International Journal of Control and Automation
Vol.7, No.9 (2014), pp.81-96
http://dx.doi.org/10.14257/ijca.2014.7.9.07

Improved Differential Evolution Algorithm based on Dynamic Adaptive Strategies and Control Parameters

Congjiao Wang, Xihuai Wang, Jianmei Xiao and Yi Ding

Department of Electrical Engineering and Automation, Shanghai Maritime University, Shanghai 201306, China
qinxiaojiayi@hotmail.com

Abstract

To solve the slow convergence speed, low precision in later period and tedious parameter setting of differential evolution when applied to complex optimization functions, an improved differential evolution algorithm (dn-DADE) based on dynamic adaptive strategy is proposed. Firstly, the elite solutions of current population are utilized in the new mutation strategy (DE/current-to-dnbest/1) to guide the search direction, and then these optional elite solutions tend to the global optimal solution in the late stage of evolution to balance the diversity of population and convergence speed. Secondly, the adaptive update strategies of scaling factor and crossover factor are designed for control parameter values self-adapting at different search stages, thus improve the stability and robustness of the algorithm. A set of 14 benchmark functions is adopted to test the performance of the proposed algorithm. The results show that dn-DADE algorithm has the advantages of remarkable optimizing ability, higher search precision, faster convergence speed and outperforms several state-of-the-art improved differential evolution algorithms in terms of the main performance indexes.

Keywords: *Differential evolution, Mutation strategy, Dynamic adjustment, Parameter self-adaptation, Global optimization*

1. Introduction

Differential Evolution (DE) algorithm, proposed by Storn and Price in 1995[1], is a very popular evolutionary algorithm that employs the unique differential mutation operator to probe the search space. As one of the most representative and promising evolutionary algorithm, DE has been successfully applied to the optimization problems arising from diverse domains of scientific research and engineering [2-4] (such as data mining, pattern recognition, power system design, scheduling problems) , due to its simple structure, ease of implementation, high efficiency, etc. Although DE algorithm has great potential, there is still a need for improving the performance of DE to face challenges from the modern application areas because of its some drawbacks. Empirical analysis [5-6] of DE has shown that it is more likely to trap in a local optimum and demonstrate slow convergence speed in late periods when solving the high dimensional multimodal optimization problems.

Many studies indicate that the DE performance highly depends on the mutation operation, crossover strategy and the corresponding control parameters (i.e., scale factor F , crossover factor CR and population size NP) [7-8]. DE may become premature convergence or stagnation caused by the inappropriate choice of trial vector generation strategies or parameter settings for a specific problem. At present, DE has variety of optional mutation strategies, among which DE/best/k, DE/current-to-best/k, DE/rand/k and DE/current-to-rand/k are widely used. Moreover, different mutation operators have different characteristics.

ISSN: 2005-4297 IJCA

For example, some of the mutation operators favor the exploration of the search space, while some other strategies favor its fast exploitation [9]. The explorative operators are more likely to locate the optimal solution, but need more generations. On the other hand, the exploitive operators can rapidly converge to a minimum of the function, but it is difficult to avoid the risk of premature convergence. In addition, the control parameters are known to be problem-dependent and thus the trial-and-error method for finding their appropriate values requires multiple optimization runs. Similarly, DE researchers have actively investigated the turning and adaptation mechanisms to automatically find an optimal set of control parameters [10-11].

In order to further improve the global convergence performance and the robustness of DE algorithm, an improved differential evolution algorithm based on the dynamic adaptive strategies (*dn*-DADE) is proposed in this paper with a new mutation strategy (DE/current-to-*dn*best/1) and a new adaptive scheme for F and CR . On the one hand, different from the DE/current-to-best/1 which always uses the global optimal solution to guide the evolution of population, DE/current-to-*dn*best/1 uses the elite solutions of the current population to locate larger promising regions of the search space in early evolutionary stage, while the number of local elite solutions is dynamically changed to tend towards the global optimal solution in the later evolution stage, thus to combine the exploration and exploitation capabilities of DE. On the other hand, self-adaptive schemes of two parameters are proposed to enhance the robustness of the algorithm. Experimental results demonstrate the effectiveness and efficiency of *dn*-DADE in comparison with several state-of-the-art DE variants and its own variants over a suite of 14 well-known benchmark functions.

2. Classical Differential Evolution Algorithm

The DE algorithm is a real-coded evolutionary algorithm. The process of DE, like other evolutionary algorithms, produces offspring for the next generation by mutation, crossover and selection operations, which are detailed below.

1) Initialization

Suppose population size is NP and the evolution generation $G = 0,1,...,G_{\max}$, the dimension of individual variables presents as D , then the i^{th} individual of the population at the current generation can be represented as:

$$X_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, \cdots, x_{D,i,G}] \quad (1)$$

The initial population (at G = 0) is generated by a uniformly random process within the search space constrained by the prescribed minimum and maximum bounds $[X_{\min}, X_{\max}]$ .Then the j^{th} dimension of the i^{th} individual is initialized as:

$$x_{j,i,0} = x_{j,\min} + rand_{i,j}[0,1] \cdot (x_{j,\max} - x_{j,\min}) \quad (2)$$

Where $x_{j,\max}$ and $x_{j,\min}$ present the upper and lower bounds of j^{th} dimension of the i^{th} individual in the population, respectively; $rand_{i,j}[0,1]$ is a random number generated from a uniform distribution on $[0,1]$.

2) Mutation Operation

After initialization, DE produces a mutant vector $V_{i,G}$ corresponding to each individual in the current population through mutation. $DE/a/b$ represent the different mutation strategies, in which a is represented as the selection mode of the basis vectors, and b is represented as the number of the difference vectors. The most frequently used mutation strategies are listed below:

$DE/rand/1$:

$$V_{i,G} = X_{r1,G} + F \cdot (X_{r2,G} - X_{r3,G}) \quad (3)$$

$DE/best/2$:

$$V_{i,G} = X_{best,G} + F \cdot (X_{r1,G} - X_{r2,G}) + F \cdot (X_{r3,G} - X_{r4,G}) \quad (4)$$

$DE/current-to-best/1$:

$$V_{i,G} = X_{i,G} + F \cdot (X_{best,G} - X_{i,G}) + F \cdot (X_{r1,G} - X_{r2,G}) \quad (5)$$

Where $r_1, r_2, r_3, r_4 \in [1, 2, ..., NP]$ are randomly chosen integers which represent the index number of different individuals in the population, different from each other and also different from i; $X_{best,G}$ is the best individual in the current population at generation G with the best fitness; $F \in (0, 2)$ is the mutation scale factor which controls the amplification of the difference vectors.

