SDE: A Stochastic Coding Differential Evolution for Global Optimization

Jing-hui Zhong and Jun Zhang

Department of Computer Science
Sun Yat-sen University
P.R. China

Key Laboratory of Digital Life
Ministry of Education
P.R. China

Key Laboratory of Software Technology
Education Dept. of Guangdong Province
P.R. China

junzhang@ieee.org

ABSTRACT

Differential Evolution (DE) is a new paradigm of evolutionary algorithm (EA) which has been widely used to solve nonlinear and complex problems. The performance of DE is mainly dependent on the parameter settings, which relate to not only characteristics of the specific problem but also the evolution state of the algorithm. Hence, determining the suitable parameter settings of DE is a promising but challenging task. This paper presents an enhanced algorithm, namely, the stochastic coding differential evolution (SDE), to improve the robustness and efficiency of DE. Instead of encoding each individual as a vector of floating point numbers, the proposed SDE represents each individual by a multivariate normal distribution. In this way, individuals in the population can be more sensible to their surrounding regions and the algorithm can explore the search space region-by-region. In the SDE, a newly designed update operator and a random mutation operator are incorporated to improve the algorithm performance. Traditional DE operators such as the mutation scheme and the crossover operator are also accordingly extended. The proposed SDE has been validated by nine benchmark test functions with different characteristics. Four highly regarded EAs are compared in the experiment study. The comparison results demonstrate the effectiveness and efficiency of the SDE.

Catergories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]:Problem Solving, Control Methods and search-Heuristic methods; G.1.6 [Numerical Analysis]: Optimization-Global optimization

Keywords

Differential Evolution, Evolutionary Computation, Global Optimization, Multivariate Normal Distribution, Stochastic Coding

1. INTRODUCTION

Differential evolution (DE) is a new paradigm of evolutionary algorithm (EA) for continuous optimization [1]. Due to its easy implementation, strong global search ability and fast convergence speed, DE has become one of the most popular optimization techniques and has been successfully applied to a wide range of applications such as system design [2] and space trajectory optimization [3] [4].

The performance of DE is largely dependent on its two parameters, namely, the mutation scale factor *F* and the crossover rate *CR*. However, determining the ideal values of *F* and *CR* is a difficult task, because their optimal values are not only dependent on the characteristics of the specific problem but also related to the evolution state of the algorithm [5]. Although several adaptive parameter control strategies have been drawn [5]-[6], developing a more effective and efficient DE for practical applications is one of the significant and challenging research topics in the EA community.

In most DE algorithms, each individual is encoded by a vector of floating numbers, representing one possible solution in the search space. Hence, they actually explore the search space in a point-by-point manner. This search mechanism could be quite inefficient because many computational efforts are wasted on evaluating un-significant regions. On the other hand, the surrounding regions of promising solutions are more likely to contain better solutions, but they are not explored sufficiently. To overcome the above drawbacks, a new coding strategy named stochastic coding strategy has been proposed in [8]-[10] recently. The key idea is to represent each individual by a stochastic region defined by a normal distribution, so that individuals are more sensible to their surrounding region, and the search space can be explored region-by-region. This profound idea has shown great potential to improve the performance of EAs [8][10].

Existing stochastic coding strategies focus on using a one-dimensional normal distribution to represent one variable of the problem, i.e., each variable is associated with a mean value and an independent variance. Inspired by the fact that multivariable normal distribution is more effective to capture the interactions between variables and can be coordinate system invariant [13]-[15], this paper proposes an enhanced stochastic coding strategy based on multivariable normal distribution. This new coding strategy is incorporated into DE and form a stochastic coding DE (SDE) for solving continuous optimization. In the SDE, a newly designed update operator and a random mutation operator are incorporated into the algorithm framework to improve the performance. Traditional DE operators (e.g., mutation and crossover) are also accordingly extended. The proposed SDE will be assessed by carrying out optimization on nine benchmark functions with different characteristics. Four highly regarded optimization methods, i.e., CLPSO [11], DE [1], CoDE[12], and CMA-ES[13] will be used for comparison.

The rest of the paper is organized as follows. Section 2 briefly describes the general framework of DE and its recent developments. Section 3 describes detailed implementations of SDE. Section 4 presents the experiment study and compares the performance of six optimization algorithms. At last, Section 5 draws the conclusions.

