right) \le f\left(X_{i,G}\right) \\ X_{i,G}, & otherwise \end{cases}. \quad (5)$$

Step 8 When the iteration criterion (T) is met, stop; else, return to Step 2.

4 Experimental study and discussion

4.1 Experimental environment and benchmark test

To investigate the performance of various DE algorithms, 21 benchmark functions were used. The corresponding bounds of each dimension (searching space), optimal values of each dimension (x^*), and minimal fitness value of each function $[f(x^*)]$ are listed in Table 1 (Fan and Yan 2015; Lee et al. 2011). We verified the performance and accuracy of the proposed SADE-FP algorithm by using the six-hump camel-back function (f_7). The function operates in only two dimensions and has six local minima. A series of experiments was performed for the other six test functions, each of which had dimension sizes of 30 and 60 (Lee et al. 2011). All test functions were minimization problems, of which three were unimodal functions $(f_1{-}f_3)$ and four were multimodal functions $(f_4{-}f_7)$. The unimodal functions were used to verify the solution quality and convergence of the algorithm, whereas the multimodal functions, which usually have many local optima, were used to demonstrate that the algorithm does not become stuck at the local minima of the search space.

Two variants of the DE algorithm were used for comparison with the proposed SADE-FP algorithm: classic DE (Storn and Price 1997) and PsDE (Lee et al. 2011). The parameter values used for each of these algorithms are listed in Table 2.

Table 2 Parameter values used for the three DE algorithms

Parameters/Des	DE/rand1	PsDE	SADE-FP
F	0.5	$N(0.4, 0.15)$	Self-adaptive
CR	0.9	0.9	$N(0.5, 0.1)$
NP	50		

To increase the efficiency of the perturbation strategy, we set the population size to 50. Each algorithm was executed over 30 independent runs and 1000 iterations. The desired accuracy was achieved by performing dimensions × 1000 fitness evaluations (FEs), which indicate 30,000 function evaluations in the case of 30 dimensions and 60,000 function evaluations in the case of 60 dimensions. We performed the experiments on a 64-bit Windows 7 computer with an i5-2400 CPU and 8 GB of RAM.

4.2 Results and discussion

We obtained the fitness values of the approximate optima of the seven benchmark functions by using the three DE algorithms. Table 3 presents the averages and standard deviations (SDs) of the best fitness values of the three DE algorithms over 30 independent runs and 1000 iterations. In accordance with Suganthan et al. (2005), we designated the best solution obtained as the approximate optimum solution of each test function when the bias of the best solution and the optimum was smaller than the termination error of 10^{-8}. The best solution of each function solved by the three DE algorithms was considered the approximate optimal solution in accordance with the benchmark functions.

The results of the experiments are summarized in Table 3. The classic DE algorithm failed to solve five of the seven test functions, and PsDE failed to solve the Rastrigin function (f_4). However, the proposed SADE-FP algorithm solved all seven test functions. The Rastrigin function (f_4) is a nonlin-

Table 3 Means and SDs of the best fitness values obtained using the three DE algorithms over 30 independent runs with 1000 iterations and 50 individuals in the population

Function/mean(SD)	DE/rand1	PsDE	SADE-FP
Dimensions: 30			
$f_1(x)$	2.25E−11	6.26E−20	2.67E−29
	(2.02E−11)	(1.08E−19)	(2.83E−29)
$f_2(x)$	3.37E−06	1.21E−11	2.08E−21
	(1.95E−06)	(7.18E−12)	(1.11E−21)
$f_3(x)$	0	0	0
	(0)	(0)	(0)
$f_4(x)$	1.93E+02	4.82E+01	0
	(3.42+01)	(8.64E+01)	(0)
$f_5(x)$	1.26E−06	6.66E−11	8.02E−15
	(5.30E−07)	(5.00E−11)	(1.23E−15)
$f_6(x)$	3.03E−03	0	0
	(5.35E−03)	(0)	(0)
$f_7(x)$	− 1.031628	− 1.031628	− 1.031628
	(0)	(0)	(0)
Dimensions: 60			
$f_1(x)$	5.57E−03	3.51E−08	5.44E−17
	(7.57E−03)	(4.72−08)	(2.85E−17)
$f_2(x)$	2.78E−02	5.63E−06	4.54E−11
	(4.50E−02)	(2.87E−06)	(1.95E−11)
$f_3(x)$	9.23E+00	0	0
	(1.30E+01)	(0)	(0)
$f_4(x)$	4.69E+02	1.95E+02	9.48E+01
	(4.39E+01)	(3.10E+01)	2.39E+01
$f_5(x)$	1.14E+00	2.42E−05	2.57E−09
	(3.25E+00)	(9.02E−06)	(1.10E−09)
$f_6(x)$	1.24E−02	4.84E−03	6.77E−16
	(1.97E−02)	(1.11E−02)	(2.48E−15)
$f_7(x)$	− 1.031628	− 1.031628	− 1.031628
	(0)	(0)	(0)

ear multimodal function and more complicated than the other functions; only SADE-FP solved this function by obtaining the best solution below the termination error (10^{-8}).

We also determined the success rate (Chen and Chiang 2015) of the three DEs for each benchmark function (Table 4) as the ratio of the number of successful runs to the total number of runs. According to Suganthan et al. (2005), an algorithm can be considered successful at solving a function in a run if the deviation of the optimal fitness value is not greater than the termination error of 10^{-8}:

$$(\text{number of } f\left(X_{i,G_{\max}}\right) < 10^{-8})/NP, \qquad i = 1, 2, \ldots, NP \quad (6)$$

The performance of the three DE algorithms was further compared by visualizing their evolutionary processes using convergence plots. Some of the DE algorithms did not converge even after the execution of more than 1000 iterations. Therefore, we individually set the number of iterations of test functions f_1−f_6 to clearly obtain a convergence. The numbers of generations were set as follows: (f_1, 20,000), (f_2, 6000), (f_3, 1000), (f_4, 6000), (f_5, 3000), (f_5, 1000), and (f_6, 1000).