3) Crossover Operation

DE has two crossover schemes: binomial crossover and exponential crossover. The target vector $X_{i,G}$ and the mutation vector $V_{i,G}$ cross discretely to generate the trial vector $U_{i,G} = [u_{1,i,G}, u_{2,i,G}, \cdots, u_{D,i,G}]$. The binomial crossover operation adopted in this paper is defined as:

$$u_{j,i,G} = \begin{cases} v_{j,i,G}, rand(j) \le CR ... or ... j = randn(i) \\ x_{j,i,G}, rand(j) > CR ... and ... j \ne randn(i) \end{cases} \quad (6)$$

Where $rand(j) \in [0,1]$ is a uniformly distributed random number and $randn(i)$ is a dimension index randomly chosen from $[1, 2, ..., D]$; $CR \in [0,1]$ denotes the crossover probability factor.

4) Selection Operation

DE employs the selection operation to choose the best, between the target and the trial vector based on their fitness value for the next generation. For the global minimization problem, the greedy selection strategy is expressed as:

$$X_{i,G+1} = \begin{cases} U_{i,G}, f(U_{i,G}) < f(X_{i,G}) \\ X_{i,G}, f(U_{i,G}) \ge f(X_{i,G}) \end{cases} \quad (7)$$

Where $X_{i,G+1}$ represents the i^{th} individual of the next generation.

3. Improved Differential Evolution Algorithm based on Dynamic Adaptive Strategies

3.1. Mutation Strategy

Compared to DE/rand/*k* [1], the DE/best/*k* and DE/current-to-best/*k* have the faster convergence rate by using the best solution information to guide the evolution [12]. But due to the population diversity reduced, in many cases, these schemes may cause the algorithm into a local optimal solution. Thus, how to balance the exploration and exploitation ability of

DE became the key problem. A new strategy, named DE/current-to-*dn*best/1 is presented in this paper to solve this problem. The members selected randomly from the top $dn(dn \in [1,2,...,NP])$ of the population, which is ranked according to the fitness values, are defined as the elite solutions of the current population. And then *dn* vary nonlinearly with the generation. The mutation scheme is generated in the following manner:

$$V_{i,G} = X_{i,G} + F_i \cdot (X_{dnbest,G} - X_{i,G}) + F \cdot (X_{r_1^i,G} - X_{r_2^i,G}) \quad (8)$$

Where $X_{dnbest,G}$ is randomly selected from the elite solutions of the current population. $X_{r_1^i,G}$ and $X_{r_2^i,G}$ are the individuals randomly selected from the current population which are different from $X_{dnbest,G}$ or $X_{i,G}$. The function expression for *dn* as:

$$dn = ceil\left\{\frac{NP}{4} \cdot \left[\cos\left(\frac{G}{G_{\max}} \cdot \pi\right) + 1\right]\right\} \quad (9)$$

Where G represents the current generation number; $G_{\max}$ represents the maximum number of generations; $Ceil(y)$ indicates the minimum integer which is bigger than y. From Figure 1 (with $G_{\max} = 5000$ for example), we can see that *dn* gradually decrease from $NP/2$ to 1 with the increase of the evolution generations. The slow decrease rate of *dn* at the early search stage can make the algorithm search the feasible solution in a relatively large range, and coordinate the conflict between the "greedy" of mutation operation and the diversity of population; while the fast decrease rate during the later stage can intensify the search in the more smaller neighborhood of the best solution and also speed up convergence of the algorithm. Thus, the DE/current-to-*dn*best/1 strategy combines the exploration and exploitation abilities of the algorithm by nonlinearly downsizing the elitist solutions of the population.

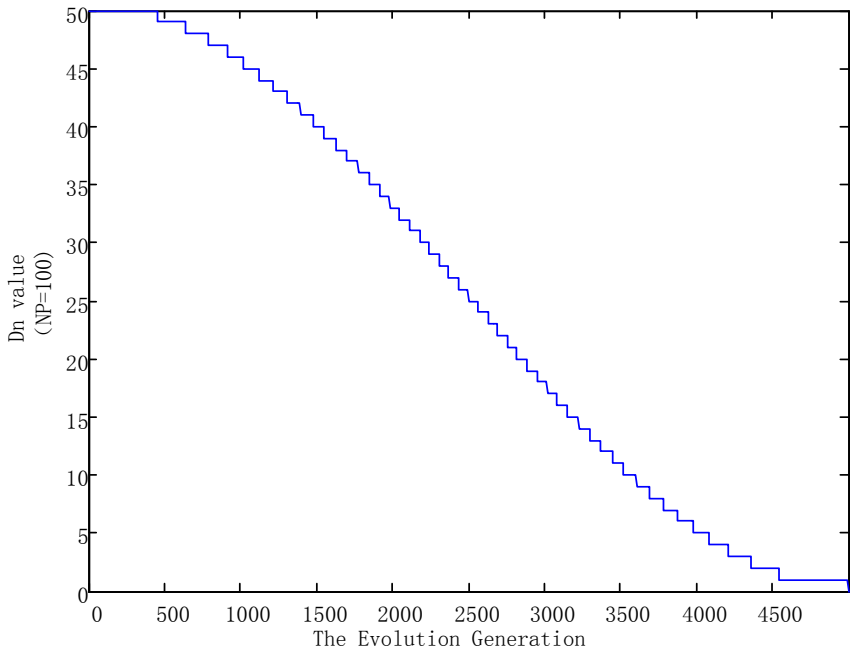


Figure 1. Change Curve of *dn*

3.2. Dynamic Adaptive Strategies of Parameters

The classic DE is very sensitive to its control parameters. The crossover probability rate CR usually relates to the properties of the problems (*i.e.*, the unimode or multimode problem, the variable independence or variable interdependence problem); Scale factor *F* effects on the convergence speed greatly. In our *dn*-DADE algorithm, CR is generated by a normal distribution, and we use the improvement rate of fitness value and the successful CR , which have produced trial vectors successfully entering the next generation, to adjust the parameters with the normal distribution. For *F*, we attempt to maintain larger *F* value at the beginning search stages and smaller *F* value during the later stages with a Cauchy distribution. This scheme is more helpful to keep both global (with large *F* values) and local (with small *F* values) ability throughout the entire evolution process.