2. TRADITIONAL DE AND ITS DEVELOPMENTS

DE is a population based algorithm which is proposed by Storn and Price [1]. It starts with a set of random individuals. Then these individuals are evolved iteratively using mutation, crossover, and selection operators until meeting the termination condition. In each iteration, the mutation operator is firstly used to create a mutation vector for each individual, i.e.,

$$Y_{i,g} = X_{r1,g} + F \cdot (X_{r2,g} - X_{r3,g}) \tag{1}$$

where $r_1, r_2, r_3 \in [1, NP]$ are three distinct random integers, F is the scaling factor, g represents the current generation and NP is the population size. Following the mutation operator, the crossover operator generates a trial vector $U_{i,g}$ for each individual by

$$u_{i,g}^{j} = \begin{cases} y_{i,g}^{j}, & \text{if } rand(0,1) < CR \text{ or } j = k \\ x_{i,g}^{j}, & \text{otherwise} \end{cases} \tag{2}$$

where $CR \in [0,1]$ is the crossover rate, k is a random integer within [1, D], D is the problem dimension, $rand$ (0,1) returns a random number uniformly distributed between 0 and 1, $u_{i,g}^{j}$ represents the jth variable of $U_{i,g}$. Thirdly, the selection operator chooses NP individuals for the next iteration by

$$X_{i,g} = \begin{cases} U_{i,g}, & \text{if } U_{i,g} \text{ is better than or equal to } X_{i,g} \\ X_{i,g}, & \text{otherwise} \end{cases} \tag{3}$$

In DE algorithm, the values of F and CR have significant influence on the behavior of the algorithm. The ideal parameter settings of DE seem to be problem dependent [5]-[7]. Traditional trial-and-error method for adjusting the values of F and CR is inconvenient in practice, therefore adaptive control techniques have been utilized recently. In [5], Brest *et al.* proposed a self-adaptive DE, where the values of F and CR are either inherited from parents or a randomly generated number. Qin *et al.* [7] proposed a self-adaptive DE by controlling the mutation schemes and parameters dynamically based on previous search experience. Zhang and Sanderson [6] incorporated a new "current-to-*p*best" mutation scheme into DE and suggested sampling F and CR from a normal distribution and a Cauchy distribution respectively. In order to improve the robustness and efficiency of DE, Wang *et al.* [12] suggested to randomly combine three mutation schemes and three parameter settings to generate offspring in the evolution.

3. THE PROPOSED SDE

3.1 Individual Representation

Unlike traditional DE that encodes each individual as a vector of floating-point numbers, the proposed SDE encodes each individual with a multivariable normal distribution, as expressed in Eq. (4).

$$P_{i,g} = [\mu_{i,g}, \Sigma_{i,g}] = \left[\left[\mu_{i,g}^{1}, \mu_{i,g}^{2}, \ldots, \mu_{i,g}^{n} \right], \begin{bmatrix} \sigma_{i,g}^{11} & \sigma_{i,g}^{12} & \ldots & \sigma_{i,g}^{1n} \\ \sigma_{i,g}^{21} & \ldots & \ldots & \sigma_{i,g}^{2n} \\ \ldots & \ldots & \ldots & \ldots \\ \sigma_{i,g}^{n1} & \ldots & \ldots & \sigma_{i,g}^{nn} \end{bmatrix} \right] \tag{4}$$

where $\mu_{i,t}$ represents the $n \times 1$ mean vector, $\Sigma_{i,t}$ represents the $n \times n$ covariance matrix, i represents the index of the individual, g represents the generation and n is the dimension of the problem. The mean vector μ is used to evaluate the fitness of the individual. Based on the stochastic coding strategy, the normal distribution of each individual can be used to sample neighboring individuals for local fine-tuning.

3.2 Algorithm Framework

```
Algorithm 1: SDE
1.  Initialize algorithm parameters and a random population P_1            step 1
2.  g=1
3.  While (termination criterion not met) do
4.      Sort P_g in best to worst order
5.      For i =1 to M do
6.          $\bar{\mu}_{i,g} = (1-\alpha)\cdot\mu_{i,g} + \alpha\cdot\mu_{best,g}$
7.          Get the eigen values (e_1,...e_n), and eigen vectors (v_1,...v_n) of $\Sigma_{i,g}$
8.          For j = 1 to n do
9.              set b = the projection of $\mu_{best,g}^{j}$ in v_j
10.              set m = the projection of $\mu_{i,g}^{j}$ in v_j
11.             If $e_j/(b-m)^2 < Q$ then set $e_j = (b-m)^2 * Q$
12.         EndFor
13.         Set $X = [v_1,...v_n]*([e_1,...e_n]*\mathbf{I})*[v_1,...v_n]^T$.
14.         Apply the Cholesky decomposition to X to obtain S, so that $X=SS^T$
15.         For j = 1 to N do
16.             $U_j = \mu_j + S\cdot Z$
17.             Set k = floor(rand(0,1) * n)                                  step 2
18.             For m = 1 to n do
19.                 If m == k then $T_j^m = U_j^m$
20.                 Else $T_j^m = \mu_{i,g}^m$
21.             EndFor
22.         EndFor
23.         Set B = the set of neighboring solutions which are better than $P_{i,g}$
24.         If $B \neq \phi$ then set $\Sigma_{i,g}$= the covariance of B
25.         If the best neighboring solution is better than $P_{i,g}$ then
26.             Set $\mu_{i,g}$ = the best neighboring solution
27.         EndIF
28.     Endfor
29.     For i =1 to NP do
30.         Set F = rand(0,1), Set CR = rand(0,1)
31.         Randomly choose $A_{r1,g}$ from the best p.NP individuals of P_g
32.         Randomly choose $A_{r2,g} \neq A_{r1,g}$ from P_g
33.         Randomly choose $A_{r3,g} \neq A_{r2,g} \neq A_{r1,g}$ from P_g
34.         Set k = floor(rand(0,1) * n)
35.           For j = 1 to n do
36.             If rand(0,1) ≥ CR and j ≠ k then                              step 3
37.                 set $u_{i,g}^j = a_{i,g}^j + F\times(a_{r_1,g}^j - a_{i,g}^j) + F\times(a_{r_2,g}^j - a_{r_3,g}^j)$
38.             Else set $u_{i,g}^j = a_{i,g}^j$
39.             If rand(0,1) < pm then set $u_{i,g}^j$ = rand(LB_j, UB_j) End if
40.             If < LB_j or > UB_j then set $u_{i,g}^j$ = rand(LB_j, UB_j) End if
41.         End for
42.         If $u_{i,g}$ is better than $P_{i,g}$ then set $P_{i,g}$ = $u_{i,g}$ and update $\Sigma_{i,g}$ of $P_{i,g}$
43.     End for
44.     g = g+1
45. End while
```