We analyzed the convergence graphs of each function for two dimension sizes: 30 and 60. Figure 3 displays the performance trends of the six test functions for 30 dimensions over 30 independent runs in the evolutionary process. Figure 3a and b plots the performance trends of the Sphere function (f_1) and Schwefel's problem 2.22 (f_2), that is, the number of convergence iterations (from few to many) when SADE-FP, PsDE, and classic DE were employed. The best fitness values obtained by all the algorithms decreased dramatically and reached the approximate optimal solution in fewer than 1000 iterations. In addition, the convergence speeds of all algorithms in case of the Step function (f_3) were higher (approximately 400 iterations) compared with those in the cases of f_1 and f_2, as illustrated in Fig. 3c. In the experiments using the three unimodal functions (f_1−f_3), SADE-FP converged considerably faster than the other two algorithms and all algorithms obtained approximately optimal solutions. In the experiments using the multimodal functions, we made the following observations. As a classic complex multimodal function based on the cosine function, the Rastrigin function (f_4) generates numerous local optima and searches tend to

Table 4 Success rates (SRs) of the three DE algorithms over 30 independent runs with 1000 iterations and 50 individuals in the population

Function/SR dimensions	DE/rand1 (30, 60)	PsDE (30, 60)	SADE-FP (30, 60)
f_1 (x)	1, 0	1, 0.93	1, 1
f_2 (x)	0, 0	1, 0	1, 1
f_3 (x)	1, 0	1, 1	1, 1
f_4 (x)	0, 0	0, 0	1, 0
f_5 (x)	0, 0	1, 0	1, 1
f_6 (x)	0.7, 0	1, 0.4	1, 1
f_7 (x)	1, 1	1, 1	1, 1
AVE	0.53, 0.14	0.86, 0.48	1, 0.86

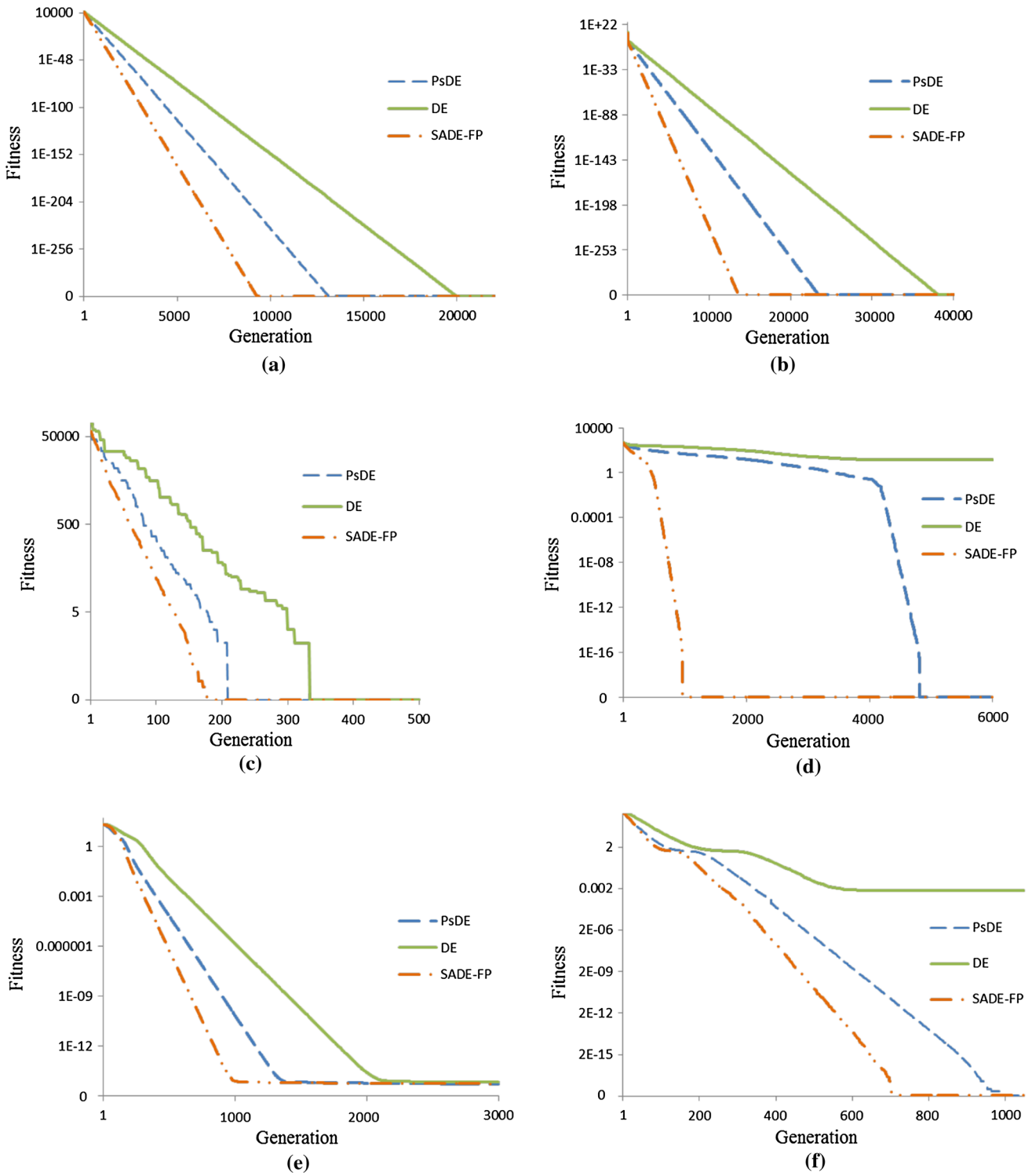

Fig. 3 Performance of the three DE algorithms in solving f_1–f_6 with $D = 30$: **a** f_1: Sphere function, **b** f_2: Schwefel's problem 2.22, **c** f_3: Step function, **d** f_4: Rastrigin function, **e** f_5: Ackley's function, and **f** f_6: Griewank function

become stuck at a local optimum. As shown in Fig. 3d, PsDE and SADE-FP evolved toward the optima and obtained an approximate solution. However, SADE-FP converged faster than PsDE. In addition, Fig. 3e and f reveals that both PsDE and SADE-FP efficiently obtained the approximate solution of Ackley's function (f_5) and the Griewank function (f_6), although classic DE could not obtain the approximate