1) Adaptive Crossover Probability Rate

Inspired by the idea that update CR by the feedback information of previous CR values in SaDE[7], *dn*-DADE adopts a weighted updating strategy which also considering the feedback information of the fitness value improvement rate to guide the generation of new CR .

The crossover probability rate for each individual according to:

$$CR_{i,G} = N(CR_{dn}, \sigma_{dn}^2) \tag{10}$$

Where $N(CR_{dn}, \sigma_{dn}^2)$ denotes a normal distribution with the mean value CR_{dn} and variance σ_{dn}^2 , and need to be truncated to [0, 1]. For each generation of the evolution process, the corresponding CR value associated with the trial vectors successfully entering into next generation are stored in an array M_{CR} , while the improvement rate ratios of the fitness values of the successful trial vectors are stored in an array $M_{\Delta f}$:

$$\Delta f(k) = \frac{f(k) - f_{new}(k)}{f(k)} \tag{11}$$

Where $f(k)$ and $f_{new}(k)$ denote the fitness value of individual before and after the evolution operation, respectively. Then the weighted adaptation scheme for CR can be expressed as:

$$CR_{dn} = \sum_{k=1}^{|M_{CR}|} \omega_k \times M_{CR}(k) \tag{12}$$

$$\omega_k = \frac{\Delta f(k)}{\sum_{k=1}^{|M_{\Delta f}|} \Delta f(k)} \tag{13}$$

Denote $|M_{\Delta f}| = |M_{CR}|$ as the number of the successful trial vectors. And the initial value of CR_{dn} is set as 0.5. Similarly, the self-adaptive σ_{dn}^2 according to:

$$\sigma_{dn}^2 = \frac{\sum_{k=1}^{|M_{CR}|} (M_{CR}(k) - CR_{dn})^2}{|M_{CR}|} \tag{14}$$

Since CR determines how many components of the mutated vector will be introduced into the population for the next generation, the probability of producing improved offspring from the same parent with a small CR is higher than that with a large CR . As we have known CR of SaDE algorithm has an implicit bias towards small values throughout the entire adaptation

process. This bias might be pernicious in case of handling nonseparable functions, while large CR value is required to change the nonseparable variables together. In addition, it is assumed that large successful CR values will obtain greater improvement on fitness values than small successful CR values for nonseparable problems because of its strong ability to change nonseparable variables together. Therefore, we can balance the bias with a weight based on the rate of the fitness value improvement.

2) Adaptive Scale Factor

Compared to the normal distribution, the Cauchy distribution is more helpful to expand the search range and thus avoid trapping into the local optimal. Therefore, the scale factor $F_{i,G}$ of each individual in *dn*-DADE is generated independently according to:

$$F_{i,G} = C(F_{dn}, r) \tag{15}$$

Where $C(F_{dn}, r)$ denotes a Cauchy distributed with the location parameter F_{dn} and the scale parameter r. The smaller values of *F* may lead to premature convergence; and the larger value may reduce the convergence speed. Therefore, we consider gradually adjusting the actual value of $F_{i,G}$ in the truncation interval $[F_{\min}, F_{\max}]$. In this paper, we set $r = 0.05$ and the initial value of F_{dn} as 0.7, and then dynamically update the F_{dn} value based on evolution generation as following:

$$\begin{cases} F_{dn} = F'_{\max} - (F'_{\max} - F'_{\min}) \times \left(\dfrac{G}{G_{\max}} \right)^{1/2} \\ F'_{\max} = F_{\max} - \theta r \\ F'_{\min} = F_{\min} + \theta r \end{cases} \tag{16}$$

According to the characteristics of Cauchy distribution, it will lead the $F_{i,G}$ value has a great probability to over the truncation interval that F_{dn} value fall near the interval boundary $[F_{\min}, F_{\max}]$ .Therefore, it is need to add an adjustment factor $\theta r (\theta = 1, 2, 3...)$, and the role of θ and the relevant value will be shown in the experimental analysis.

3.3 Realization of the *dn*-DADE Algorithm

The pseudo code of *dn*-DADE algorithm is shown in Figure 2:

1: Set the population size NP , the variable dimension D , max evolution generation $G_{\max}$ and the initial value of CR_{dn} , σ_{dn}^2 , F_{dn} and r
2: Set the initial evolution generation $G = 0$; and set $M_{\Delta f} = \varnothing$, $M_{CR} = \varnothing$
3: Create the initial population $\{X_{i,0} | i = 1,2,...,NP\}$ and sequence the fitness value of all the individuals in the initial population
4: **while** $G \le G_{\max}$ do
5: **for** $i = 1$ to NP do
6: Choose the individual $X_{r1,G}$ and $X_{r2,G}$ from the current population and set $X_{r1,G} \ne X_{r2,G} \ne X_{i,G}$
7: Calculate F_{dn} according to the formula (16)
8: Update $F_{i,G} = C(F_{dn}, 0.05)$
9: Calculate CR_{dn} and σ_{dn}^2 according to formula (12) and (14)
10: Update $CR_{i,G} = N(CR_{dn}, \sigma_{dn}^2)$
11: Calculate *dn* according to formula (9) and randomly choose an individual from the elite solutions as X_{dnbest}
12: Generate the mutation vector V by the DE/current-to-*dn*best/1 strategy according to formula (8)
13: Generate the trial vector U by the crossover operator according to formula (6)
14: **end for**
15: $G = G + 1$
16: **for** $i = 1$ to NP do
17: **if** $f(U_{i,G}) < f(X_{i,G})$ then
18: $X_{i,G}$ is replaced by $U_{i,G}$
19: Calculate the modification degree of individual fitness Δf according to the formula (11)
20: $\Delta f \to M_{\Delta f}$; $CR \to M_{CR}$
21: **end if**
22: **end for**
23: **end while**