Figure1. Algorithm framework of SDE.

Fig. 1 shows the framework of SDE, where three steps are involved during the evolution. The detailed implementations of these three steps are described as follows.

1) Step 1 - Initialization

The initialization generates a set of random individuals. For each initial individual, the mean vector is generated by

$$\mu_{i,1}^k = rand(LB_k, UB_k), \forall i \in [1, NP], \forall k \in [1, n] \quad (5)$$

where LB_k and UB_k are the lower and upper bounds of the kth variable, $rand(a,b)$ returns a random number uniformly distributed between a and b. The covariance matrix is empirically initialized as

$$\sigma_{i,1}^{jk} = \begin{cases} 0, \text{if } j \neq k \\ c_k, \text{ otherwise} \end{cases} \quad (6)$$

where c_k is a predefined constant, e.g., $c_k = (UB_k - LB_k)^2$. After generating the mean vectors and the covariance matrix of an individual, its fitness value is evaluated.

2) Step 2 – Update Operator

The update operator is used for local fine-tuning, as well as for updating the normal distribution of individuals in the population. For each given individual, a two-phased process is carried out. The first phase is to sample N neighboring solutions. N temporary solutions are sampled by

$$\mu_{i,g}^{'} = \mu_{i,g} + S \cdot Z \quad (7)$$

where S is a lower triangular matrix with $\sum_{i,g} = S \cdot S^T$, Z is a $n \times 1$ matrix with each element sampled from a standard Normal distribution. It can be verified by Eq. (8) and Eq. (9) that the mean and the covariance of $\mu_{i,g}^{'}$ are equal to $\mu_{i,g}$ and $\sum_{i,g}$ respectively.

$$E(\mu_{i,g}^{'}) = E(\mu_{i,g} + S \cdot Z) = \mu_{i,g} + S \cdot E(Z) = \mu_{i,g} \quad (8)$$

$$\begin{aligned} \text{cov}(\mu_{i,g}^{'}) &= (\mu_{i,g}^{'} - \mu_{i,g})(\mu_{i,g}^{'} - \mu_{i,g})^T \\ &= (\mu_{i,g} + S \cdot Z - \mu_{i,g})(\mu_{i,g} + S \cdot Z - \mu_{i,g})^T \\ &= (S \cdot Z)(S \cdot Z)^T \\ &= S \cdot (Z \cdot Z^T) \cdot S \\ &= S \cdot 1 \cdot S^T \\ &= \sum_{i,g} \end{aligned} \quad (9)$$

As suggested in [15][16], two parameters, namely, α and Q, are used to improve the algorithm performance. Specifically, when sampling a temporary individual, the mean vector is moved towards the best-so-far individual by

$$\bar{\mu}_{i,g} = (1 - \alpha) \cdot \mu_{i,g} + \alpha \cdot \mu_{best,g} \quad (10)$$

where $\mu_{best,g}$ is the mean vector of the best-so-far individual. The threshold Q is used to enlarge the eigenvalues for maintaining the population diversity [15]. Once a temporary solution U_j is sampled, the corresponding neighboring solution is generated by

$$T_j^m = \begin{cases} U_j^m, \text{ if } m = k \\ \mu_{i,g}^m, \text{ otherwise} \end{cases} \quad (11)$$

where k is a random integer between[1,n] is a predefined constant. In this way, only one variable of the neighboring solution is different from that of the given individual. The quality of the given individual can be improved gradually in a manner similar to the hill climb search mechanism.