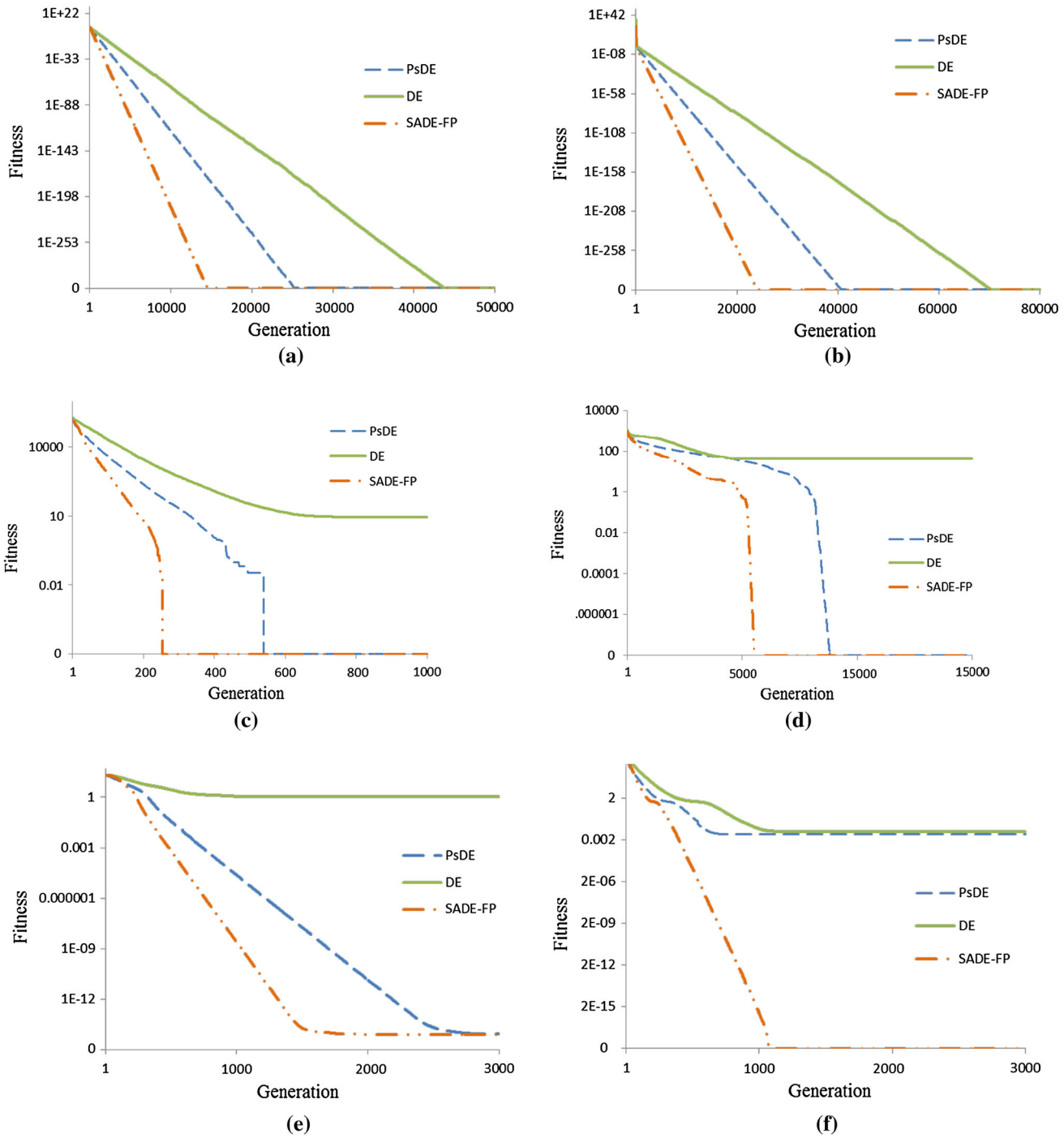

Fig. 4 Performance of the three ED algorithms in solving f_1–$f6$ with $D = 60$: **a** f_1: Sphere function, **b** f_2: Schwefel's problem 2.22, **c** f_3: Step function, **d** f_4: Rastrigin function, **e** f_5: Ackley's function, and **f** f_6: Griewank function

solution of either. Therefore, SADE-FP converges faster and obtains a better solution than the other two DE algorithms.

Figure 4 displays the convergence plots for the six test functions when 60 dimensions and 30 independent runs were employed in the evolutionary process. The differences between the three DEs are amplified compared with Fig. 3. In addition, in solving f_3–f_6, classic DE was more likely to become stuck at a local optimum than the other DE algorithms. SADE-FP also became stuck, failing to find the global optimum in its solution of f_1, f_2, f_4, and f_5. PsDE failed to solve only f_6. SADE-FP obtained favorable solutions and converged fast for most test functions—even the high-dimensional problems.