Figure 2. Pseudo Code of *dn*-DADE Algorithm

4. Experiments and Results

4.1. Test Functions

In order to evaluate the performance of *dn*-DADE algorithm, we use a test bed of 14 standard benchmark functions provided by CEC2005 special session. These functions are based on basic benchmark functions (i.e. Rastrigin's, Rosenbrock's, Griewank's, Ackley's function), which have diverse problem features such as noise in fitness, ill-conditioning, multimodality and rotation. Details of these functions can be found in [13]. In summary, J_1- J_5 are unimodal functions and J_6- J_{14} are multimodal functions (J_{13} and J_{14} are the hybrid composition functions combined by several classical functions).

4.2. Experimental Setup

dn-DADE is compared with four advanced DE algorithms, including jDE[8], SaDE[7], JADE[11] and DEahcSPX[14]. Follow the parameter settings for all the contestant algorithms

in their original literatures: for jDE, $F_l = 0.1$, $F_u = 0.9$, $\tau_1 = \tau_2 = 0.1$; for JADE, $c = 0.1$, $p = 0.05$; for DEahcSPX, $F = 0.9$, $CR = 0.9$, $np = 3$; and the setting for SADE refer to [7].

In order to further illustrate the benefits from the new mutation strategy and adaptive parameters in *dn*-DADE, we designed three variant algorithms, including *dn*-DE, DADE and *dnc*-DADE, to compare with our proposed *dn*-DADE. For fair comparison, the population size *NP* is set as 100 regardless of the dimension D, and all experiments were run 50 times on each function independently.

4.3. Comparison of *dn*-DADE with Other Advanced Optimization Algorithms

Functions f_1 - f_{14} are tested in 30 and 100 dimensions to evaluate the performance of all the competitor algorithms in aspects of the quality of the final solutions, the convergence speed and robustness. Since the optimal value of the rotation function with center offset is not at the origin, we take the error value between the actual output value and the optimal value in theory as the evaluation criteria. Denote x^* as the global optimal solution of objective function as known and x as the actual optimal solution by the algorithm, then the error value (EV) is defined as $EV = f(x) - f(x^*)$.

The maximum number of function evaluations (MAX_FES) is set to $3*10^5$ for 30-dimension problems, and $1*10^6$ for 100-dimension problems. Table 1-2 show the mean and standard deviation of EV for 50 independent runs on 14 benchmarks functions for $D = 30$ and $D = 100$, respectively. Instead of t-test, we use the Wilcoxon rank sum test [15] (significant level 5%), which is a kind of nonparametric hypothesis test, to do statistical analysis of the data from Table 1-2. And then the P-values of Wilcoxon rank sum test are shown in Table 3-4, where NA stands for not applicable for the best performing algorithm itself in each case. The best results are typed in bold.

Table 1. Mean Value and Standard Deviation of the Error Values for f_1 - f_{14} ($D = 30$)

Algorithm / Function	jDE	SaDE	JADE	DEahcSPX	*dn*-DADE
	Mean value (standard deviation)	**Mean value** (standard deviation)	**Mean value** (standard deviation)	**Mean value** (standard deviation)	**Mean value** (standard deviation)
f_1(x)	2.18E-25 (6.92E-27)	1.09E-27 (6.42E-28)	7.75E-55 (6.48E-54)	2.77E-24 (5.34E-25)	**7.25E-58 (1.83E-60)**
f_2(x)	3.65E-08 (7.41E-09)	6.27E-09 (7.29E-10)	4.16E-25 (8.57E-26)	8.14E-06 (1.31E-05)	**5.17E-26 (4.64E-26)**
f_3(x)	1.17E+04 (5.79E+03)	9.18E+04 (5.71E+04)	7.47E+03 (1.15E+03)	3.15E+08 (4.01E+07)	**2.05E+03 (8.28E+03)**
f_4(x)	5.13E+00 (6.20E-01)	9.14E-05 (6.01E-05)	4.01E-05 (7.23E-07)	1.01E+00 (9.10E+00)	**1.28E-07 (7.02E-08)**
f_5(x)	3.07E+03 (2.06E+03)	6.75E+02 (2.19E+02)	7.78E+02 (5.14E+02)	1.08E+03 (4.05E+03)	**1.71E+02 (2.69E+02)**
f_6(x)	1.67E+02 (2.13E+00)	1.44E+02 (4.27E+01)	7.28E+00 (9.91E+00)	3.28E+07 (1.85E+05)	**2.72E-01 (8.14E-01)**
f_7(x)	9.18E-03 (3.01E-03)	8.06E-03 (3.67E-03)	4.59E-03 (3.18E-03)	8,24E+00 (1.33E-01)	**4.06E-03 (2.59E-03)**
f_8(x)	2.09E+01 (1.37E-02)	2.09E+01 (2.18E-02)	2.01E+01 (3.89E-02)	2.09E+01 (4.17E-02)	**2.01E+01 (1.88E-02)**
f_9(x)	5.37E-23 (4.43E-24)	3.71E-19 (2.01E-21)	3.25E-27 (6.47E-29)	4.01E-15 (6.22E-16)	**2.04E-33 (5.01E-37)**
f_{10}(x)	7.23E+01 (3.08E+01)	6.17E+01 (5.68E+00)	4.83E+01 (7.65E+00)	4.26E+01 (5.18E+00)	**3.97E+01 (2.61E+00)**
f_{11}(x)	5.77E+01 (3.18E+01)	3.05E+01 (2.17E+00)	2.99E+01 (2.08E+00)	8.64E+01 (7.21E+00)	**1.45E+01 (5.84E+00)**