The second phase is to update the normal distribution of the given individual. Denote ***B*** as the set of neighboring solutions which are better than the given individual. If ***B*** is not empty, the covariance of the given individual is set to be the covariance of ***B***. Meanwhile, if the best neighboring solution is better than the given individual, the mean vector of the given individual would be replaced by the best neighboring solution.

Since performing the update operator on a given individual requires evaluating the fitness of N sampled individuals, it needs much computational cost to update all individuals at each generation. In order to reduce the computational cost, the SDE only chooses the top M individuals to undergo the update process.

3) Step 3 –Mutation, Crossover and Selection

This step applies the DE operators to generate NP new individuals. For each individual $P_{i,t}$, a new individual is generated by

$$\mu_{i,g}^{'} = \begin{cases} \mu_{i,g} + F(\mu_{r1,g} - \mu_{i,g}) + F(\mu_{r2,g} - \mu_{r3,g}), \text{if } rand(0,1) < CR \\ \mu_{i,g}, \quad \text{otherwise} \end{cases} \quad (12)$$

$$\sigma_{i,g}^{uv}{'} = \begin{cases} 0, \quad \text{if } u \neq v \\ \left(\mu_{r2,g}^u - \mu_{r3,g}^u\right)^2, \text{ otherwise} \end{cases} \quad (13)$$

where r_1, r_2, r_3 with $r_1 \neq r_2 \neq r_3$, are three random individual indexes. Using Eq. (13), we can set the covariance matrix according to the evolution state. It should be noticed that the new vectors $\mu_{i,g}^{'}$ is bounded by the search ranges, i.e.,

$$\mu_{i,g}^k = rand(LB_k, UB_k), \text{ if } \mu_{i,g}^k < LB_k \text{ or } \mu_{i,g}^k > UB_k \quad (14)$$

In the DE algorithm, the optimal values of F and CR are dependent on the specific problem and the evolution state. However, in practical applications, the characteristics of the problem at hand are usually unknown. In order to improve the robustness of the algorithm, we adopt a random scheme to set values of F and CR, i.e.,

$$F = rand\ (0, 1) \quad (15)$$

$$CR = rand\ (0, 1) \quad (16)$$

Followed by the crossover operator, an extra random mutation operator is utilized to change each $\mu_{i,g}^k$ with a probability of pm, i.e.,

$$\mu_{i,g}^k = rand(LB_k, UB_k), \text{ if } rand(0,1) < pm \quad (17)$$

We introduce this random mutation to improve the population diversity and avoid premature convergence, because the update operator in *step2* would gradually drive the population towards the best individual and resulting in the population losing diversity.

There is a repetition from *Step2* to *Step3* and the evolution processes iteratively until reaching the maximum number of evaluations.

4. EXPERIMENTS AND COMPARISONS

4.1 Experimental Settings

In this section, nine benchmark functions with different characteristics are used to investigate the effectiveness of the proposed SDE. The benchmark functions are listed in Table I, where f_1, f_2 are unimodal functions, while the others are multimodal functions. The performance of SDE will be compared with four EAs, i.e., CLPSO[11], DE[1], CoDE[12], and CMA-ES[13]. The parameters of all compared EAs are set according to their referenced papers, as listed in Table II. The dimension of all test functions is 30 and the maximum number of fitness evaluations is 300000. Since EAs are stochastic algorithms that may obtain different results in different runs, all compared EAs are run for 30 independent times on each test case. All algorithms

Table I. Benchmark Test Functions.

Name	Function	n	Domain	$f_{\min}$
Sphere	$f_1(x)=\sum_{i=1}^{n}x_i^2$	30	[-100,100]	0
Schwefel	$f_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	[-10,10]	0
Rosenbrock	$f_3(x)=\sum_{i=1}^{n}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2$	30	[-30, 30]	0
Schwefel	$f_4(x)=\sum_{i=1}^{n}-x_i\sin(\sqrt{\lvert x_i\rvert})$	30	[-500,500]	-12569.5
Rastrigin	$f_5(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10)]$	30	[-5.12,5.12]	0
Ackley	$f_6(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i)+20+e$	30	[-32,32]	0
Penalized	$f_7(x)=\frac{\pi}{n}\{10\sin^2(\pi y_i)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]$ $+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)$ where $y_i=1+\frac{1}{4}(x_i+1)$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a\le x_i\le a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$	30	[-50,50]	0
Penalized	$f_8(x)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]$ $+(x_n-1)[1+\sin^2(2\pi x_n)]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	30	[-50,50]	0
Griewank	$f_9(x)=\sum_{i=1}^{n}(\frac{x_i^2}{4000})-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	30	[-600,600]	0

Table II. Parameter Settings of Compared EAs.