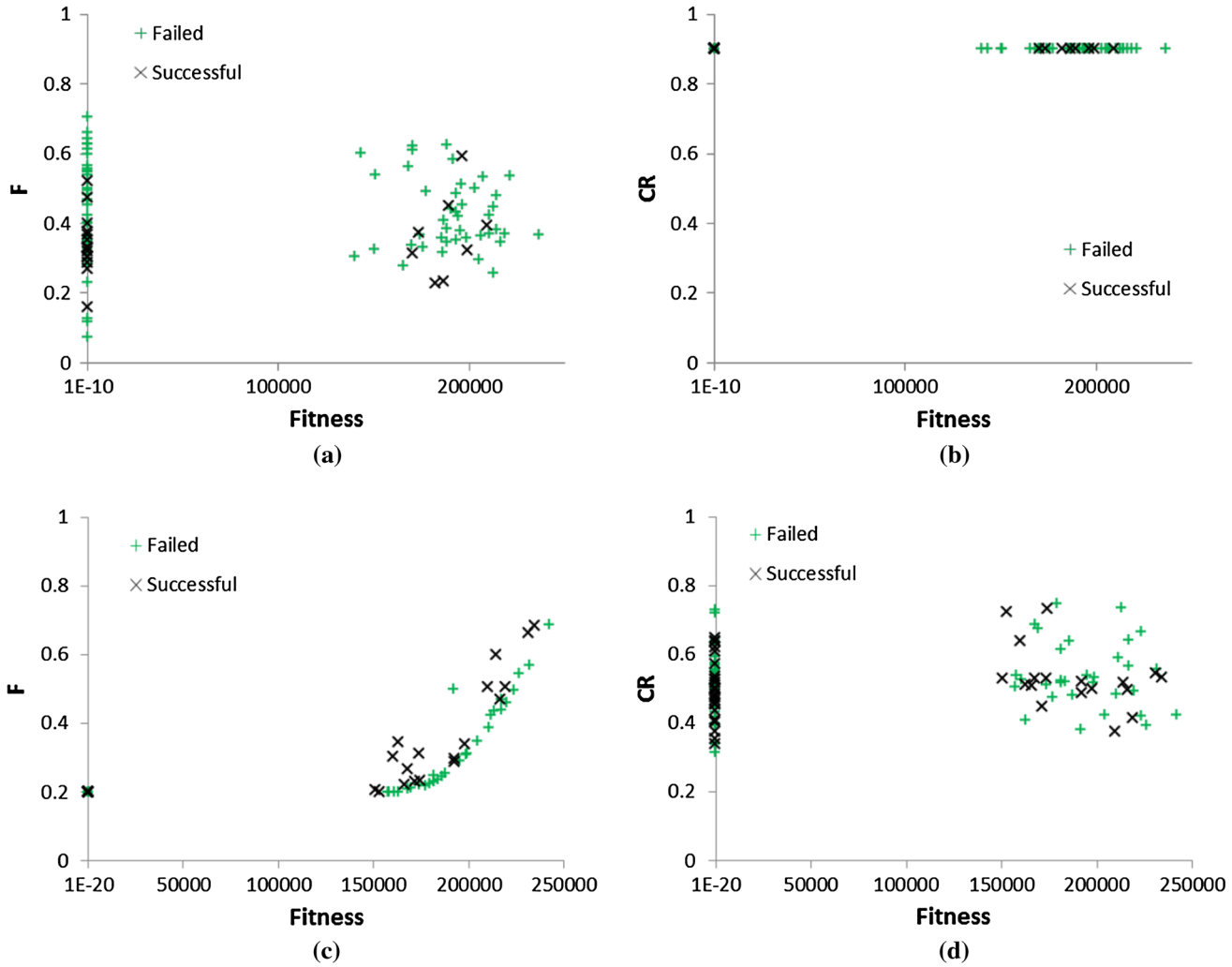

Fig. 5 F–fitness and CR–fitness plots for the Sphere function (f_1) with $D = 60$ and using 50,000 iterations: **a** F–fitness plot of PsDE, **b** CR–fitness plot of PsDE, **c** F–fitness plot of SADE-FP, and **d** CR–fitness plot of SADE-FP

Our overall purpose was to learn more about parameter values; therefore, we examined the relationship of F and CR values with fitness in the evolution process, as illustrated in Fig. 5. Brest et al. (2006) defined a parameter to be successful if using it in the mutation or crossover operator generates an offspring that is superior to its parent; then, they used F–fitness and CR–fitness plots to determine the extent to which fitness can be improved and the extent to which this improvement benefits the dynamic parameter values in the evolution process. In the present study, we took f_1 and f_4 with a high number of dimensions ($D = 60$), solved them using the DE algorithms with dynamic parameter values, and then drew F–fitness and CR–fitness plots, as presented in Figs. 5 and 6, respectively. From the two figures, we obtained the F and CR values of successful individuals.

First, we noticed that for all algorithms, the ranges of F and CR values of successful individuals in f_1 were wider than that in f_4. For example, the scope of successful F values determined using SADE-FP was (0.2, 0.8) in the case of f_1 and (0.1, 0.8) in the case of f_4 (Figs. 5c, 6c). Therefore, the scope of F was larger in solving multimodal functions in successful individuals with lower fitness values. Secondly, we discovered that a small F helped the DE algorithm avoid becoming stuck in a local optimum during the solution of f_4, as demonstrated by the plots for PsDE and SADE-FP. Next, we found that the CR values of successful individuals were lower when the fitness values were smaller, which may be why PsDE requires more iterations than SADE-FP to find the approximately optimal solution. Finally, the F plot for SADE-FP corresponds to setting the F value based on the cosine function. As Figs. 5c and 6c illustrate, the plot of the F value is similar in appearance to the sine function, which is the opposite of the shape of the cosine function, because a lower fitness value represents higher algorithm performance.

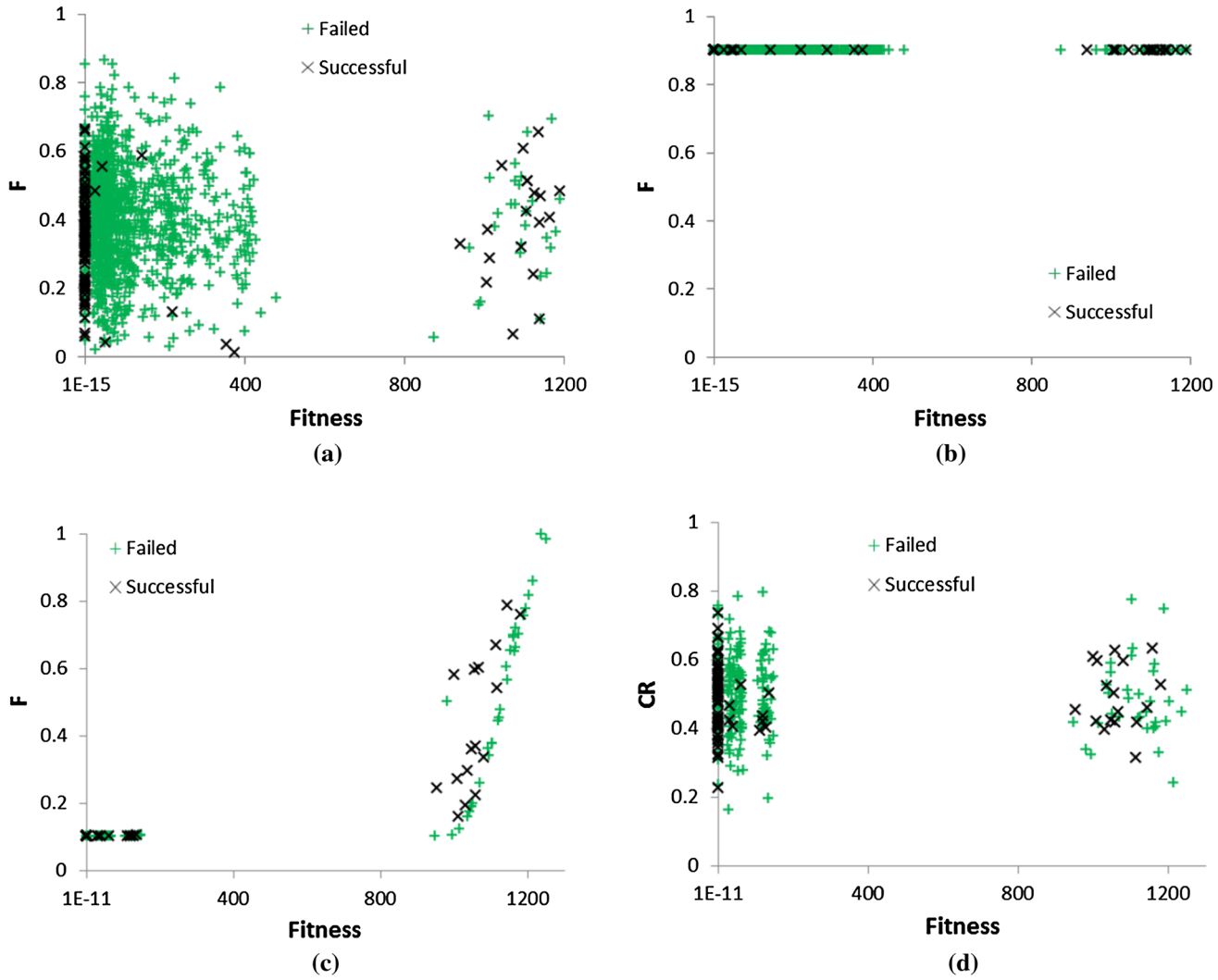

Fig. 6 F–fitness and CR–fitness plots of Rastrigin function ($f4$) with 60D over 50,000 iterations: **a** F–fitness plot of PsDE, **b** CR–fitness plot of PsDE, **c** F–fitness plot of SADE-FP, and **d** CR–fitness plot of SADE-FP