$f_{12}(x)$	1.05E+05 (7.71E+04)	7.89E+02 (8.59E+01)	6.18E+04 (4.39E+03)	**7.61E+02 (5.35E+01)**	2.37E+03 (1.05E+03)
$f_{13}(x)$	3.44E+00 (3.84E-01)	2.81E+00 (5.64E-02)	3.14E+00 (8.78E-02)	4.36E+01 (2.58E+00)	**2.13E+00 (3.82E-02)**
$f_{14}(x)$	4.47E+01 (1.23E-01)	2.29E+01 (4.38E-01)	2.09E+01 (3.91E-01)	3.58E+02 (7.69E+01)	**1.54E+01 (2.03E-01)**

Table 2. Mean Value and Standard Deviation of the Error Values for f_1 - f_{14} ($D = 100$)

Algorithm / Function	jDE	SaDE	JADE	DEahcSPX	*dn*-DADE
	Mean value (standard deviation**)**	**Mean value** (standard deviation**)**	**Mean value** (standard deviation**)**	**Mean value** (standard deviation**)**	**Mean value** (standard deviation**)**
$f_1(x)$	3.62E-16 (4.91E-18)	4.98E-18 (7.46E-19)	5.16E-21 (3.88E-23)	5.94E-10 (7.78E-12)	**6.81E-33 (4.76E-37)**
$f_2(x)$	7.69E-06 (5.46E-07)	3.16E-02 (4.49E-03)	7.91E-07 (5.04E-08)	2.88E+00 (1.87E-01)	**2.73E-10 (5.47E-12)**
$f_3(x)$	1.14E+07 (5.83E+05)	6.49E+06 (3.06E+03)	1.93E+06 (5.58E+04)	2.58E+07 (3.44E+06)	**8.43E+05 (1.56E+04**
$f_4(x)$	4.78E+04 (3.13E+03)	2.89E+04 (5.98E+03)	3.07E+04 (4.57E+03)	5.91E+04 (1.31E+04)	**2.84E+03 (1.16E+03)**
$f_5(x)$	1.08E+05 (4.31E+03)	8.57E+04 (5.08E+03)	5.36E+04 (2.41E+03)	7.67E+04 (3.58E+04)	**2.19E+04 (3.18E+03)**
$f_6(x)$	4.92E+02 (7.15E+00)	2.87E+02 (9.06E+01)	8.96E+00 (6.18E-01)	2.35E+03 (4.48E+02)	**5.81E+00 (4.56E-01)**
$f_7(x)$	1.05E-01 (6.43E-02)	8.02E-02 (4.14E-02)	7.84E-02 (3.26E-02)	3.64E-01 (2.28E-01)	**1.46E-02 (7.81E-03)**
$f_8(x)$	2.21E+01 (3.46E+00)	2.21E+01 (5.18E+00)	**2.11E+01 (8.97E-01)**	2.21E+01 (7.64E+01)	2.11E+01 (1.36E+00)
$f_9(x)$	9.98E+02 (5.37E+02)	9.08E+02 (4.26E+02)	8.07E+02 (1.84E+01)	9.67E+02 (5.04E+02)	**5.68E+02 (3.75E+00)**
$f_{10}(x)$	5.76E+02 (8.43E+01)	7.84E+02 (2.61E+02)	3.55E+02 (6.07E+01)	8.59E+02 (4.65E+02)	**1.84E+02 (4.43E+01)**
$f_{11}(x)$	5.83E+02 (2.75E+01)	7.69E+02 (2.24E+01)	1.58E+02 (9.07E+00)	1.24E+03 (6.28E+01)	**1.04E+02 (7.81E+00)**
$f_{12}(x)$	6.05E+04 (4.86E+03)	**8.81E+03 (1.24E+03)**	4.59E+04 (1.94E+04)	8.16E+04 (3.83E+04)	3.47E+04 (6.63E+03)
$f_{13}(x)$	4.72E+01 (3.06E+01)	6.58E+00 (3.59E+00)	**5.46E+00 (3.95E+00)**	6.89E+01 (5.90E+01)	2.17E+01 (5.38E+00)
$f_{14}(x)$	3.32E+01 (4.77E-01)	2.84E+01 (2.97E-01)	2.26E+01 (3.41E-01)	3.47E+01 (5.54E-01)	**1.84E+01 (1.59E-01)**

Table 3. P-values of the Wilcoxon Rank Sum Test (significance level 5%) for Test Function f_1 - f_{14} ($D = 30$)

Algorithm / Function	jDE	SaDE	JADE	DEahcSPX	*dn*-DADE
$f_1(x)$	4.19E-17	5.82E-17	2.74E-22	7.96E-17	**NA**
$f_2(x)$	2.13E-11	5.26E-16	7.63E-23	1.69E-07	**NA**
$f_3(x)$	6.18E-04	2.84E-06	1.04E-04	3.53E-02	**NA**
$f_4(x)$	5.17E-04	6.62E-03	3.47E-04	9.18E-04	**NA**
$f_5(x)$	1.82E-02	5.47E-05	2.88E-04	8.19E-03	**NA**
$f_6(x)$	5.19E-10	4.98E-09	8.27E-10	1.82E-04	**NA**
$f_7(x)$	6.52E-13	7.71E-05	5.38E-04	4.19E-02	**NA**
$f_8(x)$	4.75E-12	6.82E-11	1.09E-07	7.57E-03	**NA**
$f_9(x)$	3.19E-11	5.48E-17	8.42E-22	2.14E-20	**NA**
$f_{10}(x)$	5.13E-15	7.22E-16	1.15E-17	2.18E-15	**NA**
$f_{11}(x)$	4.41E-07	6.92E-18	5.59E-18	2.97E-05	**NA**

$f_{12}(x)$	5.84E-18	3.72E-19	1.89E-04	**NA**	8.17E-07
$f_{13}(x)$	8.28E-16	2.03E-06	4.87E-16	5.22E-05	**NA**
$f_{14}(x)$	2.76E-06	4.39E-17	6.91E-18	3.29E-04	**NA**