Algorithms	Parameter settings
DE	NP = 100, F = 0.5, CR = 0.9
CLPSO	NP = 40, w_0 = 0.9, w_1 = 0.4, c = 1.49445
CoDE	NP = 30, {F, CR}={{1.0, 0.1}, {1.0, 0.9}, {0.8, 0.2}}, mutation operator = {"rand/1/bin", "rand/2/bin", "current-to-rand/1" }
CMA-ES	$\mu=\lfloor\lambda/2\rfloor$, $\lambda=4+\lfloor 3\ln(n)\rfloor$
SDE	NP=50, M=6, N=5, pm = 0.01, p= 0.3, α = 0.15, Q = 2

Table III. Experimental Results of Five EAs on the Benchmark Test Functions.

Function		CLPSO	DE	CoDE	CMA-ES	SDE
f_1	*mean*	1.17×10^{-20}‡	2.23×10^{-31}‡	5.17×10^{-71}	**0**†	6.15×10^{-134}
f_2	*mean*	2.51×10^{-13}‡	2.28×10^{-15}‡	2.47×10^{-37}‡	5.92×10^{-17}	$\mathbf{3.89\times10^{-69}}$
f_3	*mean*	5.70‡	1.98‡	3.60×10^{-12}‡	1.33×10^{-1}	$\mathbf{1.78\times10^{-16}}$
f_4	*mean*	**-12569.5**‡	-8384.92‡	**-12569.5**‡	-8360.13‡	**12569.5**
f_5	*mean*	2.74×10^{-11}‡	1.40×10^{2}‡	**0**‡	5.23×10^{1}‡	**0**
f_6	*mean*	2.49×10^{-11}‡	4.62×10^{-15}‡	3.55×10^{-15}‡	$\mathbf{1.18\times10^{-16}}$†	3.55×10^{-15}
f_7	*mean*	5.47×10^{-22}‡	4.01×10^{-32}‡	$\mathbf{1.57\times10^{-32}}$‡	1.04×10^{-2}	$\mathbf{1.57\times10^{-32}}$
f_8	*mean*	7.29×10^{-21}‡	1.58×10^{-31}‡	$\mathbf{1.35\times10^{-32}}$‡	1.83×10^{-3}‡	$\mathbf{1.35\times10^{-32}}$
f_9	*mean*	4.86×10^{-13}‡	**0**‡	**0**‡	1.23×10^{-4}‡	**0**
Better		0	0	0	2	
Worse		9	9	8	4	
Similar		0	0	1	3	
Score		-9	-9	-8	-2	

†means the corresponding values are significantly better than those found by the SDE; ‡means the corresponding values are significantly worse than those found by the SDE; "*Better*" ("*Worse*" or "*Similar*") represents the times of the algorithm compared performing significantly better than (significantly worse than or similar to) the SDE. $Score=1\cdot Better+(-1)\cdot Worse+0\cdot Similar$.