In this study, we compared the proposed algorithm with other algorithms by using the maximum FE stopping criterion. Fourteen functions and feasible ranges were referenced from CEC2005 (Suganthan et al. 2005) and are listed in Table 5. The minimum solution $f(x^*)$ of each function was referenced from Fan and Yan (2015). In this experiment, 14 test functions (f_8–f_{21}) with 30 dimensions were employed for comparing SADE-FP with four related algorithms—namely, DE/best/1, DE/rand/1, MDE_pBX (Islam et al. 2012), and DE-EPA (Hsieh et al. 2013)—as summarized in Table 6. The means and SDs obtained using DE/best/1, DE/rand/1, MDE_pBX, and DE-EPA in Table 6 were retrieved from Hsieh et al. (2013). The original parameter settings of these algorithms were used. The initial population size of SADE-FP was set to 100, and each algorithm was executed 25 times independently according to Hsieh et al. (2013). The maximum FE criterion was set to 300,000 (Suganthan et al. 2005).

The means and SDs of the errors of the five DE algorithms for solving the $D = 30$ problems are listed in Table 6. The best results among the five approaches are shown in bold. The proposed algorithm obtained significantly better results in solving functions 10, 12, 13, and 16 than the other algorithms. The proposed method also yielded the same best result as MDE_pBX and DE-EPA for function 8.

4.3 Comparison between proposed SADE-FP and classic DE

In this section, we compare classic DE and SADE-FP by performing statistical testing for advanced analysis. We employ the widely used nonparametric test method: the Wilcoxon

Table 5 Shifting of benchmark test functions

	Function	Range	Dimension	$f(x^*)$
Unimodal functions	$f_8(x)$: shifted Sphere function	$[-100, 100]$	30	0
	$f_9(x)$: shifted Schwefel's Problem 1.2	$[-100, 100]$	30	0
	$f_{10}(x)$: shifted rotated high conditioned elliptic function	$[-100, 100]$	30	0
	$f_{11}(x)$: shifted Schwefel's Problem 1.2 with noise in fitness	$[-100, 100]$	30	0
	$f_{12}(x)$: Schwefel's Problem 2.6 with global optimum on bounds	$[-100, 100]$	30	0
Multimodal functions	$f_{13}(x)$: shifted Rosenbrock's function	$[-100, 100]$	30	0
	$f_{14}(x)$: shifted rotated Griewank's function without bounds	$[-\infty, \infty]$	30	0
	$f_{15}(x)$: shifted rotated Ackley's function with global optimum on bounds	$[-32, 32]$	30	0
	$f_{16}(x)$: shifted Rastrigin's function	$[-5, 5]$	30	0
	$f_{17}(x)$: shifted rotated Rastrigin's function	$[-5, 5]$	30	0
	$f_{18}(x)$: shifted rotated Weierstrass function	$[-0.5, 0.5]$	30	0
	$f_{19}(x)$: Schwefel's Problem 2.13	$[-\pi, \pi]$	30	0
	$f_{20}(x)$: expanded extended Griewank's plus Rosenbrock's function (F8F2)	$[-5, 5]$	30	0
	$f_{21}(x)$: shifted rotated expanded Scaffer's F6	$[-100, 100]$	30	0

signed-rank nonparametric test (Arasomwan and Adewumi 2014). To investigate additional details of the search processes of these two DE algorithms, we took the mean of the best solution at each cut point over replications as the sample data of f_1 to f_6 (Derrac et al. 2014). The experimental parameters were as follows: 30 dimensions, 40 individuals, 2000 iterations, and 10 replicate executions. The cut point was set to obtain the global best result per 100 iterations over 2000 iterations for statistical analysis of the convergence curve. We set 21 cut points per 100 iterations over 2000 iterations and conducted 10 replicate test runs to verify the reliability of the experimental data. The cut point data obtained using SADE-FP and classic DE are listed in Table 7.

Table 8 presents the signed rank score on the mean at each cut point's fitness value obtained using the two DE algorithms. Three situations can exist regarding the performance of SADE-FP and DE: SADE-FP < DE, SADE-FP > DE, and SADE-FP = DE. As given for f_1, f_4, and f_5 in Table 8, the number of instances when SADE-FP < DE was 21, indicating that the results obtained using SADE-FP were better than those obtained using classic DE over all 21 cut points. Moreover, the number of instances when SADE-FP < DE was considerably lower than that when SADE-FP = DE for f_3, and the number of instances when SADE-FP < DE was equal to that when SADE-FP = DE for f_6.

Statistical analysis was then performed by applying the Wilcoxon signed-rank nonparametric test, and the results are summarized in Table 9. The values in bold represent rejection of the null hypothesis (SADE-FP fitness > classic DE fitness). Because the P value (0.5) for f_3 was larger than the significance level (0.05), the difference SADE-FP < DE was nonsignificant. The P values for f_1, f_2, f_4, f_5, and f_6 were lower than the significance level (0.05), indicating rejection of the null hypothesis and implying that the fitness values obtained using SADE-FP were lower than those obtained using classic DE.

Overall, we discovered that the fitness obtained using SADE-FP was lower than that obtained using classic DE for five test functions and equal to that obtained using classic DE for f_3 based on the number of instances of SADE-FP = DE in Table 8.