Table 4. P-values of the Wilcoxon Rank Sum Test (significance level 5%) for Test Function f_1 - f_{14} ($D = 100$)

Algorithm / Function	jDE	SaDE	JADE	DEahcSPX	*dn*-DADE
$f_1(x)$	2.97E-16	4.15E-17	6.88E-19	5.72E-10	**NA**
$f_2(x)$	5.41E-13	6.95E-20	1.37E-20	4.84E-12	**NA**
$f_3(x)$	5.89E-09	4.37E-08	2.17E-05	3.64E-09	**NA**
$f_4(x)$	1.07E-07	9.18E-05	9.78E-07	6.47E-05	**NA**
$f_5(x)$	2.68E-04	5.39E-06	5.02E-07	6.85E-07	**NA**
$f_6(x)$	5.14E-10	4.28E-07	8.96E-14	2.28E-08	**NA**
$f_7(x)$	8.42E-13	3.14E-07	6.11E-05	7.87E-11	**NA**
$f_8(x)$	1.58E-07	5.76E-11	**NA**	3.46E-07	6.73E-12
$f_9(x)$	7.82E-11	5.68E-17	1.43E-19	2.83E-14	**NA**
$f_{10}(x)$	2.49E-15	5.96E-11	3.17E-17	6.71E-12	**NA**
$f_{11}(x)$	5.77E-06	4.03E-20	1.89E-13	8.74E-05	**NA**
$f_{12}(x)$	4.26E-17	**NA**	2.68E-14	1.56E-10	5.13E-13
$f_{13}(x)$	8.82E-20	1.09E-10	**NA**	5.29E-05	7.36E-15
$f_{14}(x)$	4.91E-09	2.78E-13	3.19E-17	6.59E-11	**NA**

Table 1-2 indicate that the performance of *dn*-DADE is obviously superior to other competitor algorithms on majority of test functions in both 30-dimensional and 100-dimensional problems. It can be seen from Table 1/ 3 that: for 30-dimensional problems, the *dn*-DADE achieves the best performance in 13 functions (especially in function f_1 and f_9), and only for f_{12} is worse than DEahcSPX and SaDE. In functions f_7 and f_8, the performance of *dn*-DADE remained comparable to that of JADE, but the former achieved the lowest standard deviation due to its better stability. And the experimental results in Table 2/ 4 show that: for 100-dimension problems, *dn*-DADE still maintained statistically superior performance compared to the other four algorithms in 11 functions, and it remain second best for f_{12} being outperformed by SaDE, as well as for f_8 and f_{13} being outperformed by JADE. Thus, *dn*-DADE algorithm has shown its advantage in terms of handling the rotation, noisy functions and even the hybrid composition functions.

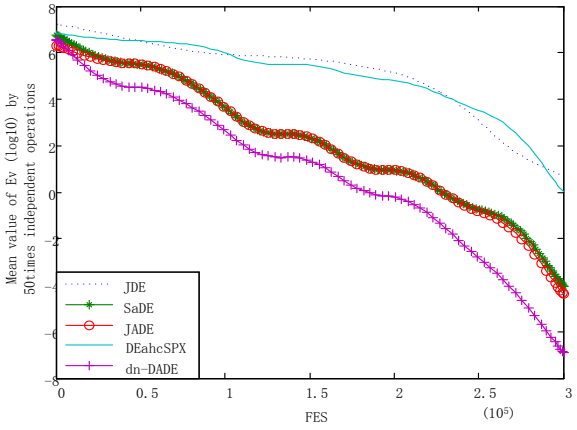


Figure 3. Convergence Graph of 5 Algorithms for f_4 ($D = 30$)

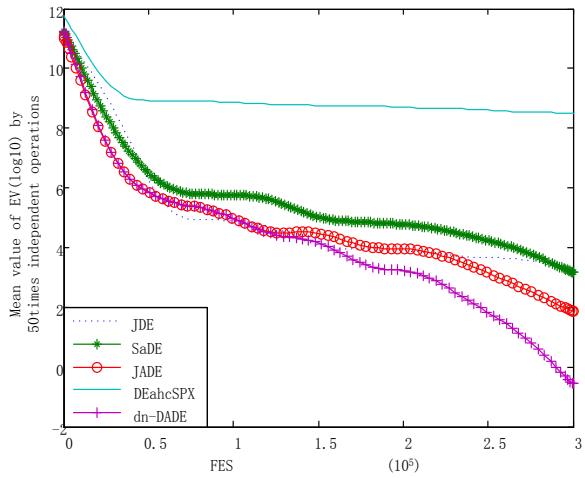


Figure 4. Convergence Graph of 5 Algorithms for f_6 ($D = 30$)

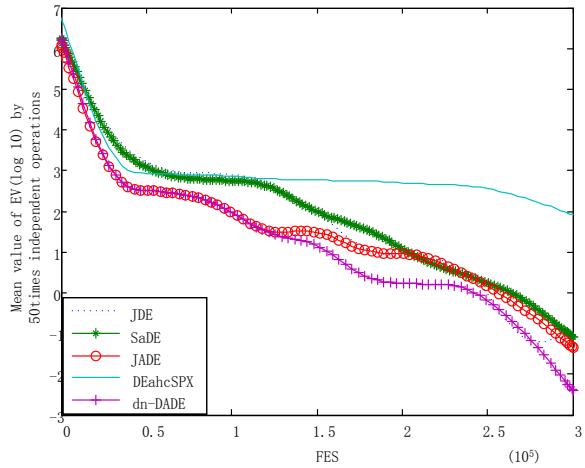


Figure 5. Convergence Graph of 5 Algorithms for f_7 ($D = 30$)