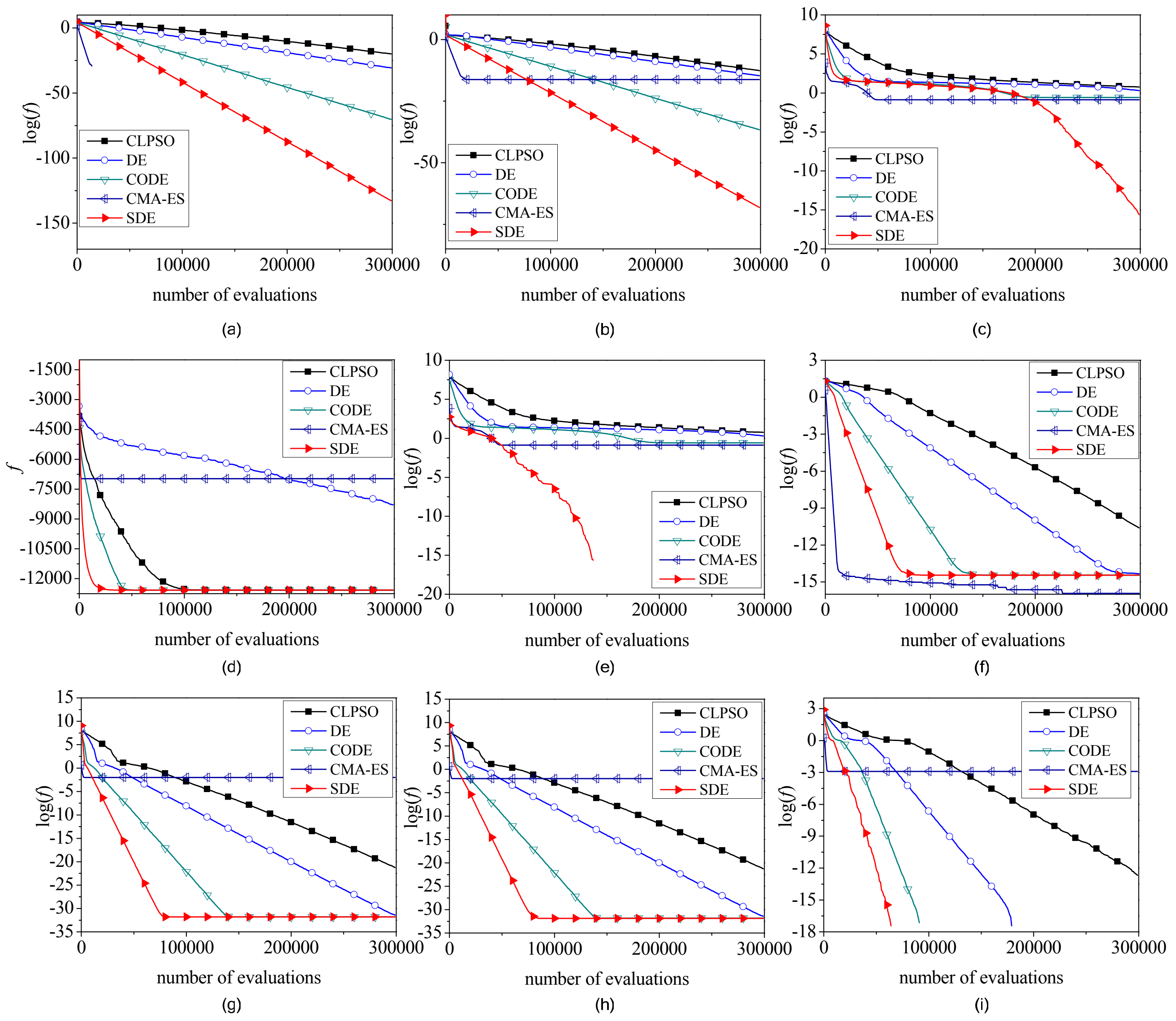

Fig. 2 Convergence graphs of compared algorithms ((a) f_1, (b) f_2, (c) f_3, (d) f_4, (e) f_5, (f) f_6, (g) f_7, (h) f_8, (i) f_9).

Intel (R) Core ™2 Quad CPU Q6600, 2.4 GHz, 1.96GB of RAM.

4.2 Performance Comparison

Table III lists the results of all algorithms compared. In Table III, we use a two-sample *t*-test to check whether an algorithm compared performs significantly different from the SDE. If the mean value of an algorithm compared is significantly better than (similar to or worse than) that of the SDE, the algorithm compared would gain a score of 1 (0, or -1). Otherwise, if the algorithm compared and the SDE find the global optima in 30 runs, we do the *t*-test on the smallest number of fitness evaluations (*FES*) required to find the optimal solution. If the *FES* of a algorithm compared is significantly smaller than (similar to or larger than) that of the SDE, it will gain a score of 1 (0, or -1). The results in Table III show that, SDE performs significantly better than CLPSO and DE on all test cases. SDE performs better than CoDE on $f_2, f_4, f_5, f_6, f_7, f_8$, and f_9, while performs competitive on other functions. Compared with CMA-ES, SDE performs significantly worse on f_1, f_6, competitive on f_2, f_3, f_7, and significantly better on f_4, f_5, f_8, f_9. According to the score values in the last row, SDE performs the best among all algorithms compared.

The average convergence speed of all algorithms compared is illustrated in Fig. 2. It can be seen in Fig.2 (a) that CMA-ES has a faster convergence speed than SDE, but CMA-ES quickly gets trapped into local optima on $f_2, f_3\, f_4\, f_5\, f_7\, f_8$ and f_9. Compared with other EAs, SDE has a much fast convergence speed on all test cases. The SDE seems to perform more stable than other EAs for it does not get trapped into local optima on the nine test cases. The above results indicate that SDE has strong global and local search ability.

In the following part of this subsection, we discuss the computational complexity of algorithms compared. We use the method presented in [17], which use four factors, namely *T0*, *T1*, $\hat{T}2$, and ($\hat{T}2$ -*T1*)/*T0* to reflect the complexity of an algorithm. The detailed computations of these four factors can be referred in [17]. In our experiments, the values of *T1* and *T2* are obtained on f_3 with D = 30 and the maximum fitness evaluations = 200000. The experiment results are list in Table IV, where all values are measured in time seconds. It can be observed that, the computational complexity of the proposed algorithm is much larger than the other algorithms compared. This is because the SDE requires M times of covariance matrix decomposition in each generation, which needs O(MD^3) time complexity. Therefore, reducing the time complexity of SDE can be a future work. Note that the convergence speed of SDE is much faster than those of the other algorithms in terms of number of fitness evaluation. Hence applying SDE to applications where the fitness evaluation process is very expensive seems to be a promising future work.

Table IV. Computational Complexity of Algorithms compared.