4.4 Example: production scheduling problem

In this study, we consider production scheduling as an industrial application example. In a typical production scheduling problem, most studies discuss factors such as production sequence, production routing, machine availability, and machine downtime. The common objective of production scheduling is to minimize tardiness in production. In our example, there are four orders (Table 10) and five machines (Table 11). Each order has a start time, due date, and some required machines. If the order is completed after the due date, a penalty may be incurred and customer satisfaction is reduced. Start time indicates the earliest available time to start the production job. Each machine has its own distinct maintenance due date and required maintenance period. Machines need to be maintained, otherwise they cannot be used after the due date. The scheduling problem to minimize tardiness can be formulated as follows:

$$Z = \min \sum_{i=1}^{n} T_i \tag{7}$$

$$\text{s.t.} \qquad T_i = C_i - d_i,\ i = 1, 2, \ldots, n, \tag{8}$$

Table 6 Experimental results of $D = 30$ test functions with 300,000 maximum FEs

Function/mean(SD)	DE/best/1	DE/rand/1	MDE_pBX	DE-EPA	SADE-FP
Dimensions: 30					
$f_8(x)$	1.5489E−27	1.8331E−02	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	(1.0404E−27)	(2.0223E−02)	**(0.0000E+00)**	**(0.0000E+00)**	**(0.0000E+00)**
$f_9(x)$	8.3214E−10	1.1591E+03	4.0865E−22	**2.7452E−28**	9.0225E−02
	(1.7506E−09)	(1.1240E+03)	(1.2701E−21)	**(1.3020E−28)**	(1.0367E−01)
$f_{10}(x)$	1.0552E+06	6.9143E+07	9.2196E+04	6.6916E+04	**4.8981E+00**
	(6.8168E+05)	(3.9599E+07)	(5.6020E+04)	(3.9116E+04)	**(4.6546E−01)**
$f_{11}(x)$	1.9467E−01	5.9893E+03	7.6155E−07	**5.3721E−23**	7.3100E−05
	(2.7168E−01)	(3.1545E+03)	(2.1841E−06)	**(1.9405E−22)**	(1.3835E−04)
$f_{12}(x)$	1.1653E+03	2.9065E+03	4.8612E+02	7.9786E+02	**0.0000E+00**
	(4.6566E+02)	(1.0463E+03)	(2.5286E+02)	(3.9145E+02)	**(0.0000E+00)**
$f_{13}(x)$	1.9070E+00	2.2474E+03	1.2757E+00	6.3786E−01	**4.3414E−02**
	(1.9818E+00)	(6.1086E+03)	(1.8597E+00)	(1.4531E+00)	**(1.0578E−02)**
$f_{14}(x)$	1.8302E−02	1.0121E+00	1.7619E−02	**6.3083E−03**	1.0451E−02
	(1.6800E−02)	(5.1349E−02)	(1.1147E−02)	**(5.1236E−03)**	(1.4732E−02)
$f_{15}(x)$	2.0946E+01	2.0947E+01	**2.0063E+01**	2.0517E+01	2.0910E+01
	(1.7939E−02)	(1.7939E−02)	**(1.1973E−02)**	(1.1973E−02)	(5.0211E−02)
$f_{16}(x)$	6.5629E+01	5.1668E+01	1.7369E+01	1.0177E+01	**8.7004E−01**
	(1.3654E+01)	(1.3654E+01)	(3.9022E+00)	(6.1192E+00)	**(9.7428E−01)**
$f_{17}(x)$	7.3348E+01	2.2894E+02	2.6816E+01	3.7132E+01	**2.6106E+01**
	(1.8695E+01)	(1.5465E+01)	(1.0360E+01)	(1.1744E+01)	**(6.4065E+00)**
$f_{18}(x)$	3.8998E+01	3.9078E+01	1.9810E+01	**1.3874E+01**	3.4576E+01
	(7.1857E−01)	(7.1857E−01)	(3.9717E−01)	**(3.9717E−01)**	(9.8017E−01)
$f_{19}(x)$	3.6825E+05	4.7866E+05	2.1400E+05	1.3653E+05	**1.5892E+04**
	(8.9462E+04)	(8.0776E+04)	(3.8571E+04)	(4.4320E+04)	**(1.0255E+04)**
$f_{20}(x)$	5.3427E+00	1.1698E+01	4.2723E+00	1.8879E+00	**1.7480E+00**
	(1.8142E+00)	(1.2054E+00)	(7.3174E−01)	(3.7689E−01)	**(9.7137E−01)**
$f_{21}(x)$	1.3710E+01	1.4254E+01	1.2744E+01	**1.2638E+01**	1.3001E+01
	(9.4731E−03)	(9.4731E−03)	(6.0359E−03)	**(6.0359E−03)**	(1.5805E−01)

$$T_i \geq 0,\ i = 1, 2, \ldots, n, \tag{9}$$

the capacity and availability constraints of the machine, (10)

where constraints (8) and (9) reflect the definitions of job tardiness.

In this experiment, which was run 30 times, the maximum FE criterion was set to 300,000 by referring to Suganthan et al. (2005). For this problem, the optimal solution was known; that is, the shortest delay time is 2. We compared the delayed times of this problem obtained using SADE-FP, classic DE/rand/1, DE/rand/2, DE/best/1, and DE/current-to-best/1. The average solution obtained using classical DE/rand/1 was 6.63, DE/rand/2 was 2.8, and SADE-FP was 2.5. The averages and SDs obtained using the five DE algorithms are listed in Table 12. Classic DE found the optimal solution in 11 of 30 runs, whereas SADE-FP obtained the optimal solution in 15 of 30 runs. Furthermore, SADE-FP converged faster than the other DE algorithms, as illustrated in Fig. 7. Based on the experimental results, SADE-FP performed better than the other DE algorithms in the industrial problem.

Table 7 Twenty-one cut points obtained using SADE-FP and classic DE on six test functions