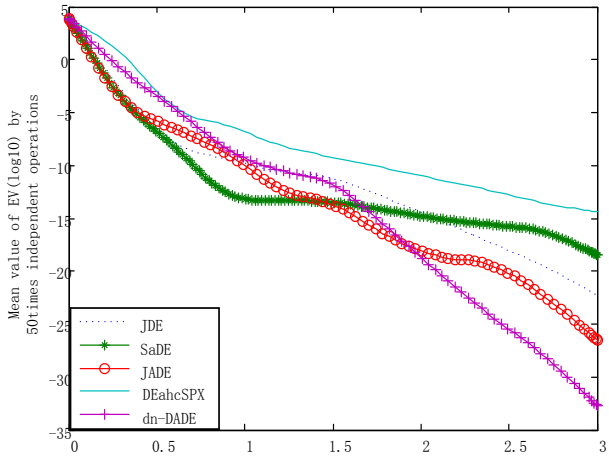


Figure 6. Convergence Graph of 5 Algorithms for f_9 ($D = 30$)

Figure 3-6 present the convergence characteristics of the best error value of the median run of five algorithms for 30-dimensiona function f_4 , f_6 , f_7 and f_9 . Clearly, for unimodal function f_4 , *dn*-DADE has significantly advantages on the convergence speed compared to other algorithms. In the cases of multimodal function f_6 and f_7 , the convergence speed of *dn*-DADE and JADE is faster than the other three algorithms due to their greedy mutation strategy. It can be observed that *dn*-DADE and JADE converge at the quickest rate among all the algorithms in early evolution stages. But with the increase of the evolution generation, *dn*-DADE speeds up remarkably because of its adaptive parameters and variation strategies. Note that for functions f_9 , *dn*-DADE has slow convergence rate in the initial stage of evolution, while at last it overtakes other competitor algorithms and attains greater final accuracy.

4.4. The Benefit of *dn*-DADE Components

The *dn*-DADE algorithm modifies the standard DE algorithm from two aspects as the mutation strategy and parameter setting by the dynamic adaptive manner. To identify the benefit of the improvement, we consider the following variants of *dn*-DADE for comparison:1) *dn*-DE: use the new mutation strategy with no adaptive parameter control (set $F_{dn}=0.7$, $CR_{dn}=0.5$, $\sigma_{dn}^2=0.01$ throughout the evolution process); 2) DADE: adopt the new parameter adaptation schemes with the mutation strategy DE/rand /1; 3) *dnc*-DADE: set *dn* as a constant $NP/4$, and the other settings are the same with *dn*-DADE. The mean and the standard deviation for 50 independent runs on 14 functions ($D=30$) are shown in Table 5.

Table 5. Mean Value and Standard Deviation of EV of 4 algorithms on Functions f_1 - f_{14} ($D=30$)

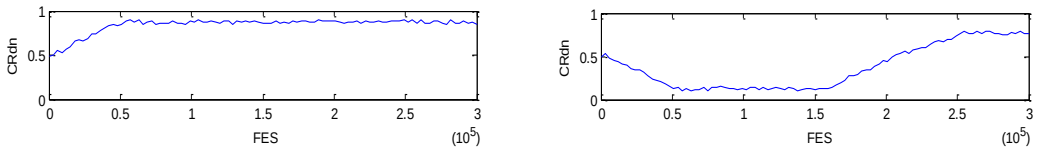
Algorithm / Function	***dn*-DE**	**DADE**	***dnc*-DADE**	***dn*-DADE**
	Mean value (standard deviation**)**	**Mean value** (standard deviation**)**	**Mean value** (standard deviation**)**	**Mean value** (standard deviation**)**
$f_1(x)$	2.19E-34 (4.83E-39)	5.07E-37 (7.83E-35)	3.58E-43 (6.78E-46)	**7.25E-58 (1.83E-60)**
$f_2(x)$	4.71E-12 (6.24E-13)	8.05E-12 (2.76E-12)	1.49E-15 (6.67E-17)	**5.17E-26 (4.64E-26)**
$f_3(x)$	4.47E+04 (2.19E+04)	3.68E+04 (1.74E+05)	6.92E+03 (8.74E+03)	**2.05E+03 (8.28E+03)**
$f_4(x)$	9.74E-03 (5.13E-05)	8.16E-04 (1.98E-03)	5.62E-05 (4.11E-06)	**1.28E-07 (7.02E-08)**
$f_5(x)$	4.95E+03 (5.12E+02)	3.28E+03 (2.91E+03)	3.15E+03 (7.64E+02)	**1.71E+02 (2.69E+02)**
$f_6(x)$	8.67E+02 (2.81E+01)	4.39E+01 (7.64E+00)	6.17E+00 (2.36E+00)	**2.72E-01 (8.14E-01)**
$f_7(x)$	5.87E-02 (4.96E-02)	4.91E-02 (4.83E-02)	7.16E-03 (3.89E-03)	**4.06E-03 (2.59E-03)**
$f_8(x)$	2.28E+01 (4.51E-01)	2.34E+01 (3.89E-01)	2.11E+01 (7.89E-02)	**2.01E+01 (1.88E-02)**
$f_9(x)$	8.27E-05 (9.24E-07)	5.51E-08 (3.47E-08)	7.49E-10 (7.16E-11)	**2.04E-12 (5.01E-14)**
$f_{10}(x)$	1.86E+02 (5.17E+01)	7.36E+01 (2.78E+01)	6.64E+01 (1.96E+01)	**3.97E+01 (2.61E+00)**
$f_{11}(x)$	1.59E+01 (1.28E+01)	2.01E+01 (9.57E+00)	1.57E+01 (1.06E+01)	**1.45E+01 (5.84E+00)**
$f_{12}(x)$	3.57E+05 (2.18E+04)	6.28E+04 (3.88E+03)	1.64E+04 (5.91E+03)	**2.37E+03 (1.05E+03)**

$f_{13}(x)$	5.91E+03 (2.87E+03)	3.24E+03 (7.18E+02)	2.47E+01 (1.82E+00)	**2.13E+00** **(3.82E-02)**
$f_{14}(x)$	9.61E+01 (7.28E+00)	9.79E+01 (3.54E+01)	9.38E+01 (3.96E+00)	**1.54E+01** **(2.03E-01)**

The results in Table 5 indicate that *dn*-DADE algorithm is significantly better than its three variants. Therefore, we can conclude that a reciprocal cooperation between the parameter adaptation and the mutation strategy *DE/current-to dnbest/1*. This should be because the new mutation strategy can balance the exploration and exploitation ability of the algorithm in evolution process, and also guide the progress direction by elite solutions to avoid blind search. Although the changes of *dn* are more inclined to "greedy" strategy in the later period and may lead to premature convergence because of the diversity decrease, the parameter adaption scheme is proposed to effectively improve the population's diversity and the convergence rate in the promising direction.