Algorithm	*T0*	*T1*	$\hat{T}2$	$(\hat{T}2 - T1)/T0$
CLPSO	0.253	1.489	2.370	3.479
DE	0.253	1.489	2.395	3.581
CODE	0.253	1.489	2.174	2.708
CMA-ES	0.253	1.489	7.479	23.676
SDE	0.253	1.489	23.979	88.893

4.3 Algorithm Analysis

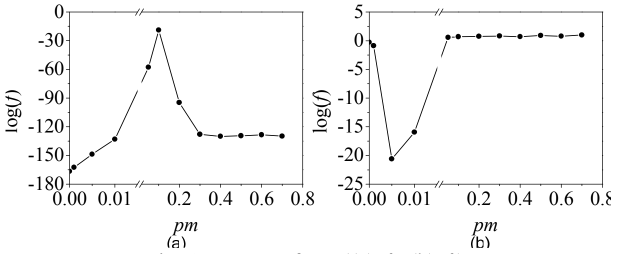

Fig. 3 Impact of *pm* ((a) f_1, (b) f_3).

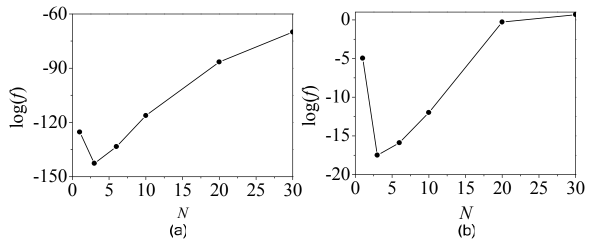

Fig. 4 Impact of *M* ((a) f_1, (b) f_3).

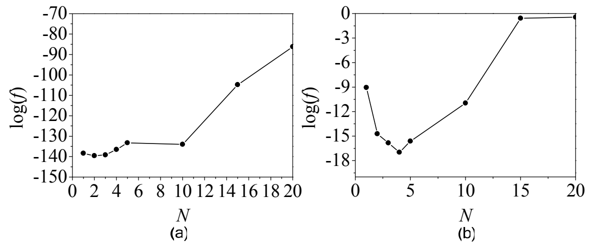

Fig. 5 Impact of *N* ((a) f_1, (b) f_3).

First, we study the impact of *pm*, which determines the random mutation rate. Generally, a large *pm* would improve the population diversity, but it would slow down the convergence at the mean time. In the following experiments, we apply SDE to a unimodal function (f_1) and a multimodal function (f_3) to investigate its performance, with the value of *pm* to be 0, 0.001, 0.005, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6 and 0.7. The other parameters are set the same as in Table II. The results in Fig. 3 demonstrate that a smaller *pm* (e.g., pm < 0.01) would leads to better results for f_1. However, as for the multimodal function f_3, SDE generally performs better as *pm* increases from 0 to 0.005, but its performance decreases as *pm* continuously increases.

Next, we study the impact of *M* by varying its value to be 1, 3, 6, 10, 20, and 30, while the other parameters are set the same as in Table II. The curves in Fig. 4 shows that *M* should not be set too small (e.g., M = 1) or too large (e.g., M = 30). It seems that setting M = 3 or M = 6 leads to a promising results.

Third, we study the impact of *N* by varying its value to be 1, 2, 3, 4, 5, 10, 15 and 20, while the other parameter settings are the same as in Table II. The curve in Fig. 5(a) shows that the performance of SDE on f_1 decreases as *N* increases. As for the multimodal function f_3, *N* should not be set too small (e.g., N = 1) or too large (e.g., N >10).

5. CONCLUSIONS

This paper has presented an enhanced differential evolution with stochastic coding strategy, namely, SDE, for global continuous optimization. In SDE, each individual is represented by a multivariate normal distribution. The multivariate normal distribution is used to evaluate the fitness of individuals as well as to sample neighboring individuals for local fine-tuning. A new update operator is designed based on the stochastic coding strategy for local fine-tuning. The DE operators, namely, mutation and crossover, are accordingly extended to generate offspring. The proposed SDE has been validated by nine benchmark functions with differential characteristics. Four highly regarded EAs, including CLPSO, CoDE and CMA-ES, have been used for comparison. The experimental results demonstrate that the SDE offers a very promising performance. Future research work includes the following: 1) reducing the time complexity of SDE, 2) applying SDE to applications where the fitness evaluation process is very expensive and time consuming (e.g., grid workflow scheduling [18] and power electronic design[19]), 3) adopting adaptive parameter controlling strategies [20]-[22] to further improve the performance of SDE, and 4) extending the proposed coding strategy to improve the performance of other algorithms such as particle swarm optimization [23] and ant colony optimization [24].

6. ACKNOWLEDGMENTS

This work was supported in part by the National Science Fund for Distinguished Young Scholars No.61125205, National Natural Science Foundation of China No.61070004 and NSFC Joint Fund with Guangdong under Key Project U0835002.