Test function	f_1		f_2		f_3	
Cut points	DE	SADE-FP	DE	SADE-FP	DE	SADE-FP
1	65,068.838	64,032.7868	3.147E+11	1.3998E+12	63,140.7333	66,801.9
2	1794.0537	47.0746386	45.4953232	0.64352056	1832.03333	32.7
3	46.8595314	0.02005565	7.07343624	0.00351262	48.9333333	0
4	1.18855804	8.2117E−06	1.10090794	1.9546E−05	2.13333333	0
5	0.0309829	3.5328E−09	0.1643956	1.0771E−07	0.03333333	0
6	0.00085042	1.6153E−12	0.02415165	5.8471E−10	0	0
7	2.2369E−05	6.9758E−16	0.00378107	3.0454E−12	0	0
8	4.9672E−07	2.7155E−19	0.00053834	1.5174E−14	0	0
9	1.3169E−08	1.3711E−22	7.8637E−05	8.0985E−17	0	0
10	3.3515E−10	6.2436E−26	1.1396E−05	4.3352E−19	0	0
11	7.9647E−12	2.6274E−29	1.7527E−06	2.0819E−21	0	0
12	1.8976E−13	1.1187E−32	2.5677E−07	1.1125E−23	0	0
13	5.1631E−15	4.1847E−36	3.8319E−08	5.7734E−26	0	0
14	1.3129E−16	1.9014E−39	5.5573E−09	3.0297E−28	0	0
15	3.6082E−18	8.2594E−43	8.2848E−10	1.6849E−30	0	0
16	1.0734E−19	3.5154E−46	1.2046E−10	9.4481E−33	0	0
17	2.4169E−21	1.2483E−49	1.8646E−11	4.7187E−35	0	0
18	6.1493E−23	5.4125E−53	2.7182E−12	2.5263E−37	0	0
19	1.4957E−24	2.6226E−56	4.1196E−13	1.3613E−39	0	0
20	3.4931E−26	9.8112E−60	6.453E−14	7.7888E−42	0	0
21	9.5517E−28	4.2481E−63	9.7579E−15	4.2567E−44	0	0
Test function	f_4		f_5		f_6	
Cut points	DE	SADE-FP	DE	SADE-FP	DE	SADE-FP
1	436.287607	432.095576	20.6217586	20.619706	572.183705	606.875117
2	249.868688	110.180682	10.0016653	5.50297993	17.95094	1.40166667
3	220.090126	40.0828848	3.30520532	0.09729421	1.43659488	0.06938333
4	206.299092	16.7331388	0.63548346	0.00149498	0.8890599	0.00030027
5	197.224881	4.30180995	0.06017925	3.0014E−05	0.08467883	1.7287E−07
6	191.073381	0.21343738	0.00865888	6.1658E−07	0.00469573	1.1343E−10
7	186.091621	0.00017467	0.00136898	1.2672E−08	0.00228439	8.6661E−14
8	179.378137	1.3025E−07	0.00020346	2.5123E−10	0.00222014	1.48E−17
9	171.815601	6.6622E−11	3.1133E−05	5.3733E−12	0.00221805	0
10	164.17286	2.3626E−14	4.9578E−06	1.0939E−13	0.00221799	0
11	155.630872	0	7.273E−07	8.0232E−15	0.00221799	0
12	143.250397	0	1.1932E−07	6.9574E−15	0	0
13	138.214124	0	2.0391E−08	6.7206E−15	0	0
14	126.728257	0	3.2154E−09	6.6021E−15	0	0
15	118.230114	0	5.2413E−10	6.3653E−15	0	0
16	111.989705	0	8.2968E−11	6.3653E−15	0	0
17	98.2563776	0	1.3684E−11	6.2469E−15	0	0
18	90.1002176	0	2.2331E−12	6.1284E−15	0	0
19	79.5494428	0	3.7277E−13	6.1284E−15	0	0
20	72.9479515	0	5.3972E−14	6.01E−15	0	0
21	65.850027	0	1.2523E−14	5.8916E−15	0	0

Table 8 Signed rank score on mean of fitness values of 100 iterations (until 2000 iterations) obtained using the proposed SADE-FP and classic DE for six test problems

Measurement	Number	Mean of scaled	Sum of scaled	Number	Mean of scaled	Sum of scaled	Number	Mean of scaled	Sum of scaled
Problems	f_1			f_2			f_3		
SADE-FP < DE	21	11	231	20	10.5	210	4	2.5	10
SADE-FP > DE	0	0	0	1	21	21	1	5	5
SADE-FP = DE	0			0			16		
Sum	21			21			21		
Problems	f_4			f_5			f_6		
SADE-FP < DE	21	11	231	21	11	231	10	5.5	55
SADE-FP > DE	0	0	0	0	0	0	1	11	11
SADE-FP = DE	0			0			10		
Sum	21			21			21		

Table 9 Wilcoxon signed-rank test of mean fitness values obtained using SADE-FP and classic DE (significance level = 0.05)

	f_1	f_2	f_3	f_4	f_5	f_6
Z	−4.015	−3.285	−0.674	−4.015	−4.015	−1.956
P value	**0**	**0.001**	0.5	**0**	**0**	**0**

H_0: SADE-FP fitness > Classic DE fitness. Bold value implies rejection of the null hypothesis (H_0)

Table 10 Order information

Order	Start time	Order due date	Machines required
O_1	1	5	M_2, M_3, M_4
O_1	7	14	M_1, M_2, M_3, M_5
O_2	15	20	M_2, M_3, M_4
O_2	30	37	M_1, M_2, M_3, M_5

Table 11 Machine information

Machines	Maintenance due date	Maintenance time
M_1	15	2
M_2	30	2
M_3	35	2
M_4	10	2
M_5	40	2

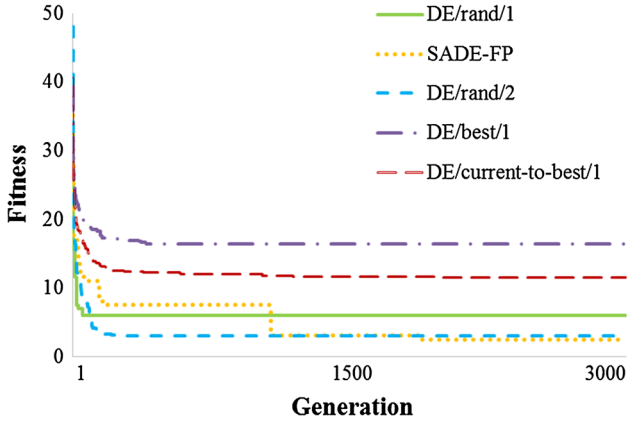

Fig. 7 Convergence rate for the production scheduling problem

5 Conclusion

The disadvantages of classic DE include unstable convergence, slow convergence in the late stages of evolution, and a tendency to become stuck in a local optimum (Jia et al. 2011; Lee and Chiang 2011). However, the traditional method of manually selecting appropriate values of the parameters is time-consuming. Therefore, numerous studies have focused on self-adaptive mechanisms to obtain favorable solutions and rapid convergence. We propose self-adaptive controlling parameters of DE based on fitness performance with a perturbation strategy (SADE-FP). Our results demonstrate that in solving unimodal and multimodal functions, compared with four other DE algorithms, SADE-FP obtains a better solu-

Table 12 Average and standard deviation of delay times obtained using five DE algorithms in the production scheduling problem

	DE/rand/1	DE/rand/2	DE/best/1	DE/current-to-best/1	SADE-FP
Ave.	6.63	2.8	16.4	11.4	2.5
SD	4.00	2.44	9.71	9.11	0.50

tion and converges faster, without becoming stuck at a local optimum.