4.5. Parameter Analysis

The performance of DE algorithm is very sensitive to the control parameters *F* and *CR*. Different optimization problems need different parameter settings. In order to achieve satisfactory performance without any users' prior knowledge, we propose a new adaptive mechanism to automatically update the control parameters in this paper. Since the value of crossover probability factor is related to the characteristic of practical problems, we select two 30-dimensional functions with different natures to observe the adaptive performance of CR_{dn}. Note that f_6 represents the non-separable problem and f_9 represents the separable problem. Figure 7 shows the self-adaption curves of CR_{dn} for function f_6 and f_9. It can be seen, for different problems, CR_{dn} show different trend in different search stages. This further illustrates the effectiveness and efficiency of parameter adaptation.



(a) f_6 (b) f_9

Figure 7. Self-adaption Curves of CR_{dn}

In addition, the adjustment value $\theta r(\theta = 1, 2, 3\ldots\ r = 0.05)$ is proposed to avoid $F_{i,G}$ drop out of the limited range as far as possible according to the characteristics of Cauchy distribution. The exceeding probability of $F_{i,G}$ is only related to the initial value of F_{dn} and θ. In case of the initial value of F_{dn} set as 0.7, we select $\theta = 2$ to balance the range of F_{dn} and the exceeding probability.

Table 6. The Relationship between θ and the Exceeding Probability of $F_{i,G}$

θ	exceeding probability
$\theta = 1$	25%
$\theta = 2$	15%
$\theta = 3$	10%
$\theta = 4$	8%

5. Conclusions

In this paper, we propose an improved differential evolution algorithm based on dynamic adaptive strategies and parameters, namely *dn*-DADE, which incorporates the mutation strategy DE/current-to-*dn*best/1 and parameter adaptation schemes. In the mutation strategy, the information of elite solutions, whose number varies dynamically, is utilized to guide the search direction, as well as balance the diversity of population and the greediness of the mutation. And the parameter adaptation strategies further improve the stability and robustness of the algorithm.

The *dn*-DADE algorithm is compared with four other state-of-the-art DE algorithms and three its variants over 14 benchmarks functions provided by CEC2005 special session. The experimental results indicate that its overall performance is better than the other competitors, and it has remarkable advantages in terms of the quality of the final solutions, convergence speed and robustness. In addition, the effectiveness of adaptive control parameter settings is experimentally studied.

Acknowledgements

This work was supported by Innovation Program of Shanghai Municipal Education Commission (12ZZ158) and National Science Foundation "Research of Container Terminal Multi-stage Logistics System Operations and Cost Control" (2014-2016,71301101).

References

[1] R. Storn and K. Price, "Differential Evolution-A simple and efficient heuristic for global optimization over continuous spaces", Journal of Global Optimization, vol. 4, no. 11, **(1997)**, pp. 341-359.
[2] R. Storn and K. Price, "Differential Evolution-A practical approach to global optimization", **(2006)**, pp. 133-152.
[3] S. Das, A. Abraham and A. Konar, "Automatic clustering using an improved differential evolution algorithm", IEEE Transaction on Systems, Man and Cybernetics, vol. 1, no. 38, **(2008)**, pp. 218-236.
[4] M. Han, M. H. Wang and J. C. Fan, "Trajectory optimization based on improved differential evolution algorithm", Control and Decision, vol. 2, no. 27, **(2012)**, pp. 247-251.
[5] S. Das and A. Abraham, "Differential evolution using a neighborhood-based mutation operator", IEEE Trans on Evolutionary Computation, vol. 3, no. 13, **(2009)**, pp. 526-553.
[6] J. G. Yuan, Z. G. Sun and G. J. Qu, "Simulation Study of Differential Evolution", Journal of System Simulation, vol. 20, no. 19, **(2007)**, pp. 4646-4648.
[7] A. K. Qin, V. L. Huang and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization", IEEE Transaction on Evolutionary Computation, vol. 2, no. 13, **(2009)**, pp. 398-417.
[8] J. Brest, S. Greiner and B. Boskovie, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems", IEEE Transactions on Evolutionary Computation, vol. 6, no. 11, **(2006)**, pp. 646-657.
[9] R. Mallipeddi, P. Suganthana and Q. Pan, "Differential evolution algorithm with ensemble of parameters and mutation strategies" IEEE Transactions on Evolutionary Computation, vol. 11, **(2011)**, pp. 1679-1696.
[10] A. Salman, A. P. Engelbrecht and M. G. H. Omran, "Empirical analysis of self-adaptive differential evolution", European Journal of Operational Research, vol. 2, no. 183, **(2007)**, pp. 785-804.
[11] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive", IEEE Transaction on Evolutionary Computation, vol. 5, no. 13, **(2009)**, pp. 945-958.
[12] E. Mezura-Montes, J. Velázquez-Reyes and C. A. C. Coello, "A comparative study of differential evolution variants for global optimization", Proceedings of Genetic Evolutionary Computation Conference (GECCO-2006), **(2006)**, pp. 485–492.
[13] P. N. Suganthan, N. Hansen and J. J. Liang, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization", Nanyang Technological University, Singapore, **(2005)**.
[14] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search", IEEE Transaction on Evolutionary Computation, vol. 1, no. 12, **(2008)**, pp. 107-125.

[15] J. Derrac, S. García, D. Molin and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms", Swarm Evolutionary Computation, vol.1, no. 1, **(2011)**, pp. 3-18.

Author

Congjiao Wang (Jiangsu, China, 1984-11), Doctoral student. Major: Electrical Engineering and Automation, Shanghai Maritime University.

Her research interests include artificial intelligence, evolutionary algorithms, differential evolution and real-world applications in complex systems. She has published over 10 papers in journals and conference proceedings.