7. REFERENCES

[1] R. M. Storn and K. V. Price, “Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces,” *J. Global Optimization*, vol. 11, pp. 341-359, 1997.

[2] R. Storn, “System design by constraint adaptation and differential evolution”, *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 1, pp. 22-34, Apr. 1999.

[3] M. Vasile, E. Minisci,and M. Locatelli, “An Inflationary Differential Evolution Algorithm for Space Trajectory Optimization”, *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 267-281, Apr. 2011

[4] S. Das, and P. N. Suganthan, “Differential Evolution: A Survey of the State-of-the-Art”, *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[5] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, " Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, Dec. 2006.

[6] J. Q. Zhang, and A. C. Sanderson, "JADE：Adaptive Differential Evolution with Optional External Archive", *IEEE Transactions on Evolutionary Computation*, Vol. 13, No. 5, 2009, pp. 945-958.

[7] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential Evolution Algorithm with Strategy Adaption for Global Numerical Optimization," *IEEE Transactions on Evolutionary Computation*, Vol. 13, No. 2, pp. 398-417. 2009.

[8] T. Zhenguo, L. Yong, "A robust stochastic genetic algorithm (StGA) for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol.8, no.5, pp. 456- 470, Oct. 2004

[9] K. Krishnakumar, R. Swaminathan, S. Garg, and S. Narayanaswamy, "Solving large parameter optimization problems using genetic algorithms", *Proc. Guidance, Navigation, Contr. Conf.*, pp.449-460. 1995.

[10] J. H. Zhong and J. Zhang, "Adaptive Multi-objective Differential Evolution with Stochastic Coding Strategy," in *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation (GECCO 2011)*, pp. 665-672. July 12 - 16, 2011.

[11] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp.280-295, June 2006.

[12] Y. Wang, Z. Cai and Q. Zhang, "Differential Evolution with Composite Trial Vector Generation Strategies and Control Parameters," *IEEE Trans on Evolutionary Computation*, Vol.15, No.1, pp55-66, 2011.

[13] N. Hansen and A. Ostermeier, "Completely derandomized self-adaption in evolution strategies," *Evolut. Comput.*, vol. 9, no. 2, pp.159-195, 2001.

[14] N. Hansen, "Adaptive encoding: How to render search coordinate system invariant," In G. Rudolph, T. Jansen, S. Lucas, C. Poloni, N. Beume, (eds.) PPSN 2008. LNCS, vol.5199, pp.205-214, Springer, Heidelberg (2008).

[15] B. Yuan and M. Gallagher, "Experimental results for the special session on real-parameter optimization at CEC 2005: a simple, continuous EDA," In *Proc. of Congress on Evolutionary Computation (CEC 2005)*, vol. 2, pp. 1792–1799, 2005.

[16] B. Yuan, M. Gallagher, "On the importance of diversity maintenance in estimation of distribution algorithms," *In Proc. of the Genetic and Evolutionary Computation Conference-GECCO*-2005, ACM, New York, USA, pp.719-726.

[17] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC2005 Special Session on Real-Parameter Optimization," *Nanyang Technol. Univ., Singapore, KanGAL Rep. 2005005*, May 2005.

[18] W. N. Chen and J. ZHANG, "Ant Colony Optimization Approach to Grid Workflow Scheduling Problem with Various QoS Requirements", *IEEE Transactions on Systems, Man, and Cybernetics--Part C: Applications and Reviews,* Vol. 31, No. 1,pp.29-43,Jan 2009.

[19] J. ZHANG, and et al., "Implementation of a Decoupled Optimization Technique for Design of Switching Regulators Using Genetic Algorithms," *IEEE Transactions on Power Electronic*. Vol.16, No.5 Nov. 2001.

[20] J. ZHANG, and et al., "Evolutionary Computation Meets Machine Learning: A Survey", *IEEE Computational Intelligence Magazine*, pp.68-75, NOVEMBER 2011.

[21] Z. H. Zhan, and et al., "Adaptive Particle Swarm Optimization", *IEEE Transactions on Systems, Man, and Cybernetics--Part B*. VOL. 39, NO. 6, Dec 2009, Page 1362-1381.

[22] J. ZHANG, H. Chung and W. L. LO, "Clustering-Based Adaptive Crossover and Mutation Probabilities for Genetic Algorithms", *IEEE Transactions on Evolutionary Computation* Vol.11, No.3, June 2007，Page. 326-335.

[23] W. N. Chen, and et al., "A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems", *IEEE Transactions on Evolutionary Computation*, Vol.14, No.2, pp.278-300, April 2010.

[24] X. M. Hu, J. ZHANG, Y. Li, and H. Chung, "SamACO: Variable Sampling Ant Colony Optimization Algorithm for Continuous Optimization", *IEEE Transactions on Systems, Man, and Cybernetics--Part B*, VOL. 40, NO. 6, pp.1555-1566, Dec 2010.