Furthermore, the experimental results obtained reveal that SADE-FP satisfies five of the six characteristics of a favorable scheme for adaptive control of DE parameters, which were summarized by Chen and Chiang (2015); the only exception is the mechanism of a random distribution. Therefore, applying a random distribution in the parameter adaptation mechanism would be a worthwhile direction in future research.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflicts of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Aleti A, Moser I (2016) A systematic literature review of adaptive parameter control methods for evolutionary algorithms. ACM Comput Surv (CSUR) 49(3):56

Arasomwan MA, Adewumi AO (2014) Improved particle swarm optimization with a collective local unimodal search for continuous optimization problems. Sci World J 2014:798129. https://doi.org/10.1155/2014/798129

Brest J, Greiner S, Bošković B, Mernik M, Zumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evolut Comput 10:646–657

Brest J, Bošković B, Greiner S, Žumer V, Maučec MS (2007) Performance comparison of self-adaptive and adaptive differential evolution algorithms. Soft Comput 11:617–629

Chen C-A, Chiang T-C (2015) Adaptive differential evolution: a visual comparison. In: IEEE congress on evolutionary computation (CEC), IEEE, pp 401–408

Chiang T-C, Chen C-N, Lin Y-C (2013) Parameter control mechanisms in differential evolution: a tutorial review and taxonomy. In: 2013 IEEE symposium on differential evolution (SDE), IEEE, pp 1–8

Chuan-Kang T, Chih-Hui H (2009) Varying number of difference vectors in differential evolution. In: IEEE congress on evolutionary computation (CEC), pp 1351–1358. https://doi.org/10.1109/CEC.2009.4983101

Das S, Suganthan PN (2011) Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput 15(1):4–31

Das S, Konar A, Chakraborty UK (2005) Improved differential evolution algorithms for handling noisy optimization problems. In: The 2005 IEEE congress on evolutionary computation, 2005. IEEE, pp 1691–1698

De Falco I, Della Cioppa A, Maisto D, Scafuri U, Tarantino E (2014) An adaptive invasion-based model for distributed differential evolution. Inf Sci 278:653–672

Derrac J, García S, Hui S, Suganthan PN, Herrera F (2014) Analyzing convergence performance of evolutionary algorithms: a statistical approach. Inf Sci 289:41–58

Dexuan Z, Liqun G (2012) An efficient improved differential evolution algorithm. In: Chinese control conference (CCC), IEEE, pp 2385–2390

Eiben AE, Hinterding R, Michalewicz Z (1999) Parameter control in evolutionary algorithms. IEEE Trans Evolut Comput 3:124–141

Fan Q, Yan X (2015) Differential evolution algorithm with self-adaptive strategy and control parameters for P-xylene oxidation process optimization. Soft Comput 19:1363–1391

Hsieh S-T, Su T, Wu H-L (2013) An improved differential evolution with efficient parameters adjustment. In: 2013 first international symposium on computing and networking (CANDAR), IEEE, pp 627–629

Hu Z, Xiong S, Su Q, Zhang X (2013) Sufficient conditions for global convergence of differential evolution algorithm. J Appl Math 2013:139196

Iacca G, Caraffini F, Neri F (2012) Compact differential evolution light: high performance despite limited memory requirement and modest computational overhead. J Comput Sci Technol 27:1056–1076

Islam SM, Das S, Ghosh S, Roy S, Suganthan PN (2012) An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. IEEE Trans Syst Man Cybern Part B (Cybern) 42:482–500

Jia D, Zheng G, Khan MK (2011) An effective memetic differential evolution algorithm based on chaotic local search. Inf Sci 181:3175–3187

Jiang LL, Maskell DL, Patra JC (2013) Parameter estimation of solar cells and modules using an improved adaptive differential evolution algorithm. Appl Energy 112:185–193

Lee W-PC, Chang-Yu Cai, Wan-Ting (2011) A differential evolution algorithm with perturb strategy. In: International journal of advanced information technologies (IJAIT) p 5

Lee W-P, Chiang C-Y (2011) A self-adaptive differential evolution algorithm with dimension perturb strategy. J Comput 6:524–531

Li X, Yin M (2016) Modified differential evolution with self-adaptive parameters method. J Comb Optim 31:546–576

Lin Y-C, Cheng C-Y (2015) Self-adaptive parameters adjusting in differential evolution based on fitness information. Paper presented at the 15' CIIE Chinese institute of industrial engineers,

Liu J, Lampinen J (2005) A fuzzy adaptive differential evolution algorithm. Soft Comput 9:448–462

Mezura-Montes E, Velázquez-Reyes J, Coello Coello CA (2006) A comparative study of differential evolution variants for global optimization. In: Proceedings of the 8th annual conference on genetic and evolutionary computation, ACM, pp 485–492

Mi M, Huifeng X, Ming Z, Yu G (2010) An improved differential evolution algorithm for TSP problem. In: International conference on intelligent computation technology and automation (ICICTA), IEEE, pp 544–547

Omran MG, Salman A, Engelbrecht AP (2005) Self-adaptive differential evolution. In: Computational intelligence and security. Springer, Berlin, pp 192–199

Ponsich A, Coello CAC (2013) A hybrid differential evolution–tabu search algorithm for the solution of job-shop scheduling problems. Appl Soft Comput 13(1):462–474

Price K, Storn R, Lampinen J (2005) Differential evolution–a practical approach to global optimization. Springer, Berlin

Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evolut Comput 13:398–417

Rajesh K, Bhuvanesh A, Kannan S, Thangaraj C (2016) Least cost generation expansion planning with solar power plant using differential evolution algorithm. Renew Energy 85:677–686

Salman A, Engelbrecht AP, Omran MG (2007) Empirical analysis of self-adaptive differential evolution. Eur J Oper Res 183:785–804

Sauer JG, Coelho LDS (2008) Discrete differential evolution with local search to solve the traveling salesman problem: fundamentals and case studies. In: IEEE international conference on cybernetic intelligent systems. IEEE, pp 1–6

Storn R, Price K (1997) Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11:341–359

Suganthan PN, Hansen N, Liang JJ, Deb K, Chen Y-P, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. KanGAL report 2005005:2005

Tang L, Zhao Y, Liu J (2014) An improved differential evolution algorithm for practical dynamic scheduling in steelmaking-continuous casting production. IEEE Trans Evolut Comput 18:209–225

Trivedi A, Srinivasan D, Biswas S, Reindl T (2015) Hybridizing genetic algorithm with differential evolution for solving the unit commitment scheduling problem. Swarm Evolut Comput 23:50–64

Wang HB, Ren XN, Li GQ, Tu XY (2016) APDDE: self-adaptive parameter dynamics differential evolution algorithm. Soft Comput 1–21

Xue F, Sanderson AC, Graves RJ (2009) Multiobjective evolutionary decision support for design-supplier-manufacturing planning Systems. IEEE Trans Man Cybern, Part A: Syst Hum 39:309–320

Yildiz AR (2013) Hybrid Taguchi-differential evolution algorithm for optimization of multi-pass turning operations. Appl Soft Comput 13(3):1433–1439

Zaharie D (2007) A comparative analysis of crossover variants in differential evolution. In: Proceedings of IMCSIT pp 171–181

Zhang J, Sanderson AC (2007) JADE: self-adaptive differential evolution with fast and reliable convergence performance. In: IEEE congress on evolutionary computation, IEEE, pp 2251–2258